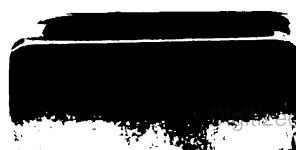
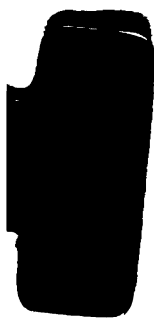

This is a reproduction of a library book that was digitized by Google as part of an ongoing effort to preserve the information in books and make it universally accessible.

GoogleTM books

<https://books.google.com>







ENGR. LIBR. APR 30 1993

DATE DUE

AUG 27 1998		
AUG 11 2003		
GAYLORD		PRINTED IN U.S.A.

*NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY/
NATIONAL COMPUTER SECURITY CENTER*

14TH NATIONAL COMPUTER SECURITY CONFERENCE

**October 1-4, 1991
Omni Shoreham Hotel
Washington, D.C.**



**PROCEEDINGS
VOLUME I**

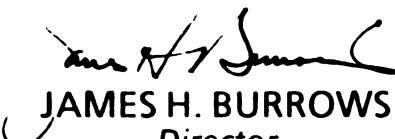
**Information Systems Security:
Requirements & Practices**

Welcome!

The National Computer Security Center (NCSC) and the Computer Systems Laboratory (CSL) are pleased to welcome you to the Fourteenth Annual National Computer Security Conference. We believe that the Conference will stimulate a vital and dynamic exchange of information and foster an understanding of emerging technologies.

The theme for this year's conference, "Information Systems Security: Requirements & Practices," reflects the continuing importance of the broader information systems security issues facing us. At the heart of these issues are two items which will receive special emphasis this week -- Information Systems Security Criteria (and how it affects us) and Education, Training, and Awareness. We are working together, in the Government, Industry, and Academe, in cooperative efforts to improve and expand the state-of-the-art technology to information systems security. This year we are pleased to present a new track emphasizing the integration of information security solutions. These presentations will provide you with some thoughtful insights as well as innovative ideas in developing your own solutions. Additionally, we will be presenting an educational program which addresses the automated information security responsibilities. This educational program will refresh us with the perspectives of the past, and will project directions of the future.

We firmly believe that security awareness and responsibility are the cornerstone of any information security program. For our collective success, we ask that you reflect on the ideas and information presented this week; then share this information with your peers, your management, your administration, and your customers. By sharing this information, we will develop a stronger knowledge base for tomorrow's foundations.


JAMES H. BURROWS
Director
Computer Systems Laboratory


PATRICK R. GALLAGHER, JR.
Director
National Computer Security Center

Engr
QA
76
.9
A25
N27
1991
v.1

Conference

Dr. Marshall Abrams

James P. Anderson

Jon Arneson

Devolyn Arnold

James Arnold

Al Arsenault

V.A. Ashby

David Balenson

Dr. D. Elliott Bell

James W. Birch

W.Earl Boebert

Dr. Martha Branstad

Dr. John Campbell

Lisa Carnahan

R.O. Chester

David Chizmadia

Dorothea deZafra

Donna Dodson

Karen Doty

Dr. Deboah Downs

Jared Dreicer

Ellen Flahavin

Daniel Gambel

L. Dain Gary

Virgil Gibson

Dennis Gilbert

Irene Gilbert

Captain James Goldston, USAF

Dr. Joshua Guttman

Douglas Hardie

Ronda Henning

Dr. Harold Highland, FICS

Jack Holleran

Hilary H. Hosmer

Russell Housley

Howard Israel

Dr. Sushil Jajodia

Wayne Jansen

The MITRE Corporation

J.P.Anderson Company

National Institute of Standards and Technology

Department of Defense

Department of Defense

Air Force Academy

The MITRE Corporation

Trusted Information Systems, Inc.

Trusted Information Systems, Inc.

Secure Systems, Inc.

Secure Computing Technology Corporation

Trusted Information Systems, Inc.

Department of Defense

National Institute of Standards and Technology

Martin Marietta

Department of Defense

Public Health Service

National Institute of Standards and Technology

CISEC

The AEROSPACE Corporation

Los Alamos National Laboatory

National Institute of Standards and Technology

Grumann Data Systems

Mellon National Bank

Grumann Data Systems

National Institute of Standards and Technology

National Institute of Standards and Technology

AFCSC

The MITRE Corporation

Unisys Corporation

Harris Corporation

Compulit, Inc.

National Computer Security Center

Data Security, Inc.

XEROX Information Systems

AT&T Bell Laboratories

George Mason University

National Institute of Standards and Technology

Referees

Carole Jordan
 Dr. Maria M. King
 Leslee LaFountain
 Steven LaFountain
 Paul A. Lambert
 Dr. Carl Landwehr
 Robert Lau
 Dr. Theodore M.P. Lee
 Steven B. Lipner
 Teresa Lunt
 Dr. William V. Maconachy
 Sally Meglathery
 Dr. Jonathan Millen
 Warren Monroe
 William H. Murray
 Noel Nazario
 Ruth Nelson
 Peter Neumann
 J.D. Nichols
 Steven Padilla
 Nick Pantiuk
 Donn Parker
 Richard Pethia
 Dr. Charles Pfleeger
 Kenneth Rowe
 Professor Ravi Sandhu
 Marvin Schaefer
 Dr. Roger R. Schell
 Emilie J. Siarkiewicz
 Suzanne Smith
 Brian Snow
 Professor Eugene Spafford
 Mario Tinto
 James Tippet
 Eugene Troy
 LTC. R. Vaughn, USA
 Grant Wagner
 Kenneth vanWyk
 Howard Weiss
 Roy Wood
 Carol Worden

*Defense Investigative Service
 The AEROSPACE Corporation
 Department of Defense
 Department of Defense
 Motorola GEG
 Naval Research Laboratory
 Department of Defense
 Trusted Information Systems, Inc.
 Digital Equipment Corporation
 SRI International
 National Security Agency
 ISSA
 The MITRE Corporation
 Hughes Aircraft
 Deloitte & Touche
 National Institute of Standards and Technology
 GTE
 SRI International
 Independent Consultant
 SPARTA
 Grumann Data Systems
 SRI International
 Carnegie Mellon University
 Trusted Information Systems, Inc.
 Department of Defense
 George Mason University
 Trusted Information Systems, Inc.
 GEMINI
 Rome Air Defense Center
 Los Alamos National Laboatory
 Department of Defense
 Purdue University
 Department of Defense
 Department of Defense
 National Institute of Standards and Technology
 U.S. Naval Academy
 Department of Defense
 Carnegie Mellon University
 SPARTA
 Department of Defense
 State of Minnesota*

14th National Computer Security Conference

Table of Contents

x Authors Cross Index

Tutorials

- 1 From Tuples to Trusted Subjects to TDI: A Brief Tutorial on Trusted Database Management Systems
John R. Campbell, National Security Agency
- 13 Tutorial Series on Trusted Systems
Joel E. Sachs, Dr. William F. Wilson, Arca Systems, Inc.

PAPERS (refereed)

- 15 Accreditation Strategy for the Air Force Satellite Control Network (AFSCN)
Lt Col William Price, USAF, Air Force Space Command
Michael O'Neill, Frank White, CTA, Inc.
- 25 An Analysis of Application Specific Security Policies
Daniel F. Sterne, Martha Branstad, Trusted Information Systems, Inc.
Brian Hubbard, SPARTA, Inc.
Barbara Mayer, Atlantic Research Corporation
Dawn Wolcott, MITRE Corporation
- 37 Another Factor in Determining Security Requirements for Trusted Computer Applications
David Ferraiolo, National Institute of Standards and Technology
Karen Ferraiolo, Grumman Data Systems
- 45 Apparent Differences Between the U.S. TCSEC and the European ITSEC
Dr. Martha Branstad, Dr. Charles Pfleeger,
Trusted Information Systems, Inc.
Dr. David Brewer, Gamma Secure Systems, Ltd.
Mr. Christian Jahl, Mr. Helmut Kurth, IAGB Software Technology
- 59 Auditing of Distributed Systems
D. Banning, G. Ellingwood, C. Franklin, C. Muckenhirn, D. Price,
SPARTA, Inc.
- 69 Building a Multi-Level Application on an Untrusted DBMS in a UNIX System V/MLS Environment - A Project's Experience
David S. Crawford, Canadian Department of National Defence
- 78 Building a Multi-Level Secure TCP/IP
Deborah A. Fletcher, Brian K. Yasaki, The Wollongong Group
Ron L. Sharp, AT&T Bell Laboratories
- 88 The Cascade Problem: Graph Theory Can Help
John A. Fitch, III, Lance J. Hoffman, George Washington University

- 101 **A Case Study for the Approach to Developing a Multilevel Secure Command and Control Information System**
James Obal, Supreme Allied Commander Atlantic
William Grogan, Contel Federal Systems
- 110 **Contractors and Computer Security--Awareness, Education, and Performance**
Ronald E. Brunner, Ronald G. Brunner & Associates
- 120 **Covert Channel Analysis Planning for Large Systems**
Lee Badger, Trusted Information Systems, Inc.
- 137 **Dealing With a Malicious Logic Threat: A Proposed Air Force Approach**
Howard L. Johnson, Information Intelligence Sciences
Chuck Arvin, Earl Jenkinson, CTA, Inc.
Captain Bob Pierce, USAF, Electronic Security Command
- 147 **Developing Applications on LOCK**
Richard O'Brien, Clyde Rogers, SCTC
- 157 **The Development of a Low-To-High Guard**
Michelle J. Gosselin, MITRE Corporation
- 167 **DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype**
Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che-Lin Ho,
Karl N. Levitt, Biswanath Mukherjee, University of California, Davis
Stephen E. Smaha, Steven R. Snapp, Haystack Laboratories, Inc.
James Brentano, Pacific Gas and Electric Company
Lt. Tim Grance, USAF, Daniel M. Teal, USAF,
United States Air Force Cryptologic Support Center
Douglass L. Mansur, Lawrence Livermore National Laboratories
- 177 **A Distributed Implementation of the Transform Model**
Ravi S. Sandhu, Gurpreet S. Suri, George Mason University
- 188 **Employee Privacy and Intrusion Detection Systems: Monitoring on the Job**
Lorrayne J. Schaefer, The MITRE Corporation
- 195 **Experience of Commercial Security Evaluation**
Peter Fagan, Julian Straw, Secure Information Systems Limited
- 205 **Experiences in Multi-Level Security on Distributed Architectures**
Karl A. Siil, AT&T Bell Laboratories
- 215 **An Expert System Application for Network Intrusion Detection**
Kathleen A. Jackson, David H. DuBois, Cathy A. Stallings,
Los Alamos National Laboratory
- 226 **Formal Verification of a Network Security Device: A Case Study**
Hicham N. Adra, William Sandberg-Maitland,
CGI Information Systems & Management Consultants
- 237 **A Framework for Advancing Integrity Standardization**
Terry Mayfield, Stephen R. Welke, John M. Boone,
Catherine W. McDonald, Institute for Defense Analyses-

- 246 **A Framework for Developing Accreditable MLS AISs**
R. K. Bauer, J. Sachs, M. L. Weidner, W. F. Wilson, Arca Systems, Inc.
- 257 **Generalized Framework for Access Control: Towards Prototyping the ORGCON Policy**
Marshall D. Abrams, Jody Heaney, Osborne King, Leonard LaPadula, Manette Lazear, Ingrid Olson, The MITRE Corporation
- 267 **Honest Databases That Can Keep Secrets**
Ravi Sandhu, Sushil Jajodia, George Mason University
- 283 **Identifying and Controlling Undesirable Program Behaviors**
Maria M. King
- 295 **Improvement of Data Processing Security by Means of Fault Tolerance**
Gilles Trouessin, Yves Deswarte, Jean-Charles Fabre, LAAS-CNRS & INRIA
Brian Randell, Computing Laboratory, The University Newcastle upon Tyne
- 305 **Information Security: Can Ethics Make a Difference**
Corey D. Schou, John A. Kilpatrick, Idaho State University
- 313 **Information Security Risk Analysis and Risk Management: Which Approach?**
Professor J.H.P. Eloff, K.P. Badenhorst, Rand Afrikaans University
- 328 **Information Systems Security: A Comprehensive Model**
Capt. John R. McCumber, USAF, Joint Staff, the Pentagon
- 338 **Integrating B2 Security into a UNIX System**
Kevin Brady, UNIX System Laboratories, Inc.
- 347 **Knowledge Based Computer Security Advisor**
William Huntman, M. B. Squire, Los Alamos National Laboratory
- 357 **The Logistics of Distributing a Smart Token**
Dawn Brown, Department of Defense
- 362 **A Method to Detect Intrusive Activity in a Networked Environment**
L. Todd Heberlein, Biswanath Mukherjee, Karl Levitt, University of California
- 372 **Model Based Intrusion Detection**
Thomas D. Garvey, Teresa F. Lunt, SRI International
- 386 **Notification: A Practical Security Problem in Distributed Systems**
Vijay Varadharajan, Hewlett-Packard Laboratories
- 397 **Output Perturbation Techniques for the Security of Statistical Databases**
Kasinath C. Vemulapalli, Elizabeth A. Unger, Kansas State University
- 407 **An Overview of Informix-Online/Secure**
Rammohan Varadarajan, Informix Software, Inc.
- 417 **Peeling the Viral Onion**
Russell Davis, Planning Research Corporation, Inc.

- 427 **Practical Models for Threat/Risk Analysis**
Mark W.L. Dennison, Kalman C. Toth
CGI Information Systems & Management Consultants, Inc.
- 436 **Predicate Differences and the Analysis of Dependencies in Formal Specifications**
D. Richard Kuhn, National Institute of Standards and Technology
- 446 **Preventing Weak Password Choices**
Eugene H. Spafford, Purdue University
- 456 **Putting Policy Commonalities to Work**
D. Elliott Bell, Trusted Information Systems, Inc.
- 472 **Reconciling CMW Requirements with Those of X11 Applications**
Glenn Faden, Sun Microsystems, Inc.
- 480 **Restating the Foundations of Information Security**
Donn Parker, SRI International
- 494 **The Role Of Network Security In A Methodology For Information Security Design And Implementation**
Professor J.H.P. Eloff, Mr. A.J. Nel, Rand Afrikaans University
- 505 **A Secure European System for Applications in a Multi-vendor Environment (The SESAME Project)**
T. A. Parker, ICL Secure Systems
- 514 **A Secure Quorum Protocol**
Masaaki Mizuno, Mitchell L. Nielsen, Kansas State University
- 524 **Security Guidance for VAX/VMS Systems**
Debra L. Banning, SPARTA, Inc.
- 533 **Sneakernet: Getting a Grip on the World's Largest Network**
Captain James B. Hiller, USAF, Space and Warning Systems Center
- 543 **A Socio-Technical Analysis of a USA National Computer Security Conference**
Stewart Kowalski, Stockholm University & Royal Institute of Technology
- 533 **Standardized Certification**
Captain Charles R. Pierce, USAF, Air Force Cryptologic Support Center
- 563 **A Strategic Framework For Information Security Management**
Rolf Moulton, BP America
Santosh Misra, Cleveland State University
- 572 **A System Security Engineering Process**
J. D. Weiss, AT&T Bell Laboratories
- 582 **Teaching Computer Systems Security in an Undergraduate Computer Science Curriculum**
Alfred W. Arsenault, Captain Gregory B. White, USAF,
U. S. Air Force Academy
- 598 **Toward Certification, A Survey of Three Methodologies**
Captain Charles R. Pierce, USAF, Air Force Cryptologic Support Center

- 608 **Trusted Distributed Computing: Using Untrusted Network Software**
E. John Sebes, Richard J. Feiertag, Trusted Information Systems
- 619 **Trusting X: Issues in Building Trusted X Window Systems or What's not
 Trusted About X?**
Jeremy Epstein, TRW Systems Division
Jeffrey Picciotto, MITRE Corporation
- 630 **Using Existing Management Processes to Effectively Meet the Security Plan
 Requirement of the Computer Security Act: The IRS Experience**
Richard A. Stone, Joseph Scherer, Internal Revenue Service
- 634 **Viruses in an OS/2 Environment: Remembrances of Things Past and a
 Harbinger of Things to Come**
Kevin P. Haney, National Institutes of Health
- 644 **Why Does Trusted Computing Cost So Much?**
Susan Heath, Phillip Swanson, Daniel Gambel, Grumman Data Systems

PANEL Executive Summaries (unrefereed)

- 654 **PANEL: Acquiring Computer Security Services and Integrating Computer
 Security and ADP Procurement**
Dennis Gilbert, National Institute of Science and Technology
Barbara Guttman, National Institute of Science and Technology
- 655 **PANEL: Compartmented Mode Workstation(CMW) Program Overview**
Steven Schanzer, Moderator, Defense Intelligence Agency
- 658 **PANEL: The Computer Emergency Response Team System (CERT
 System)**
E. Eugene Schultz, Lawrence Livermore Laboratory
Richard Pethia, Software Engineering Institute, Carnegie Mellon University
- 663 **PANEL: Computer Security Management and Planning**
Christopher Bythewood, National Computer Security Center
- 664 **PANEL: Cracking the Cracker Problem**
Dorothy E. Denning, Moderator, Georgetown University
- 665 **The Role of Technology**
Matt Bishop, Dartmouth College
- 666 **PANEL: Electronic Dissemination of Computer Security Information
 Executive Summary**
Marianne Swanson, National Institute of Science and Technology
- 667 **What Can Dockmaster Offer You?**
Cindy Hash, Department of Defense
- Session: Guidelines & Evaluations**
- 669 **Towards Mutual Recognition of Security Evaluations**
Andrea Arnold, Digital Equipment Corp
Cornelia Persy, SIEMENS
Gottfried Sedlak, IBM

- 674 **PANEL:** Fielding COTS Multilevel Security Solutions: The Next Step
James Litchko, Trusted Information Systems Inc.
- 675 **PANEL:** Inference and Aggregation in Multilevel Databases: Research Directions
Teresa F. Lunt, Moderator, SRI International
- 676 Detecting and Evaluating Inference Channels
Thomas D. Garvey, SRI International
- 679 Inference Prevention in Databases: Data Design vs. Query Processing
Catherine Meadows, Naval Research Laboratory
- 680 Challenges in Addressing Inference and Aggregation
LTC. Gary Smith, USA, National Defense University
- 681 Approaches to Handling the Inference Problem
Bhavani Thuraisingham, The MITRE Corporation
- 684 **PANEL:** Military and Telecommunications Security: Specialized Methods
Richard Lefkon, Moderator, New York University
- 685 Malicious Code Prevention for Embedded Computer Weapon Systems
Debra L Banning, Gail M. Ellingwood, SPARTA
- 689 Computer Viruses as Electronic Warfare
Myron Cramer, Booz-Allen & Hamilton
- 690 Preventing Virus Insertion Through Switches
Ed Fulford, Northern Telecom
- 693 Nuclear Disaster and The Millennium Horse
Richard Lefkon, New York University
- Session: National Issues**
- 695 Reduced Defense Spending Increases the Need for Trusted Systems
Carole S. Jordan, Defense Investigative Service
- 696 **PANEL:** 1991: A Year of Progress in Trusted Database Systems
John R. Campbell, Moderator, National Security Agency
- 698 Recent Developments in Some Trusted Database Management Systems
Bhavani Thuraisingham, The MITRE Corporation
- 701 Oracle and Security: Year in Review 1990-91
Linda L. Vetter, Oracle Secure Systems
- 704 1991 SYBASE Secure Products: Executive Summary
Helena B. Winkler-Parenty, SYBASE
- 706 **PANEL:** Requirements and Experiences
Dennis Gilbert, National Institute of Science and Technology
- 708 **PANEL:** Risk Management
Irene Gilbert, National Institute of Science and Technology
- 709 **PANEL:** Specifying, Procuring, and Accrediting MLS System Solutions
Joel E. Sachs, Arca Systems, Inc.
- 714 **PANEL:** Trusted Applications in the Real World
Stephen Walker, Trusted Information Systems Inc.
- 715 **PANEL:** Winning Strategies in Information Systems Security Education, Training, and Awareness
W. V. Maconachy, Moderator, Department of Defense

Authors Cross Index

Abrams, Marshall D.	257	Franklin, C.	59
Adra, Hicham N.	226	Fulford, E.	690
Arsenault, Alfred W.	582	Futcher, Deborah A.	78
Arvin, Chuck	137	Gambel, Daniel	644
Arnold, Andrea	669	Garvey, T. D.	372, 676
Badenhorst, K.P.	313	Gilbert, Dennis	654, 706
Badger, Lee	120	Gilbert, Irene	708
Banning, D. -	59, 524, 685	Goan, Terrance L.	167
Bauer, R. K.	246	Gosselin, Michelle J.	157
Bell, D. Elliott	456	Grance, Tim, Lt.	167
Bishop, M.	665	Grogan, William	101
Boone, John M.	237	Guttman, Barbara	654
Brady, Kevin	338	Haney, Kevin P.	634
Branstad, Martha	25, 45	Hash, Cindy	667
Brentano, James	167	Heaney, Jody	257
Brewer, David	45	Heath, Susan	644
Brown, Dawn	357	Heberlein, L. T.	167, 362
Brunner, Ronald E.	110	Hiller, J. B. ,Capt	533
Bythewood, C	663	Ho, Che-Lin	167
Campbell, John R.	1, 696	Hoffman, Lance J.	88
Cramer, Myron	689	Hubbard, Brian	25
Crawford, David S.	69	Hunteman, William	347
Davis, Russell	417	Jackson, Kathleen A.	215
Denning, Dorothy E.	664	Jahl, Christian	45
Dennison, Mark W.L.	427	Jajodia, Sushil	267
Deswarte, Yves	295	Jenkinson, Earl H.	137
Dias, Gihan V.	167	Johnson, Howard	137
DuBois, David H.	215	Jordan, Carole	695
Ellingwood, G. M.	59, 685	Kilpatrick, John A.	305
Eloff, J.H.P.	313, 494	King, Maria M.	283
Epstein, Jeremy	619	King, Osborne	257
Fabre, Jean-Charles	295	Kowalski, Stewart	543
Faden, Glenn	472	Kuhn, D. Richard	436
Fagan, Peter	195	Kurth, Helmut	45
Feiertag, Richard J.	608	LaPadula, Leonard	257
Ferraiolo, David	37	Lazear, Manette	257
Ferraiolo, Karen	37	Lefkon, Richard	684, 693
Fitch, John A., III,	88	Levitt, Karl N.	167, 362

Authors Cross Index

Litchko, James	674	Schultz, E. Eugene	658
Lunt, Teresa F.	372, 675	Sebes, E. John	608
Maconachy, W. V.	715	Sedlak, Gottfried	669
Mansur, Douglass L.	167	Sharp, Ron L.	78
Mayer, Barbara	25	Siil, Karl A.	205
Mayfield, Terry	237	Smaha, Stephen E.	167
McCumber, John R.	328	Smith, Gary	680
McDonald, C. W.	237	Snapp, Steven R.	167
Meadows, Catherine	679	Spafford, Eugene H.	446
Misra, Santosh	563	Squire, M. B.	347
Mizuno, Masaaki	514	Stallings, Cathy A.	215
Moulton, Rolf	563	Sterne, Daniel F.	25
Muckenhirn, C.	559	Stone, Richard A.	630
Mukherjee, B	167, 362	Straw, Julian	195
Nel, A. J.	494	Suri, Gurpreet S.	177
Nielsen, Mitchell L.	514	Swanson, Marianne	666
Obal, James	101	Swanson, Phillip	644
O'Brien, Richard	147	Teal, Daniel M., Lt.	167
Olson, Ingrid	257	Thuraisingham, B.	681, 698
O'Neill, Michael	15	Toth, Kalman C.	427
Parker Donn B.	480	Trouessin, Gilles	295
Parker, T. A.	505	Unger, Elizabeth A.	397
Pethia, Richard	658	Varadarajan, R.	407
Persy, Cornelia	669	Varadharajan, Vijay	386
Pfleeger, Charles	45	Vemulapalli, K. C.	397
Picciotto, Jeffrey	619	Vetter, Linda	701
Pierce, C. R. Capt	137, 533, 598	Walker, Stephen	714
Price, D.	59	Weidner, M. L.	246
Price, William, Lt Col	15	Weiss, J. D.	572
Randell, Brian	295	Welke, Stephen	237
Rogers, Clyde	147	White, Frank	15
Sachs, Joel E.	13, 246, 709	White, G. B., Capt	582
Sandberg-Maitland, W.	226	Wilson, W. F., Dr.	13, 246
Sandhu, Ravi S.	177, 267	Winkler-Parenty, H.	704
Schaefer, Lorrayne J.	188	Wolcott, Dawn	25
Schanzer, Steven	655	Yasaki, Brian K.	78
Scherer, Joseph	630		
Schou, Corey D.	305		

FROM TUPLES TO TRUSTED SUBJECTS TO TDI: A BRIEF TUTORIAL ON TRUSTED DATABASE MANAGEMENT SYSTEMS

John R. Campbell

National Security Agency

9800 Savage Road

Fort George G. Meade, Maryland 20755-6000

301-859-4387

INTRODUCTION

Over ninety percent of the nation's mainframes and most minicomputers and microcomputers contain database management systems (DBMS). Our most critical data, including defense, intelligence, law enforcement, social welfare, and financial data, are stored on such systems. Applications ranging from financial systems to national defense mechanisms depend on the security of these systems.

The building of these systems and the construction of applications for these systems is a multi-billion dollar industry. Yet, to date, little has been done to secure database management systems. Vendors have emphasized performance and ease of use, with security being an afterthought. Often any security included in the database system is done without regard to consistency with the existing operating system security mechanisms.

This lack of interest in DBMS security, however, is starting to change. The threat to data, due to nondisclosure, lack of integrity and unavailability, is being addressed. Trusted products are being introduced commercially. Vendors and potential vendors of trusted products include Atlantic Research Corporation (ARC), DEC, Informix, Infosystems Technology, Ingres, Oracle, Sybase and Teradata. A second significant gain in 1991 is the completion of the Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria (TDI). The TDI extends the evaluation classes of the Trusted Computer System Evaluation Criteria (TCSEC) to trusted applications in general, and to database management systems in particular. The evaluation of trusted database systems has been started by the National Computer Security Center. As of this writing, two products were under evaluation; others are in preparation for evaluation.

Database security is maturing somewhat as a discipline. Some very tough issues are being examined and understood. For example, we know a lot more about the causes of, the problems associated with and the potential solutions for polyinstantiation now than when we put it in a contract to force people to look at the problem. There has been good research and development in this area. For example, Rome Labs is sponsoring the development of a B2 system, Oracle is examining the relationship between integrity and confidentiality and we are supporting the development of a trusted database system with A1 Mandatory Access Control. Research is being done, among other things, on distributed, multimedia, and object-oriented trusted database systems.

This tutorial gives the background, describes the issues and offers some proposed solutions for database security. The title was deliberately chosen. The "tuple" is a record instance or row in a table. I will briefly discuss database systems, and, more specifically, relational database systems, as systems based on this model are currently the most widely used systems. "TCB- Subsets" is a key concept in database security because, as an application, it sits on other software, perhaps an operating system. The concept permit efficient evaluation of trusted software in a very high skill, labor intensive process. The "TDI" is an important work, not only because it aids evaluators of trusted database systems but because it deals with layering and applications in general.

DATABASES AND DATABASE SECURITY

In the August 1989 issue of Computer [JACO89], the reviewer of a book on computer security makes two comments, both I especially agree with for database security. First, he states that the entire field of computer security has substantial weaknesses. This is especially true for database security. For example, trusted distributed database management systems present many unanswered questions. There is no general theory of control for inference and aggregation, although there are some application specific controls. Verification tools are weak. There are many other unanswered issues.

Second, the reviewer states that the field of computer security is quickly evolving. Again, this is especially true for database security. It is junior to operating system security because it often has to depend on a trusted operating system. But, until now, there were few trusted operating system products. Several years ago, we talked about the possibility of trusted database systems. Today there are at least eight prototypes, half of which are commercial quality. Truly the field is rapidly evolving.

What is a database? Date [DATE86] defined them as collections "of stored operational data used by the application systems of some particular enterprise." The operational data could include product, account, patient, student or planning data. It does not include input or output data, work queues, temporary results or any purely transient information. Databases are increasing in complexity. The data can now be pictures, rules, or derived information.

What is a database management system? Date [DATE86] defines these as systems that provide users with a view of the database that is elevated somewhat above the hardware level, and support user operations such as SQL operations that are expressed in terms of that higher level view. "SQL", or Structured Query Language, is a high level query language that contains both data manipulation and data definition features. It also contains data control features, "grant" and "revoke", for example. Database management systems are also increasing in complexity. Some database systems have natural language, rule manipulation and other artificial intelligence components. Some are distributed. Database security must meet these challenges.

WHY DATABASE SECURITY IS IMPORTANT

Database security is important because databases are so very important. The DoD, the intelligence, financial, law and social services communities depend on them to be safe and correct. Two billion dollars was spent in 1987 on database systems. It is estimated that six billion will be spent in 1992. Applications for these systems cost many times more. Ninety percent of mainframes use database systems

Database security is important because even with a trusted operating system underneath, data is at risk if you are not using a trusted database system. One problem is granularity. Operating systems usually protect at the file level. Databases need finer granularity such as table or relation, row or tuple, or even element. Database systems can provide protection at these levels of granularity. In addition, different discretionary security policies are often desired for database systems that restrict access to specific data through specific database operations, such as insert, update, retrieve and delete. Such controls are not available in operating systems.

Database security is important because database systems are the most widely used class of application on computer systems. As such, much learned about database systems, such as trusted operating system interface, can be transferred to our knowledge of securing other applications.

Database security includes data integrity. Data integrity is important because a database is useless if the information you get out of it is wrong. The importance of integrity has long been realized by database system vendors and they have provided some capabilities to preserve integrity. However, the active data dictionary, where data constraints are recorded and enforced, is a relatively new concept.

Concentrated work done now on both database security and integrity is important because the list of problems is constantly growing. In addition to the vanilla stand-alone commercial database systems, which by themselves are quite complex, we now have commercial expert, multimedia and/or distributed database systems. These, plus intelligent, temporal, historical and object-oriented databases add to the complexity of the problem.

SOME ARCHITECTURES AND MODELS

Database systems employ different architectures and these present differing problems. Database machines are computers dedicated to database activities. All data is stored on these machines. Host computers issue queries to the database machine. This machine processes the query, finds and manipulates the data and returns the answer. Under this configuration, the machine's operating system (OS) and database system are usually one; therefore the OS/DBMS interface does not exist.

In host-based DBMSs, the OS/DBMS interface is a serious problem. Here the DBMS runs on a general purpose computer that, in addition to the DBMS, usually has other applications running on it. Some vendors want to port their database management systems to as many computers as possible. How is this accomplished in

an efficient yet secure manner? There are no standard security interfaces. Therefore, in order to be truly portable, DBMS vendors may choose to duplicate the security functionality of the operating system and not use the security functionality of the operating system. This avoids having to make several custom interfaces, but it increases the complexity and size of DBMS security components. Also, if the DBMS is trusted, its interactions with the operating system trusted computing base must be controlled.

Client-server architectures are becoming popular. Data could be stored in a database on a larger computer or server. The data is then usually accessed by smaller computers called clients. Many users on personal computers or workstations could then efficiently access a large database on a larger server. The clients and servers are connected by perhaps a LAN. A problem is that the system: clients, server and LAN must recognize and protect security labels. This recognition may not be easy, especially if each component comes from a different vendor.

Finally, distributed database systems have added additional complexities to the security problem. The data in these system may have different physical locations, may be on heterogeneous nodes and may be redundant. How do you audit? How do you identify and authorize? How do you assure the integrity of redundant information? What form of concurrency do you use? We are seeing repeatedly that data integrity conflicts with confidentiality. How do you get both? What are the tradeoffs? We are beginning to address these issues.

The DBMS model used may also affect security. Is the model relational, network, hierarchical, object-oriented or other. A secure entity relationship study reported that it was easier to secure a system based on an entity relationship model than a relational model. One reason he gave was that he had the freedom to choose the entity-relation model that could best contain security. There is no standard model. The relational model, however, has solidified into almost a standard, a standard where initially security was not considered, and therefore retrofitting security, especially multilevel security, is difficult. While this is still a research topic, object-oriented systems also appear to be easier to secure.

WHAT IS SECURITY?

Security, in some areas, has been equated only with nondisclosure. A system is secure if you can prevent unauthorized users from reading sensitive information. However, we also include integrity and availability or denial of service components in this definition. If you can modify or destroy my data or otherwise deny me access to my data, then the data is not secure. Consequently, our definition agrees with what the Strategic Defense Initiative calls "security *" which includes nondisclosure, data integrity and availability.

Our definition also includes ease-of-use as a requirement for "security". If the user or security administrator finds a system too difficult to use because of security, then the security features will not be used. This is easily done as most security features on database systems are optional. A goal then is to build systems that appear to be very similar to vanilla systems, that use standards such as the Structured Query Language (SQL), and that are compatible as possible to previous databases and database systems.

WHAT IS INTEGRITY?

We've seen a list of 150 definitions of integrity. One we like is "sound, unimpaired or perfect condition" [NCSC88a]. Is what you get out of the database what you put in it?

Three integrity components have been noted. The Department of Defense Trusted Computer Evaluation Criteria (TCSEC or "Orange Book") [DOD85] recognizes two types, label integrity and system integrity. Label integrity assures that the security labels accurately represent the classifications of subjects or objects with which they are associated. System integrity is the correct operation of the on-site hardware and firmware elements of the TCB. This "TCB" is the totality of protection mechanisms within a computer which is responsible for enforcing a security policy.

What the TCSEC doesn't explicitly mention, the third integrity component, data integrity, is something very important to DBMS users. We define it as the "property that data has not been exposed to accidental or malicious alteration or destruction [NCSC88b].

DATA INTEGRITY IMPLEMENTATION

Data integrity may be implemented as part of the overall security policy. For example, the Biba integrity model [BIBA77] may be implemented with Bell-LaPadula nondisclosure model [BELL73] to produce a model that enforces both integrity and security. SeaView did this using a modified Biba model and Bell-LaPadula. The model can then be translated into an operational system.

Even though a security policy may not be explicitly stated, integrity components may exist. Entity integrity, for example, does not permit null primary keys. In general, under referential integrity, foreign keys must reference existing primary keys. Also, integrity constraints and typing may be used. For example, one field or attribute may allow only months of the year, with the first letter capitalized. The system will check that each item entered into this field satisfies these constraints. Both secure recovery and the concept of serializability are also important for data integrity.

Finally, it is important to note that nondisclosure and data integrity may conflict. Referential integrity may enable someone at a lower classification level to know whether something at a higher level exists. Hiding the existence of high data from low users may also require that polyinstantiation be used. Under this concept, multiple data objects with the same name, differentiated by their access class, may exist simultaneously [DENN88]. Is this an integrity violation? And couldn't it cause data integrity problems?

Concurrency controls are integrity controls that enable many users to run their programs and access the database at the same time. They prevent incorrect interactions between transactions. In this way throughput and availability of the database management system are enhanced. Standard controls however, can be

used as signalling channels, thereby harming nondisclosure. This area is a research topic and work is being done.

BREAKDOWN OF THE PROBLEMS

It is useful to break down the database security problem into historical components. Research that has been done in each of these components may be useful in building a secure database system.

The first component is operating system security. Many of the concepts that originated in operating system security are also used in DBMS security. In addition, in the computer system, the DBMS may be layered on top of the OS, may depend on the OS for services and may share the responsibility for security policy enforcement with the OS.

The second component is network security. Network security concepts will be useful in client-server and distributed database work.

Some are handled as database security issues. The problems of inference and aggregation are not unique to database systems. They deal with relationships between data. However, the inference and aggregation problems are exacerbated by database management systems, because these systems are designed to easily manipulate large quantities of data. Some issues, such as granularity, are unique to database security.

Some issues are treated as database security issues because they had to be solved before a trusted database system could be built. Layering and TCB subsets were studied for trusted database systems but they apply to trusted applications in general.

Finally, there are issues that seem to be unique to the distributed DBMS. How do you update replicated data or recover in a secure fashion? These also are research questions.

STANDARDS/INTERPRETATIONS

Several useful standards and interpretations are available. The previously mentioned TCSEC, although traditionally used on stand-alone operating systems, has many concepts applicable to database systems. The Trusted Network Interpretation is a trusted computer/communications network systems interpretation of the TCSEC. Similarly, the TDI will add insight into the evaluation of database management systems and other applications.

TCB SUBSETS

Wouldn't it be of advantage to a vendor who ports a DBMS to many computers and to the evaluator not to have to evaluate the operating system of each target computer with the DBMS? If it can be shown that the DBMS does not interfere with the underlying security mechanisms of the os, then this can happen.

The TCB or Trusted Computing Base is the totality of protection mechanisms in a computer system. The combination of these mechanisms is responsible for enforcing a security policy [DOD85]. A TCB Subset is a logical partition or layer of the TCB that enforces a subset of the security policies and supporting accountability policies enforced by the combined TCB [NCSC89]. With this approach, the TCB is divided into TCB Subsets, and each subset enforces a distinct part of the security policy. Good software engineering would also dictate layering.

A TCB subset M is a set of software and/or firmware and/or hardware that mediates the access of a set S of subjects to a set O of objects on the basis of a stated access control policy P and satisfies the properties:

1. M mediates every access to objects in O by subjects in S;
2. M is tamper resistant; and
3. M is small enough to be subject to analysis and tests, the completeness of which can be assured. [NCSC91]

OTHER CONSIDERATIONS

A Trusted Path has been defined as a mechanism by which a person at a terminal can communicate directly with the TCB. To prevent spoofing, the mechanism cannot be imitated by untrusted software. A trusted path is also needed between the system security officer and the TCB.

In good software engineering, a design and development process that promotes modifiability, efficiency, reliability and understandability [BOOC83] should be used.

Finally appropriate audit mechanisms should be used. The issue is to get the granularity to record needed information while not severely impacting performance. To achieve this balance we have recommended the use of summary audit records to the TDI Chairman/Project Leader. Summary audit records log a count of the accesses for each subject accessing each level/compartment in a relation.

INFERENCE AND AGGREGATION

Inference and aggregation are big security problems. Inference is the derivation of information at a level for which the user is not permitted access by referencing other information to which he has access. In aggregation, the sensitivity level of a collection of data may be higher than the level of any individual datum. Therefore, in either case, the data's security label is not enough to protect the data. Neither is mentioned in the TCSEC. Again, they are not specifically DBMS problems but are aggravated by the DBMS because the DBMS has been built to facilitate the manipulation and combination of data.

AN INFERENCE EXAMPLE

Who makes widgets? The answer is known but it is a secret. Is it company A, B, C, D or E?

It is known that widget makers need lots of water for cooling. Therefore the plant must be on a lake, river, etc. Also, they need lots of fossil fuel. Therefore the plant needs to be on a railroad siding or a barge pier. Finally, widget makers need chemical engineers.

The following additional information has been obtained from databases:

1. Company A is on a lake. Companies D and E are on rivers.
2. Companies A, C and E have railroad sidings.
3. Companies B and E advertise for Chemical Engineers.

Who? E.

INFERENCE/AGGREGATION CONTROLS

To control inference, and yet to keep classifications as low as possible, the applications designer, in a relational system, can classify table linkages or keys, but not the actual data in the tables. Or, the inference problems may be defined and the system could check queries for the problems. Control of aggregation could be done with query response history information. This however, presents a data aging/system performance problem. That is, the more history you have, the better the control, but the longer it takes to scan the history.

SQL STANDARDS CONSIDERATIONS

"SQL" is a data definition and data manipulation language and is currently an ANSI standard. "SQL3", a proposed future ANSI standard, provides for triggers, mechanisms by which a user can affect the consistency of the database. Therefore the impact of SQL on integrity must be considered. Also SQL must be enriched to handle additions of audit, role and security level requirements.

CURRENT IMPLEMENTATIONS

There, fortunately, has been much activity in implementing commercial versions of trusted database systems. The vendors include ARC, Informix, Oracle, Sybase and Teradata. Other trusted systems are being developed.

The most popular implementation is a Trusted Computing Base (TCB) implementation where the DBMS enforces Mandatory Access Control on the DBMS objects. Part of the DBMS is a trusted subject. Performance here is independent of

the number security levels and compartments. Evaluation is more complex and difficult. Sybase and Informix are examples.

In another Trusted Computing Base implementation, the operating system provides the mandatory access control, while both operating system and DBMS may provide discretionary access control. The evaluation should be easier. However, each combination of security level and compartment requires a separate database instance. Performance should decrease with increasing numbers of security level/compartment combinations. Unclassified data may be separately stored as such. Oracle's product offers the choice of either this or the first approach.

The integrity lock approach uses a trusted filter in front of an untrusted DBMS. The filter mediates all accesses between the users and the database, and performs trusted downgrades where necessary when providing at lower security levels with data from the database. [WINK89] A trusted operating system at least the filter level and B1 or higher is required to enforce the separation between DBMS end users. Both discretionary and mandatory access controls are at least in part located in the filter.

This method should require minimal additional trusted code and minimal changes to an existing DBMS, and therefore be less costly to build. Because the DBMS is untrusted, there may be covert channel problems [LAND88] and more direct attacks. ARC is an example.

The TCB implementations place the assurance and security functionality in a relatively small kernel of code. The smallness of the kernel invites verification and other proofs of correctness. The TCB may be broken into subsets, with each subset enforcing a part of the policy.

One additional approach has been called the "distributed" approach. Here, one untrusted computer is used for each security level/compartment combination. A central trusted computer handles computer selection and query parsing. Two varieties exist. In the first, each machine has security combination. In the second, each machine has all the data up to that security combination. In the first, joins must be done in the central computer; in the second, joins can be done in the untrusted computers. Both could require much hardware. We know of no vendor examples. Research is being done.

NATIONAL COMPUTER SECURITY CENTER (NCSC) DISCRETIONARY SECURITY PROTOTYPE CONSIDERATIONS

Some of the factors considered in the "C2" prototypes developed at the NCSC are:

- discretionary access control
- object reuse
- identification and authentication
- audit
- security testing
- data integrity
- performance

These are typical factors that would be considered in a trusted implementation.

NCSC MANDATORY SECURITY PROTOTYPE CONSIDERATIONS

In addition to the "C2" prototype considerations, the following are being considered in the "B"-level prototypes developed at NCSC:

- labels
- label integrity
- exportation to
- multilevel hosts
- single level hosts
- exportation of labeled information
- mandatory access control

DISTRIBUTED DATABASE MANAGEMENT SYSTEMS (DDBMS)

Distributed database management systems form an important set of security problems and opportunities. This type of DBMS has multiple sites connected together into a communications network in which a user at any site can access data at any site. Characteristics of this DBMS may include the physical location of the data being transparent to the user, redundant data for performance and heterogeneous nodes. Vendors who have current implementations include Oracle and Ingres.

The DDBMS may be very efficient because data can be stored where the user uses it. Data can be better controlled by isolating it on particular nodes. The DDBMS, with multiple nodes and redundant data and communication paths answers the system availability or denial of service problem. System performance may be enhanced by local storage of frequent used data and by other distribution of data. Also, there are opportunities for the parallel execution of queries.

Problems also are many. How do you maintain database consistency with redundant data during updates/deletes and restores? What is the best method of identification and authentication? What is the best way to audit? Deadlocks must be controlled and priorities maintained. Other problems include the construction of a distributed MTCB, the part of the TCB that manages mandatory access control. Also, we must look at the distributed management of DAC, the Discretionary Access Control, and the problem of the consistency of DAC on replicated tables. How do you handle distributed transactions? Can serializability be maintained without creating inference channels? Can we use weak consistency? Are there new covert channels? A subsetted TCB could be very large and complex and therefore difficult to verify.

Encryption would be very useful between nodes and to store data. Long term keys are a problem. What algorithms should be used? How does this affect performance? How should the DDBMS be administered? What tools are needed? How do you resolve heterogeneous security policies? How do you assure the security of the system?

SUMMARY

Database security is a young interdisciplinary science, filled with promise and opportunities. The demand already exists. C-level operating systems and some B-level operating systems are here. An evaluation aid, the Trusted Database Interpretations, has been published. Trusted DBMS products are being produced. In the future there will be an increasing demand for database security. Many databases will be very large, distributed and with heterogeneous nodes. Databases will be smart, with multimedia data, where rules, and derived knowledge are stored and used. Parallel, array and fault tolerant processing will be the norm. Operating systems may have some database management system functionality. Security research and development is needed in all of these areas.

GLOSSARY

aggregation problem - The aggregation problem refers to the fact that the sensitivity level of a collection of data may exceed the sensitivity level of any individual datum in that collection. [NCSC89]

B - A TCSEC Division. The notion of a TCB that preserves the integrity of sensitivity labels and uses them to enforce a set of mandatory access control rules is a major requirement in this division. Systems in this division must carry the sensitivity labels with major data structures in the system. [DOD85]

C2 - A TCSEC class. Systems in this class enforce a more finely grained discretionary access control than C1 systems, making users individually accountable for their actions through login procedures, auditing of security-relevant events, and resource isolation. [DOD85]

Discretionary Access Control - A means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control). [DOD85]

domain - The set of objects that a subject has the ability to access. [NCSC91]

inference - derivation of new information from known information. The inference problem refers to the fact that the derived information may be classified at a level for which the user is not cleared. [NCSC89]

Mandatory Access Control - A means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity. [DOD85]

subset-domain - A set of system domains. For evaluation by parts, each candidate TCB subset must occupy a distinct subset domain such that modify-access to a domain within a TCB subset's subset-domain is permitted only to that TCB subset and (possibly) to more primitive subsets. [NCSC91]

trusted subject - A subject that is permitted to have simultaneous view and alter access to objects of more than one sensitivity level. [NCSC91]

REFERENCES

- [BELL73] Bell, D., and L. Lapadula, "Secure Computer Systems: Mathematical Foundations and Model", MITRE Report MTR 2547, v2 Nov 1973.
- [BIBA77] Biba, K., "Integrity Considerations for Secure Computer Systems", U.S. Air Force Electronic Systems Division, 1977.
- [BOOC83] Booch, Grady, Software engineering with Ada, Menlo Park: the Benjamin Cummings Publishing Company, 1983.
- [DATE86] Date, C. J., An Introduction to Database Systems, Reading, MA: Addison-Wesley, 1986.
- [DENN88] Denning, D. E., "Lessons Learned From Modeling a Secure Multilevel Relational Database System", Database Security: Status and Prospects, Amsterdam: Elsevier Science Publishers, 1988.
- [DOD85] DoD, Department of Defense Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, 1985. [JACO89] "Security In Computing", Computer, August, 1989, p. 150.
- [LAND88] Landwehr, C. E., "Database Security, Where Are We?", Database Status and Prospects, Amsterdam, Elsevier Science Publishers, 1988.
- [NCSC88a] National Computer Security Center, Glossary of Computer Security Terms, NCSC-TG-004-88, 1988.
- [NCS288b] National Computer Security Center, Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria, NCSC-TG-005, 1987.
- [NCSC89] National Computer Security Center, Draft Trusted DBMS Interpretation of the DoD Trusted Computer System Evaluation Criteria, 1989.
- [NCSC91] National Computer Security Center, Trusted Database Management Criteria, 1991.
- [WINK89] Winkler-Parenty, C. E., "Can You Trust Your DBMS", Database Programming & Design, July 1989, pp. 50-59.

Tutorial Series On Trusted Systems

Joel E. Sachs and Dr. William F. Wilson
Arca Systems, Inc.
2841 Junction Ave., Suite 201
San Jose, CA 95134
408-434-6633

Schedule

Tuesday, October 1	0800	Trust Fundamentals - Part I
	1030	BREAK
	1100	Trust Fundamentals - Part II
		Network Security Fundamentals
	1200	LUNCH
	1400	System Solutions and Security
		Distributed Security
	1530	BREAK
	1600	Certification & Accreditation
		Trusted Integration
	1730	ADJOURN

Description

These tutorials are based on Arca Systems' public and on-site Information Security Courses and experience learned in applying Arca's security consulting and engineering services to systems solutions. Arca provides support to its clients on both secure MLS system solutions and security products in all facets of trusted system design, analysis, development, implementation, testing, verification, integration, certification and accreditation. Arca has focused particularly on both trusted applications development and trusted integration of many products into secure system solutions. The tutorials relate experience from supporting systems integrators, applications developers, and end-users, as well as product vendors, who are addressing security in a variety of MLS system solutions for command and control, communications, and intelligence systems, development environments, and embedded systems.

The tutorials will be presented in lecture format with questions and answer periods. While there is a logical flow between the tutorials, each tutorial will be presented as a separate unit so that conference attendees can attend any or all of them. The morning tutorials concentrate on information security basics and the afternoon ones focus on addressing security in system solutions. The tutorials are intended to introduce many and varied security topics as opposed to exploring them in-depth. Brief descriptions of each tutorial identified above follows:

Trusted Fundamentals - Part I focuses on security and (TCSEC) trust concepts. Topics include security policies, mandatory and discretionary access controls, identification and authentication; security mechanisms, reference monitors, trusted computing bases, trusted path, least privilege; and assurance, formal and informal verification, covert channel analysis, security design analysis, security and penetration testing.

Trusted Fundamentals - Part II focuses on the TCSEC Evaluation Classes. The tutorial presents an overview of the TCSEC, its evaluation classes, and the NCSC evaluation process.

Network Security Fundamentals focuses on basic points in network security and gives an overview of the TNI. Topics include network security concerns and services, the structure of the TNI and its Evaluation Classes for both network TCBs and network components, and an overview of the TNI evaluation process.

System Solutions and Security focuses on system-wide security requirements in the context of system solutions. Topics include system solution characteristics, models, and development methodologies; and system-wide security problems, concerns, and threats and vulnerabilities.

Distributed Security focuses on the role of network security in today's distributed system solutions. Topics include system composition and interconnection, single system views versus interconnected automated information systems [AISs], cascading, encryption, and trusted network interfaces.

Certification & Accreditation focuses on the development of the certification evidence and inputs and decision process for accreditation. Topics include an overview of the certification and accreditation process, critical considerations, modes of operation, risk analysis, overall assurance requirements, and collecting system-wide evidence and assurance.

Trusted Integration focuses on integration issues that arise when developing and integrating secure and MLS system solutions. Topics include system-wide views of security policy, mechanism, and assurance; system, subsystem and component level interpretations for the roles of security policies, security policy models, and security top level specifications; security impact on the development methodology; and overview of security trade-offs.

ACCREDITATION STRATEGY FOR THE AIR FORCE SATELLITE CONTROL NETWORK (AFSCN)

By Lt Col William R. Price
Air Force Space Command/LKXS, Peterson AFB, CO 80914
Michael E. O'Neill, Ph.D. and Frank O. H. White
CTA Incorporated
7150 Campus Drive, Colorado Springs CO 80920

ABSTRACT

This paper examines the accreditation approach for a large, complex computer network, namely the Air Force Satellite Control Network. The network represents many existing computer networks, and as such, the approach for accreditation has broad application to the computer security community. The paper provides a brief background and history of the AFSCN. The accreditation approach is then described, followed by specific implementation stages for accreditation. The last section addresses "lessons learned" in the development of an accreditation strategy for the complex network.

Section 1-INTRODUCTION

This paper describes an ongoing effort to accredit a large, military communications-computer network. Although the paper describes a particular network, the Air Force Satellite Control Network, the authors believe it is representative of many existing networks and the approach taken has broad application to the computer security community. Functionally, the network supports the tracking, telemetry and commanding of military satellites. Telemetry from satellites provides status and health functions of on-orbit platforms (e.g., navigation, orientation, status of power system). The network typically does not process data collected or transmitted by satellite mission sensors (e.g., weather data). It provides both voice and data connectivity among satellite control sites throughout the world. This "real world," operational network has evolved over many years without the benefit of modern computer or network security theory and practice. Accreditation of the network is challenging from both a technical and management perspective.

The large and complex AFSCN has evolved over the last three decades. It employs a variety a variety of technologies. These technologies range from second generation computers and patch panel based communication systems to modern computers, workstations and computer controlled communication switching systems using fiber optics. Although security was not ignored in the design and evolution of the network, most AFSCN security protections predate the Trusted Computer Security Evaluation Criteria (TCSEC) and its Trusted Network Interpretation (TNI). A significant part of the accreditation effort is to devise or "reverse engineer" the overall network security concept and and document it.

Several organizations are involved in the management, operation and use of the AFSCN. Air Force Space Command (AFSPACECOM), an Air Force major command, is the network manager and, consequently, the Network Designated Approving Authority (DAA). Other organizations involved are Air Force major commands, DoD activities and civilian agencies such as the National Aeronautics and Space Administration (NASA). No one organization has complete control over the network, and the accreditation of the network must involve mutual benefit and agreement rather than the dictates of a single organization.

The remainder of this paper describes the ongoing efforts to accredit the network. Section 2 describes the AFSCN in more detail, providing information on its basic functions as well as the complexity encountered in addressing security architecture and security management relationships

in the AFSCN community. Section 3 discusses the approach to accreditation that is being pursued and the considerations that determined the approach. Section 4 describes the accomplishments to date and remaining activities planned. Section 5 describes our lessons learned which may of value to security managers involved in approving other communications-computer networks.

Section 2-BACKGROUND

AFSCN Components and Functions

The AFSCN provides spacecraft owner/operators (Air Force, NASA and others) the capability to track their satellites, send them commands and downlink health and status telemetry. These tracking, telemetry and commanding (TT&C) functions are depicted in Figure 1, AFSCN Concept of Operations. Several key AFSCN facilities, also referred to as AFSCN components in this paper, are shown in Figure 1. These components are briefly described below:

- Mission Control Centers (MCCs) are owner/operator facilities that remotely monitor and control spacecraft from launch to the end of their on-orbit life. MCCs maintain tracking information on their satellites and contact them as required to send commands and download telemetry related to spacecraft health and status. MCCs are operated by a variety of military and civilian agencies (AFSPACCOM, Air Force Systems Command (AFSC), NASA and others). Some MCCs support specialized Research and Development (R&D) spacecraft while others manage mature operational satellite systems. Most Air Force MCCs are located at the Consolidated Space Test Center, Onizuka Air Force Base (AFB), California or the Consolidated Space Operations Center, Falcon AFB, Colorado. Some MCCs supporting joint NASA/DoD operations are located at Johnson Space Center near Houston, Texas.

- There are nine Remote Tracking Stations (RTSs) located worldwide that provide spacecraft interface to the AFSCN environment. Most RTSs have two or more independent antennas and associated ground equipment sets for acquiring, tracking and communicating with spacecraft. A wide variety of radio frequency links and data link protocols can be supported by the ground equipment. RTS equipment, especially that installed by the Automated Remote Tracking Station (ARTS) program, can be remotely controlled by an MCC interfaced through the AFSCN communications network.

- AFSCN has redundant network control nodes at Onizuka and Falcon AFBs. Each node consists of a Resource Control Complex (RCC) and a Communications Control Complex (CCC). The RCC schedules network resources and directs configuration of network facilities to support the unique requirements of each spacecraft contact support mission. Under direction of its associated RCC, the CCC establishes connectivity called for in the contact mission support schedule. The CCC establishes circuits for commands, status and control, timing, telemetry and secure voice. Circuits to perform these functions are set up on both primary wideband and narrowband communications links to the RTSs. Bandwidth and data formats vary greatly from mission to mission due to the characteristics of the supported satellite.

- There are many other facilities in the AFSCN environment that support those described above. Software development facilities, test laboratories, satellite and RTS simulators, test driver systems and command centers are just examples. A variety of development, operations and logistics organizations operate these and a host of contractor support systems.

The AFSCN provides the communications services for satellite operators in MCCs to contact and control their spacecraft. The following scenario describes a typical satellite contact support mission:

Air Force Satellite Control Network (AFSCN) Concept Of Operations

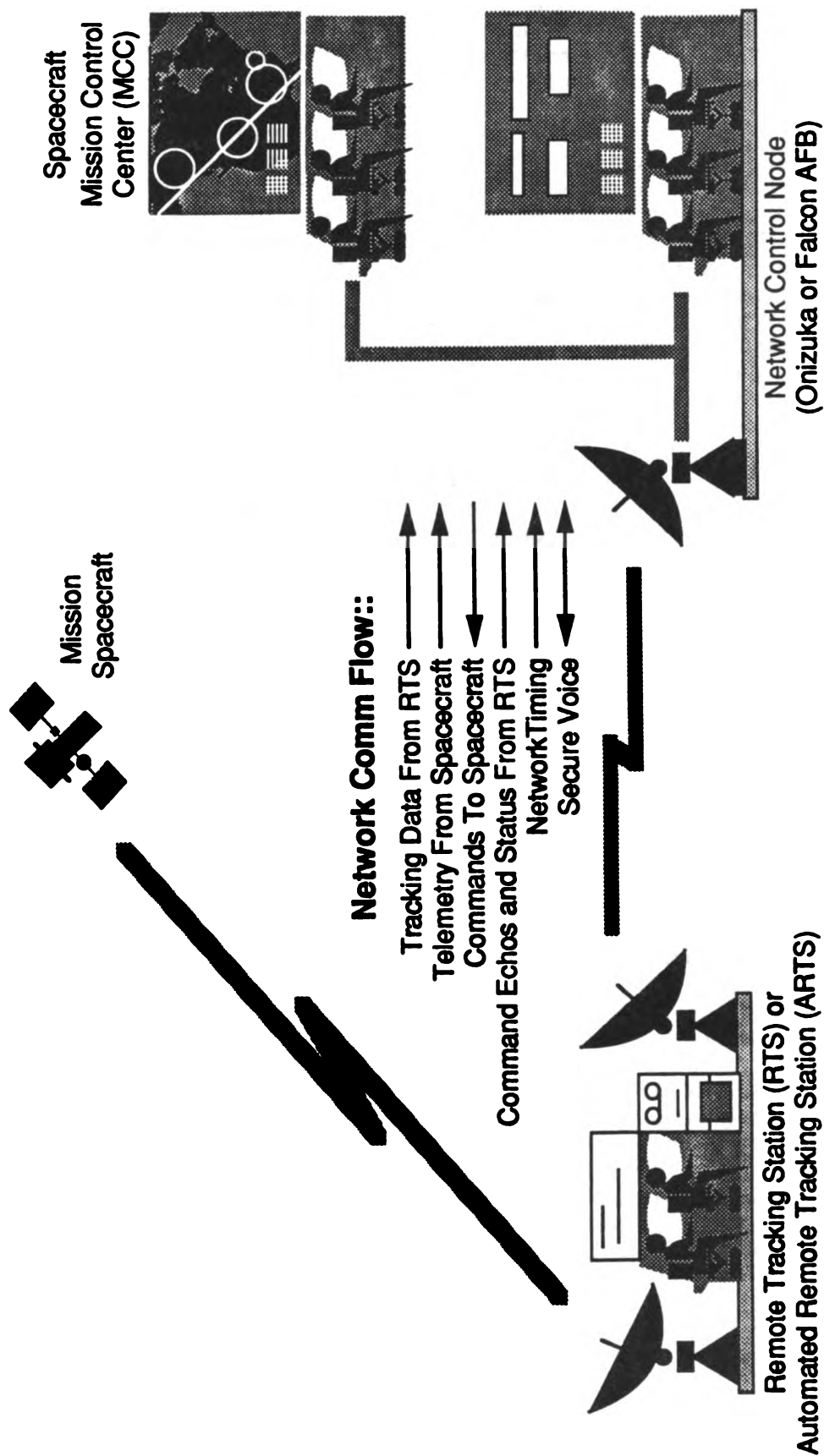


Figure 1 AFSCN Concept Of Operations

- Usually weeks in advance, an MCC coordinates with the AFSCN's primary RCC at Onizuka to schedule a contact support mission for its spacecraft. The contact support mission calls for the MCC to be connected to an RTS that has the satellite in its line-of-sight long enough for the required TT&C operations.

- The RCC schedules the network resources (the RTS, the mission-unique circuit mix on primary and alternate communications paths, and CCC interface to the MCC). This configuration is called a contact support mission string. The string may be needed for a few minutes or several hours depending on the spacecraft, its orbit and the TT&C functions to be performed. Hundreds of contact support missions are requested each week by MCCs representing many different spacecraft programs. There are sufficient network resources to support several simultaneous contact support missions (i.e., multiple mission strings operating in overlapping time frames), but often not all requests can be accommodated at the same time. The RCC continually deconflicts these competing requirements, often negotiating alternate times and network configurations with MCCs. The RCC manually generates a seven-day projection and a final 24-hour schedule to task network resources.

- Just prior to mission time, the RCC coordinates with the RTS and CCC via secure voice to configure the mission string. Establishing the mission string involves a combination of automated processes and manual patching by operators in the RCC, CCC, MCC and RTS. After verifying circuit connection on both primary and alternate communications systems, as well as proper functioning of RTS equipment, the RCC transfers computer control of the mission string to the supported MCC.

- After the RTS acquires the spacecraft, the MCC's Contact Support Processor receives and records tracking data from the RTS and sends commands to both the spacecraft and RTS equipment to start telemetry transfer. Typically, commands to the spacecraft and downlinked telemetry are protected at the Secret level. This end-to-end communications security is provided by peer encryption devices on the spacecraft and on the front end of the MCC processor. Unclassified mission data (status and control messages, timing, etc.) exchanged between the MCC and RTS are protected at the Unclassified Sensitive level. This transmission security protection is provided by bulk encryption devices on network communications links.

- When the MCC completes its spacecraft contact, the RCC disconnects the MCC and resumes control of network resources. Equipment and circuits in the mission string are returned to the pool of network resources for allocation to other scheduled missions. After disconnect, the MCC processes the telemetry data and often transfers it to support facilities for further reduction and analysis.

Some History

Until recently, AFSCN facilities were developed, owned and operated primarily by AFSC, an Air Force major command responsible for research and development. This changed in 1987 when the newly formed AFSPACECOM began to assume operational responsibility for AFSCN assets not dedicated to research programs. Over time, AFSPACECOM became owner/operator of the RTSs around the world, AFSCN satellite and terrestrial transmission systems and RCCs and CCCs at Onizuka and Falcon. They also activated some new Air Force MCCs at Falcon. AFSC retained responsibility for R&D spacecraft and continues to support them from MCCs at the Onizuka. With the transfer of most AFSCN operational systems to AFSPACECOM, the Colorado Springs based command was designated the overall AFSCN Manager and assumed primary responsibility for security management.

The Network Security Environment

Two AFSPACECOM organizations were tasked to implement security management: (1) , the Headquarters DAA who is responsible for approving operation of all computer and communications systems operated by AFSPACECOM; (2) the AFSCN Security Manager in the 2 Space Wing (2 SWG) who is responsible for day-to-day management of the AFSCN system security program. In assessing the state of security management in AFSCN, the DAA and Security Manager found the following:

- On the whole, competent security programs and security engineering had been put in place by various program offices over the years, but these typically focused on the computer system or facility being fielded or modified at the time. There were numerous security accreditations on file for individual computer systems and even a few for logical groupings of computer and communications facilities. The various accreditations for individual systems were for different modes of operation and security classification levels. Several security classification guidance changes were under consideration, but lacking an overall security concept or policy, assessing the impact of these proposed changes on the network was virtually impossible. Like most complex networks in place before promulgation of the national network security policy, the existing system and facility accreditations were like pieces of a complex puzzle. No one had yet begun to assemble the puzzle.

- There were many ideas how the puzzle should be assembled, but none of these seemed practical from a security standpoint. Reviews of planning and program management documents, as well as extensive interviews with managers of key development, operations and support organizations, revealed multiple network definitions. Each of these definitions made sense when seen through the eyes of their advocates, but no one definition provided a useful basis for understanding security-relevant services, facilities, interfaces and bounds. Although differing in detail, most definitions seemed all encompassing, driving the security analyst to examine unfathomable detail: scores of computer facilities and systems, hundreds of interfaces and a labyrinth of connectivity. In short, there was not a well articulated security architecture and there was not enough time or money to perform a network security analysis using the complex definitions offered up by various constituents in the AFSCN community.

- The network environment was highly dynamic, with literally hundreds of hardware and software upgrades underway at any one time. These upgrades were advocated and managed by a large number of organizations and programs, often competing for resources. Configuration control across the network was extremely complex. Network security enjoyed a very low priority in all this.

- Security management roles of the various commands, agencies and organizations involved in the community needed to be more clearly defined. There were many competent and highly motivated security managers throughout the community, but they focused on the facilities and systems within their sphere of influence. They were aware of evolving national guidance on network security, but the concepts and mechanisms of network security had yet to be institutionalized.

- There were differences in interpretation of Air Force computer security policy among the organizations. There was a perception that AFSCN systems would have to be scrutinized by both the individual System DAA and the Network DAA. Some organizations resisted such an approach where systems that had received security approval (System DAA) would have to be reviewed and approved again by an outside organization (Network DAA). These organizations felt that the Network DAA had no authority or responsibility for their operations. Another concern was that the individual System DAAs would duplicate efforts and, moreover,

could reach different conclusions based on differing interpretations of Air Force computer security policy.

Section 3-THE APPROACH

This section outlines the approach and methodology that was used to develop an accreditation strategy for the network. The challenge was to develop means and methods that would build consensus and approval for the approach and subsequent development of the AFSCN Network Security Policy.

Simplification Vs. Complexity

To get a reasonable handle on the network from a security perspective, a workable network definition was needed. Stepping back from the complex network definitions available in the community, we developed a simple model for trying to understand the security relevant relationships among the various facilities and systems in the AFSCN environment. This model, depicted at Figure 2, allocates the various facilities and systems to one of three layers. The model consists of the "Core Network" surrounded by two additional layers, External Interfaces and Support Components.

The Core Network consists of all the tracking, transmission, switching, and resource control facilities required to connect a spacecraft to its respective MCC, whether that MCC is operated by AFSPACECOM, AFSC, NASA or some other activity. The Core Network is anchored by the control nodes at Onizuka and Falcon AFBs. The other components in the Core Network are RTSs and ARTSs.

The External Interface layer of the model contains all AFSCN facilities that connect to the Core Network for TT&C services. In the main, these are MCCs, but other AFSCN components do connect to the Core Network from time to time for the purpose of TT&C testing and training. Also included in the External Interface layer are two satellite control networks dedicated to specific programs: Defense Metrological Support Program and the Global Positioning System.

All other AFSCN components are allocated to the Support Component layer of the model. These components frequently connect to External Interfaces for data analysis, software maintenance and other functions, but they rarely, if ever, connect to the Core Network. Allocation of components to this layer, where they have no direct impact on Core Network operational or security services, greatly simplified the complexity in our accreditation task.

We decided to look at the AFSCN as analogous to a telephone company providing service to its customers. As the sole operator of the Core Network, AFSPACECOM provides spacecraft owner/operators TT&C services much in the same way a telephone company provides telecommunications services to its customers. By thinking of the AFSCN and the Core Network in this manner, it allowed us to develop a strategy for accreditation that could be supported by the myriad of owners and users of AFSCN assets.

The telephone company view provided both technical and management benefits. Technically, the view allows a "divide and conquer" approach. The Core Network is the communications subsystem and provides for communication of unclassified data. It mediates MCC accesses to RTSs and knows nothing about individual users (people) in the MCCs. From the perspective of an MCC, the MCC communicates with a peripheral (i.e., the spacecraft) through the Core Network. The management benefit of this view is that it divides the network along organizational lines. The components of the Core Network are the responsibility of AFSPACECOM, the Network DAA. Individual MCCs and other External Interfaces are the responsibility of their operating commands, primarily AFSPACECOM and AFSC. The approach

Layered Approach to AFSCN Security

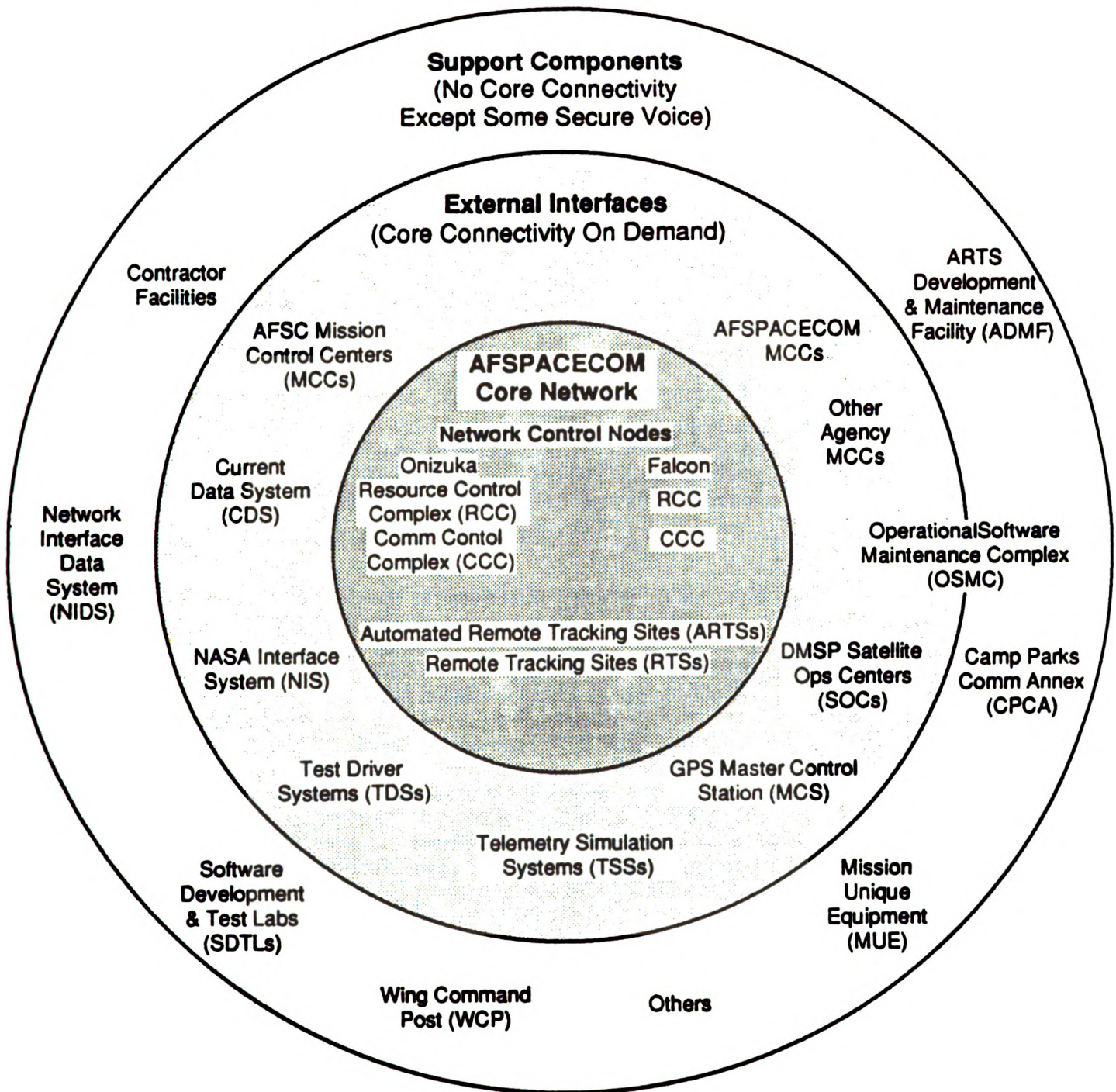


Figure 2
Layered Approach to
AFSCN Security

to network approval involves mutual actions of the DAAs with the mutual benefit of secure operations. The Network DAA certifies to the MCC DAA the security properties of the Core Network (assuming the External Interface satisfies certain conditions of connection). Correspondingly, the DAA for the MCC certifies compliance with network connection rules and approves operation of the MCC (assuming the Core Network maintains its security properties). This concept of mutual security assurance applies to all External Interfaces, not just MCCs.

Consensus Building

The development of a consensus among key AFSCN community players was absolutely necessary for a successful accreditation strategy. The newly formed AFSCN Security Working Group was instrumental in this effort.. This group represents the breadth of the AFSCN community, including security professionals, space operations personnel, developers and support managers. As a forum for presenting and refining the security model, it played a key role in getting support for the security model, the security concept of operations and their codification in the AFSCN Network Security Policy.

The Network Security Policy

The Network Security Policy first defines the AFSCN in terms of the layered model discussed earlier: a Core Network (tracking stations and communications that transport real time data and voice services in support of spacecraft contact missions), External Interfaces to the Core Network, and Support Components. This layered approach provides a method to understand who has what authority, who is responsible for what components and who is held accountable for what AFSCN security matters.

The policy defines three major security objectives for the Core Network. They are: 1. Network Confidentiality (non-disclosure), 2. Network Integrity and 3. Network Assurance of Service. An integrated program of protective security measures is employed across the Core Network for each security objective. Responsibility for implementing Core Network security objectives and protections are discussed as are specific connection rules for External Interfaces.

Dissecting the network into understandable components provided the framework to identify management responsibilities, authority and a means to provide accountability for the security policy objectives. Basically, the policy calls for command/agency DAAs to accredit individual AFSCN components in accordance with AFR 205-16 or equivalent agency security policy directives. They certify to the Network DAA that these formal accreditations are accomplished and that their components are compliant with all applicable requirements of the Network Security Policy.

Operational Perspective

In developing the Network Security Policy with the AFSCN community, the Network DAA and representatives from organizations operating network components, recognized the need for simple, streamlined security procedures that minimize impact on network operations. Based on the concept of mutual trust and security competence among DAAs, the Network Security Policy established the Letter Of Assurance (LOA) as the administrative mechanism for the Network DAA to maintain an ongoing assessment of the network security posture. The LOA is a one page document whereby DAAs for network components (Core Network and External Interfaces) certify to the Network DAA that their components are compliant with the security protection standards and connection rules in the Network Security Policy. The LOAs provide the basis for the Network DAA to accredit the Core Network and authorize connection of its External Interfaces.

Section 4-IMPLEMENTATION ACTIONS

Having developed a general consensus within the AFSCN security community regarding the Network Security Policy, detailed implementation of the policy is underway. Some of the major steps are discussed below:

- The first step is to create a network security management structure for implementing and enforcing the Network Security Policy. This structure includes all organizations that operate components in the Core Network and in the External Interface layer of network security model. With the Network Security Policy laying out the authorities, responsibilities and key relationships, this structure is headed by the Network Security Manager (NSM). The NSM is responsible for overall implementation and enforcement of the policy across the network and across organizational lines. The System Security Working Group, with representatives from all component organizations, is the NSM's advisory group for surfacing and resolving policy issues. Network Security Officers (NSOs) appointed in each Core and External Interface component execute the NSM's program on a day to day basis. These "hands-on" security managers implement and enforce detailed security procedures, investigate incidents and implement corrective actions. The security management structure also includes the Network DAA and DAAs from the various organizations that operate Core and External Interface components.
- The NSM must develop, coordinate and publish a Network Security Plan that provides detailed guidance for managing the Network Security Program. This document must include methods and standards governing risk analyses and security test and evaluations.
- Procedures must be developed and implemented to enforce the Network Security Policy in the requirements and configuration control processes. The DAAs and NSM must have visibility of new requirements and network changes so that they may assess security impacts and favorably influence implementation.
- All Core Network components and External Interfaces must be accredited by their respective DAAs. Most computer systems and facilities have current Interim or Final approvals to operate; however, some communications and tracking facilities in the Core Network have never been accredited. Additionally, some existing accreditations are nearing three-years of age and must be reaccomplished.
- As DAAs accredit components, they will certify these approvals to operate to the Network DAA through the Network Security Manager. As discussed earlier, the Letter Of Assurance will be the vehicle for this certification.
- When all the Core Network components and External Interfaces are accredited, the Network DAA will be able to accredit the network.

Section 5-LESSONS LEARNED

The AFSCN had been in place for many years before promulgation of national network policy. As such, it represented an evolving collection of complex components. It was a significant challenge to take this amalgam of components and develop a strategy for its accreditation. The lessons learned in this process are as follows:

- Develop a forum of security professionals for consensus building and negotiation of critical security considerations in the network. Look for people who have a vested interest in development of the forum and ultimate accreditation of the network.

- Define a simple, understandable architectural security model. It was necessary to develop a single but realistic definition of the AFSCN from a security standpoint. The concept of the Core Network was developed in order to bound the network. This was done through consensus building and negotiation with the security professionals from throughout the AFSCN community.

- Develop a spirit of mutual trust among the developers and operators that represent various organizational interests. This involves divorcing the Network DAA from the detailed risk analysis activities and holding the various organizations accountable for their portion of network risk assessment and accreditation. The Network DAA should, however, assure network integrity to its users and operators by initiating a Network Security Policy that precisely defines security protection mechanisms and connection rules. The basis for assurance from the Network DAA and DAAs accrediting network components and interfaces are Letters of Assurance that certify compliance with the Network Security Policy.

- The Network DAA should have a realistic and flexible attitude toward the network. It would be unrealistic to think that the Network DAA could shut down the AFSCN.

- Focus on a security management structure for implementing and enforcing the Network Security Policy. When defining network security responsibilities and authority, look for an existing organizational structure that can fulfill these duties whenever possible. For example, Network Security Officer responsibilities can be assigned to personnel who currently perform Computer System Security Officer functions at the various network facilities.

- Get senior management involvement from all organizations at critical stages of the accreditation process. Continually brief senior management on progress and strategy. This will develop the necessary support when critical decisions are required that cut across organizational boundaries.

AN ANALYSIS OF APPLICATION SPECIFIC SECURITY POLICIES

Daniel F. Sterne Martha A. Branstad Brian S. Hubbard[†] Barbara A. Mayer[‡]
Dawn M. Wolcott[§]

Trusted Information Systems, Inc., Glenwood, Maryland

Abstract

The TCSEC [20] is concerned primarily with the DoD confidentiality. As a result, for many applications, systems that satisfy the TCSEC may nevertheless provide an insufficient base of security policy enforcement. This paper summarises a study whose objective is the identification of a broader range of security policies that merit automated support, particularly in tactical computer systems.

The study analysed operational requirements of a collection of tactical and non-tactical application scenarios. Synopses of several example scenarios are presented, and the findings of the study are discussed. The study suggests that while many policies are application specific, there exists a core of policy elements common to a broad range of such policies, and that this core merits automated support in future trusted systems.

Keywords: *security policy, access control, roles, integrity, denial of service.*

1 Introduction

The TCSEC [20] is oriented primarily toward confidentiality policies, and in particular, the protection of classified information from disclosure to insufficiently cleared individuals. As a result, systems that satisfy the TCSEC may fail to address other important security requirements, particularly those associated with tactical military applications. If systems capable of satisfying broader ranges of security requirements are to be constructed, the security policies that underlie these requirements must be more clearly articulated. To the extent that these policies may be application specific, it is important that policy elements common among them be identified, that these elements become candidates for automated support in future trusted systems.

This paper summarises the initial phase of a project whose ultimate objective is the construction of a prototype system that can be configured to support a range of application specific security policies, and in particular, policies associated with military systems [24]. The objective of this initial phase is the identification of security policies and common policy elements that merit automated support in tactical computer systems.

The U.S. Department of Defense (DoD) is under increasing pressure to use commercial-off-the-shelf (COTS) hardware and software, and to avoid procuring customised system components. Because commercial systems,

This work was funded by DARPA through RADC contract F30602-89-C-0125

[†]Currently with SPARTA, Inc., Columbia, MD

[‡]Currently with Atlantic Research Corp., Hanover, MD

[§]Currently with the MITRE Corp., McLean, VA

like tactical systems, may also need to support policies not addressed directly by the TCSEC, another objective of the initial phase is to examine the commonality of commercial and other non-tactical policies with those of the tactical realm. If commonality exists, systems designed to support commercial and non-tactical security policies may be able to support tactical policies as well.

1.1 Approach

This study could have proceeded based purely on conjecture and an abstract conceptual view of tactical operations. That approach seemed overly speculative, and unlikely to produce meaningful results. To keep the study more closely tied to reality, a somewhat different approach was taken. A sampling of applications "scenarios" was selected, and for each scenario, information was gathered and then analysed. The tactical scenarios chosen deal with the Navy's Aegis combat system, the command, control, and communications interactions associated with the Air Force nuclear weapon release process, and Army field operations and support services. The non-tactical scenarios concern government procurement document preparation and release, commercial accounting and data processing, air traffic control, and medical information system usage.

If the security policies associated with these scenarios were clearly understood and had been clearly articulated by their associated organisations, it would have been sufficient for this study to have collected and catalogued existing policy statements; little if any policy analysis would have been required. However, the distinction between a "security policy" and other kinds of regulations, operational procedures, and critical system requirements has not been clearly established. Consequently, for many organisations, it is not clear that distinct security policy statements actually exist, apart from those concerned with confidentiality.

In the absence of such policy statements, the study proceeded by examining operational and system requirements for each scenario. For scenarios dealing with existing organisations and systems, to the extent practical, these requirements were collected from technical articles and discussions with knowledgeable individuals. For scenarios dealing with future systems whose requirements and impacts on organisations have not yet been completely established (e.g., CALS [9]), incomplete information about operational requirements was augmented by educated guesses.

Each scenario was then analysed to identify underlying security policies and policy characteristics. While the analysis produced results the authors believe are useful, these results are of necessity partly subjective; policy statements cannot be mathematically derived from operational requirements, but can only be loosely inferred. Moreover, the analysis was not exhaustive; it did not attempt to consider all requirements or identify all possibly relevant security concerns. The analysis of each scenario concentrated on a small set of security concerns that seemed most fundamental with respect to the overall mission and threats. Consequently, the results reported here are not intended as a definitive analysis. Rather, they represent an illustrative sampling of security policies in which an emphasis has been placed on security concerns other than confidentiality.

1.2 Organization

This paper is organised as follows. First, a few fundamental definitions are given. Next, excerpts from the security analysis of three example scenarios are presented to illustrate the range of security policy elements identified in tactical and non-tactical scenarios. The examples are followed by a summary and discussion of the study's findings, and a short section on future work.

2 What Is A “Security Policy” ?

The scenarios examined in the study encompass a wide spectrum of critical operational and system requirements. Given the objective of identifying security policies, in particular, policies beyond confidentiality, it became apparent early on that a means for distinguishing security policies from other kinds of critical requirements was needed. Since recent trends in terminological usage have tended to blur the distinction between criticality and security, a set of fundamental definitions was developed for use in the study. These definitions, described below, also distinguish between policies that govern human activity and those that govern automated processes on a computing system. Furthermore, these definitions describe a somewhat different view of security than that implied by the maxim “confidentiality, integrity, and assured service” [27, 7, 22]. A more complete discussion of these definitions can be found in [25].

2.1 Definitions

Security Policy Objective - A statement of intent to protect an identified resource from unauthorised use. The statement must identify the kinds of uses that are regulated. A security policy objective is meaningful to an organisation only if the organisation owns or controls the resource to be protected.

This definition establishes the primary notion of security upon which the other definitions are based: protection of tangible assets from unauthorised use. Examples of security policy objectives include protecting classified information from unauthorised disclosure or modification, preventing unauthorised distribution of financial assets, preventing unauthorised use of long-distance telephone circuits, preventing unauthorised dispensing of prescription drugs. The notion of a security policy used here is broader than that of the TCSEC, which is concerned with protecting a single kind of resource: information.

Organisational Security Policy (OSP) - The set of laws, rules, and practices that regulate how an organisation manages, protects, and distributes resources to achieve specified *security policy objectives*. These laws, rules and practices must identify criteria for according individuals authority, and may specify conditions under which individuals are permitted to exercise or delegate their authority. To be meaningful, these laws, rules, and practices must provide individuals reasonable ability to determine whether their actions violate or comply with the policy.

An OSP describes how a security policy objective is to be manifested in the routine activities of the organisation. The OSP definition is patterned after the security policy definition given in the TCSEC glossary,¹ but addresses protection of resources other than information. In addition, it explicitly cites the authorisation of individuals as fundamental to the notion of a security policy, and allows authorisation to be based on attributes other than clearance and need to know. For example, authorisation may be based on job title, employer, training, licensing, enrollment, or membership.

Automated Security Policy (ASP) - The set of restrictions and properties that specify how a computing system prevents information and computing resources from being used to violate an *organizational security policy*.

An ASP specifies what a trusted system is trusted to do. The ASP for a TCSEC-oriented trusted system (class B or higher) typically includes the Bell-LaPadula properties [3], labeling requirements for human readable output, I&A-oriented restrictions (e.g., minimum password length), audit capture requirements, and so forth.

¹“The set of laws, rules, and practices that regulate how an organisation manages, protects, and distributes sensitive information.”

2.2 The Meaning of "Security Policy" In this Paper

This study is concerned with *organizational* security policies, that is, laws, rules, and practices that govern the activities of people. The analysis of organisational security policies is a prerequisite for analysis of automated security policies. Throughout this paper, the terms "security policy", and "policy" are used for brevity, but are intended to refer to organisational rather than automated security policies.

3 Roles

In the sections that follow, the term "role" [2, 15, 26] occurs frequently. We will use the term role to mean a named group of rights; these rights are permissions to access, operate on, or otherwise use resources in particular ways. A financial officer role might include rights to disburse financial assets (by signing checks) and to approve release of corporate financial information. The role of payroll clerk may include the right to examine employee salary data. The role of pharmacist includes the right to dispense drugs but not prescribe them; that right belongs to the physician role. A DoD security officer role might include rights to add new user accounts to a classified computing system and to control the system's audit data collection. Individuals belonging to an organisation are assigned to roles and are then able to exercise the rights associated with those roles. Consequently, roles are a means of naming and describing many-to-many relationships between individuals and rights.

Role exclusion rules may be associated with roles. These rules place constraints on the ability of individuals to be authorised for roles or to assume roles for which they are otherwise authorised. For some roles, there may be a limitation on how many individuals can be concurrently active in the role [15]. For example, in certain military organisations, only a single individual may be able to assume the role of watch officer at a time. Other individuals who are otherwise authorised to assume the watch officer role, cannot assume the role until it has been relinquished. Some combinations of roles may be considered "conflicting" because together they provide more authority than the organisation permits any one individual to hold; there may be a prohibition against any one individual being assigned (authorised for) more than one of these. For example, in a commercial corporation, an individual may be prohibited from acting as both a financial officer and a financial auditor. This kind of exclusion rule is equivalent to so-called "static" separation of duty, as defined in [19], and discussed elsewhere in the literature [6, 18].

4 The Aegis Combat System

The Aegis combat system is a sophisticated shipboard combat system used in U.S. Navy cruisers and destroyers [8]. The Aegis system includes a variety of sensors, including radar and sonar, and weapons, including surface-to-air missiles, surface-to-surface missiles, miscellaneous anti-submarine devices, guns, and small multi-purpose helicopters. These assets are monitored and controlled from the Combat Information Center (CIC), a room containing numerous operator consoles and large screens used to display situation maps and tactical summaries. In order to support mission requirements for high fire power and rapid response to threats, the Aegis system provides extensive automated response capabilities that can be programmed by the ship's crew.

Three organisational security policies were identified from descriptions of the Aegis¹ system. These concern the prevention of 1) unauthorised disclosure of classified information, 2) unauthorised modification of information, and 3) unauthorised release of weapons. Each is discussed in a subsection below.

4.1 Information Disclosure Policy

The information disclosure policy is based on well-established DoD regulations and is directed at protecting classified information from individuals lacking sufficient clearances. Crew members may be uncleared, cleared for shipboard tactical information, or cleared for both intelligence and shipboard tactical information. In addition, uncleared visitors may occasionally be aboard. Most members of the crew are cleared for tactical information, which includes targeting data, locations of friendly forces, mission plans and situation tactics, and information about capabilities and limitations of sensors, weapons, and other equipment. A small fraction of the crew may, in addition, be cleared for access to intelligence information.

Both kinds of information are protected by physical and procedural security measures. Armed guards prevent unauthorised individuals from boarding the ship when it is in port. While at sea, access to the CIC and to the intelligence room is controlled by locks on entry doors. When information is transmitted among the ships in the fleet, communications security measures are employed to prevent eavesdropping.

4.2 Information Modification Policy

The information modification policy is concerned with preventing unauthorised individuals from supplying, changing, or deleting intelligence and tactical information. To a lesser extent, it may also be concerned with preventing authorised individuals from modifying such information in an clearly erroneous manner. This policy is not explicitly articulated, but has been inferred by the authors from descriptions of operational procedures.

Intelligence information and tactical information must only be accepted from designated sources. Designated sources may be organisations, or individuals assigned to particular job functions. Designated sources vary according to the type of information. Accepting information from sensors, computers, or other equipment is authorised if the equipment is operated under the auspices of a designated source organisation or individual. The authority to act as a designated source for a particular kind of information constitutes a role.

Cleared shipboard personnel are authorised to extract, derive, delete, enter, or otherwise modify tactical information. Similarly, personnel with intelligence clearances are authorised to modify intelligence information. When authorised individuals make such modifications, they are expected to employ any applicable designated processing methods or algorithms² so that modifications are minimally subjected to simple error checks. In some cases, however, the organisation must rely primarily on the considered tactical judgment of senior officers to ensure that information modifications are valid, i.e., consistent with reality and the intentions of superiors. Moreover, all authorised individuals are trusted not to introduce intentional inaccuracies into protected information, except as required for sanitisation purposes.

Ships in the fleet may share tactical data (e.g., concerning potential targets) in digital form via radio-based ship-to-ship communications. As a result, console operators on one ship have a limited measure of authority to influence (modify) another ship's tactical information base. The extent of this authority is constrained by protocols and algorithms that are used to resolve conflicts among multiple information sources. Depending on the circumstances and kind of information involved, conflicting information received from other ships may replace or be added to the information generated by a ship's own sensors and crew. Alternatively, conflicting information may be discarded, or mathematically combined.³ Thus for each ship, an authorisation distinction is made between console operators on that ship, and those on other ships in the fleet; these constitute different roles. Except for cleared members of fleet crews and designated information sources, no other individuals have authority to modify shipboard information.

²These may be embedded in the ship's computer programs.

³Planned for future system upgrades.

4.3 Weapon Release Policy

The weapon release policy is directed at preventing weapons, especially missiles, from being released without appropriate authorisation. Only the ship's Commanding Officer (CO) has the authority to order the release of weapons, although he may delegate this authority to the Tactical Action Officer (TAO). Although several individuals on a ship may be authorised to assume the TAO role, only one can assume it at a time; this is an example of a role exclusion rule. The CO or TAO can order (authorise) one or more of the combat system console operators to release a weapon. A weapon release order can be given directly to the console operator, or may propagate downward to the operator through the chain of command. Similarly, only the CO and TAO have the authority to order the creation, modification, enabling or disabling of programmable automated weapon release rules called "doctrine statements". The Combat System Coordinator (CSC) is the only role given authority to enter or alter these statements and typically is authorised to do so only when specifically directed. Furthermore, typical operating procedures dictate that doctrine statements be written on paper and signed by the CO prior to being entered into the system by the CSC.

4.4 Policy Summary

The security policy objectives for this scenario include preventing unauthorised disclosure and modification of information, and preventing unauthorised release of weapons. Authorisation to use these resources is contingent on clearances, roles and role exclusion rules, delegation of authority, and non-repudiable (signed) orders.

5 Nuclear Command, Control, and Communications

The principal requirements of the nuclear command, control, and communications (NC3) system are 1) rapid response to authorised orders directing the release of nuclear weapons, 2) prevention of unauthorised weapon release, and 3) prevention of unauthorised disclosure of classified information associated with deployment plans and the release process.

5.1 Weapon Release Policy

A nuclear weapon release requires collaborative actions on the part of multiple individuals, each of whom has been assigned one of three specific roles. The civilian authority authorises the use of nuclear weapons. The military authority generates specific targeting orders that must comply with previously established plans. These orders are then carried out by launch control officers. This division of authority amongst the civilian authority, the military authority, and the launch control officers constitutes separation of duty. No unilateral action by any individual in any of these roles, by itself, should allow a nuclear weapon release to be successfully initiated. Forced collaboration among these roles during the release process is accomplished via cryptographic procedures. In addition, a split knowledge policy among the individuals assigned the role of military authority requires that at least two of these individuals collaborate (by combining secrets) before they are able to execute a release successfully. Stringent source authentication requirements play a central role in the protocols used by these roles during their interactions; in some cases, the protocol prohibits a role from proceeding with its duties without having successfully authenticated the source of a received directive.

Following authorisation, two-person or N-person controls are used extensively; each launch control officer is assigned to a team, and is prohibited from carrying out launch control related activities unless authorised and accompanied by his team member(s). These controls prohibit a single launch control officer from accessing launch control information, facilities, authenticators, and cryptographic materials.

5.2 Denial of Service Policy

The denial of service policy for the NC3 system is directed at preventing unauthorized individuals from inhibiting an authorized release of nuclear weapons. (Having such a policy does not preclude the possibility that some individuals may be authorized to prevent weapon releases, for example, after cessation of hostilities.) This policy is manifested in a host of personnel, physical, and communications security measures that are beyond the scope of this discussion. We note, however, that the N-person controls described above for essential components of the launch control process also make less likely the unauthorized modification, replacement, theft, or destruction of these components. Because a loss, or loss of effectiveness, of any such component may inhibit weapon release, these N-person control measures also support the denial of service policy.

5.3 Information Disclosure Policy

As for the Aegis information disclosure policy, this policy is based on well-established DoD regulations and is directed at protecting classified information from individuals lacking sufficient clearances. Among the kinds of information of concern for the NC3 system are plans and contingencies for weapon deployment, and current status. The latter may include current capabilities, information about deployments in progress, and heightened states of operational preparedness. This information is protected by a variety of physical, procedural, and communications security measures.

5.4 Information Modification Policy

This policy is a subordinate policy whose objective is primarily to support the NC3 weapon release policy and denial of service policy. For example, if release orders are subject to unauthorized modification prior to being carried out, then it may be possible to subvert the intent of the release authorities, causing an unauthorized release. Similarly, to the extent that information is used as an enabling element in the launch control process, an unauthorized information modification could inhibit release, resulting in an unauthorized denial of service. Weapons orders, plans, and other types of release-governing information are protected against unauthorized modification by a variety of communications, physical, and procedural security measures including the N-person control procedures described above.

5.5 Policy Summary

The security policy objectives for the NC3 scenario include unauthorized disclosure and modification of information, unauthorized release of weapons, and unauthorized denial of service. Authorization to access or use these resources is contingent on clearances, roles, separation of duty, split knowledge, N-person control, and source authenticated inputs.

6 Government Procurement Document Preparation

This scenario is concerned with the security policies associated with the government procurement process, primarily as they affect the government participants. The Computer-aided Acquisition and Logistics Support (CALS) program [21], is an ambitious attempt to automate much of this process in the future, as well as other activities supporting the design, manufacture, and logistical support of systems used by DoD. Unlike the previous two scenarios, which are based on existing operational and system requirements, this scenario is based on hypothetical future requirements extrapolated from fragmentary published descriptions of the

CALS program [21, 9, 10]. It concentrates on the preparation, approval, and release of Requests for Proposals (RFPs), allocation of government funds and manpower, and evaluation of competing bids.

6.1 Information Disclosure Policy

Much of the information associated with the procurement process may be sensitive with respect to disclosure. The procured items may have specifications that are classified. To ensure fair competition among bidders, information about the contents of RFPs under development must not be "leaked" prematurely to prospective bidders. There may also be information which is considered proprietary to a particular vendor. Dissemination controls (e.g., NOFORN, NOCONTRACTOR) and export control restrictions may need to be enforced.

6.2 Information Modification and Release Policy

The RFP development process consists of a sequence of draft, approval, and release phases. Among other purposes, these phases serve to prevent unauthorised procurement documents (e.g., erroneous RFPs and contracts) from resulting in unauthorised expenditure commitments of government funds and manpower resources. At each stage in the RFP development process, approval must be obtained prior to proceeding to the next stage. Furthermore, authority to submit, modify, or approve procurement documents at various stages is reserved to individuals who have been assigned particular roles. To the extent that procurement documents are kept on-line, controls are required to ensure that only appropriate individuals are able to update or modify information at each stage.

RFP development is initiated by a technical team whose members are authorised to generate a statement of work (SOW). Before an RFP can be generated, the SOW must be approved by management, and procurement funds must be allocated. The SOW is then forwarded to the contracts department, whose personnel are authorised to generate an RFP. The RFP must be approved by the legal department and approved for release by a contracting officer. As part of the release process, an authenticating code may be attached to assist bidders in verifying the authenticity of the RFP prior to committing their own resources for bid development.

6.3 Other Constraints

The roles held by individuals may be subject to role exclusion constraints. For example, members of the technical review team for bid evaluation may be prohibited from participating on the cost review team; this can be viewed as a form of separation of duty. Furthermore, they may also be forbidden from finding out about contents of the cost portions of bids. It may also be the case that an individual who has had access to a contractor's bid containing proprietary information may be forbidden from accessing a competing contractor's proprietary information for a set period of time.

6.4 Policy Summary

The security policy objectives for this scenario include preventing unauthorised disclosure of information, unauthorised modification and release of information, and unauthorised expenditure commitments of funds and manpower. Resource usage authorisation is based on clearances and dissemination controls, roles, role exclusion constraints (including separation of duties), operation sequencing constraints and source authentication.

7 Observations

An analysis of the scenarios studied, including those outlined above, leads to a number of observations.

- Access control according to clearances or roles appears to be a fundamental aspect of each scenario. In particular, access control to protect information from both unauthorised disclosure and unauthorised modification was an element of every scenario. In addition, numerous other role-based access controls regulating use of resources other than information (e.g., weapons, financial assets) were found.
- Infrastructure support, particularly in the form of communications, identification and authentication, and auditing services, is likely to be applicable, independent of policy objectives. The extent of applicability depends on the geographic distribution of the organisation and the extent of its reliance on automation.
- Separation of duty constraints were found in several scenarios. This suggests that separation of duty is a well-established general principle that is widely employed when an organisation is reluctant to entrust unilateral control over a resource to any single individual. Furthermore, separation of duty requirements were sometimes accompanied by operation sequencing requirements. In some military environments, however, the principle of separation of duty may conflict with the need to ensure that, at all times, at least one individual (e.g., a commanding officer) has *sufficient authority* over resources to carry out an assigned mission successfully.
- Each scenario encompasses a unique combination of policy elements. No clear-cut patterns emerged to distinguish the policies for tactical scenarios, as a group, from those for non-tactical and commercial scenarios. However, because responsibilities must be rapidly and flexibly reassigned following combat casualties, tactical policies may tend to rely more heavily on fluid personnel authorisation methods including delegation of authority.
- Source authentication or non-repudiation requirements stipulating that personal or electronic signatures accompany data or permission to act, (e.g., military orders) appear to be widespread.
- “N-person rules” requiring teams of people to act simultaneously (or nearly so), and split knowledge requirements, were not common. This may be because their implementation is too costly or cumbersome to be used on a routine basis unless the resources being protected are extremely critical, as in the case of nuclear weapons.
- Numerous *requirements* related to denial of service, including requirements for reliability, survivability, and performance were encountered. However, few denial of service *policies* (as defined above) were identified; such policies govern the authority of particular individuals to use or operate on resources in ways that may deny use of those resources to otherwise authorised individuals. Several explanations for this result can be posited. First, denial of service remains an ill-understood problem, and denial of service policies remain difficult to identify definitively. Second, the security policy definitions used in this study deliberately exclude from consideration a variety of critical requirements that are commonly treated as security policy manifestations [27, 7, 22]. Third, primary threats to assured service in tactical systems include electronic warfare and the destruction of combat assets by the enemy. Threats of this nature are more naturally addressed by military tactics and improved equipment capabilities than by computer security technology, and were consequently deemphasised in the study.

These observations suggest that there exists a core set of security policies and policy elements that merit support in computing systems intended for a broad range of tactical, non-tactical, and commercial applications. This policy core includes protection of information from unauthorised disclosure and modification, role-based access control, role exclusion rules, (e.g., static separation of duty), delegation of authority, and

operation sequencing.⁴ The core also includes identification and authentication, auditing, and reliance on secure communications, especially to support source authentication and non-repudiation requirements.

These observations also support the contention that the TCSEC requirements are incomplete in comparison with secure tactical computing needs. A number of other security policies beyond confidentiality are integral to the contexts in which tactical systems are used, and it appears that tactical systems should be capable of providing some degree of automated support for these policies. Determining the extent to which policy support can be automated usefully, especially when possible system failure modes are taken into account, will require a significant level of continuing research addressing both human factors and systems engineering issues.

The authors feel it unlikely that automated mechanisms designed primarily for TCSEC requirements are well-suited to support these other policies. (Similar sentiments have been published elsewhere [11, 16]. See [4, 5] for a different perspective.) On the other hand, it appears that the information disclosure policies toward which TCSEC requirements are targeted remain crucial to tactical systems. Consequently, computing systems designed for a broad range of tactical applications should be minimally capable of satisfying TCSEC requirements in addition to supporting other organisational security policies.

8 Summary and Future Work

This paper has summarised the results of a study intended to identify security policies and common policy elements that may merit support in systems designed for tactical applications. While each analysed scenario appears to encompass a different combination of policy elements, the study suggests that these combinations may share a common policy core. This offers the hope that by supporting this common core, a single, configurable system may be able to support a wide variety of application specific security policies in the tactical, non-tactical, and commercial realms.

While a number of previous research papers have discussed table-driven, rule-driven, or otherwise configurable systems that may support multiple policies [14, 1, 17, 26, 4, 5], the feasibility and assurance potential of such systems remains an open research question; much more work is needed before a definitive answer can be put forth. Toward this end, functional requirements for a prototype system to support the policy core have been developed, and high-level design activities have been initiated. Follow-on plans include implementation of the prototype and an assessment of the applicability, effectiveness, and assurance of its enforcement mechanisms.

References

- [1] Abrams, M.D., Eggers, K.W., La Padula, L.J., Olson, I.M., "A Generalised Framework For Access Control: An Informal Description," Proceedings of the 13th National Computer Security Conference, Washington, D.C., October, 1990.
- [2] Baldwin, R.W., "Naming and Grouping Privileges to Simplify Security Management in Large Databases," Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy, Oakland, CA, 1990.
- [3] Bell, D.E., La Padula, L.J., "Secure Computer Systems: Unified Exposition and Multics Interpretation," Technical Report No. ESD-TR-75-306, Electronics Systems Division, AFSC, Hanscom AF Base, Bedford MA, 1976.
- [4] Bell, D.E., "Lattices, Policies, and Implementations," Proceedings of the 13th National Computer Security Conference, Washington, D.C., 1990.

⁴Although dynamic separation of duty [10, 23, 12], is not discussed above, we include it as an extension of operation sequencing.

- [5] Bell, D.E., "Putting Policy Isomorphisms to Work," Report 355-D, Trusted Information Systems, Glenwood, MD, November 1990.
- [6] Clark, D.D., Wilson, D.R., "A Comparison of Commercial and Military Computer Security Policies," Proceedings of the 1987 IEEE Symposium on Security and Privacy, Oakland, CA, 1987.
- [7] Gasser, M., *Building a Secure Computer System*, Van Nostrand Reinhold Co., New York, NY, 1988.
- [8] Gersh, J., Private Communications, Johns Hopkins University Applied Physics Laboratory, October 1989, January 1990, August 1990, February 1991, June 1991.
- [9] Gorham, Jr., W.C., "Data Protection Requirements in Computer-Aided Acquisition and Logistics Support," Proceedings of the Fifth Annual Computer Security Applications Conference, Tucson, AZ, December 1989.
- [10] Gove, R.A., Friedman, A.R., "A Structured Risk Analysis Approach to Resolve the Data Protection and Integrity Issues for Computer-Aided Acquisition Logistics Support (CALS)," Proceedings of the Fifth Annual Computer Security Applications Conference, Tucson, AZ, December 1989.
- [11] Graubart, R., "On the Need For a Third Form of Access Control," Proceedings of the 12th National Computer Security Conference, Baltimore, MD, October, 1989.
- [12] Karger, P., "Implementing Commercial Data Integrity with Secure Capabilities," Proceedings of the 1988 IEEE Symposium on Security and Privacy, Oakland, CA, 1988.
- [13] Karp, B.C., "The CALS Data Protection and Integrity Industry Working Group," Proceedings of the Fifth Annual Computer Security Applications Conference, Tucson, AZ, December 1989.
- [14] La Padula, L.J., "Formal Modeling in a Generalised Framework for Access Control," Proceedings of the Computer Security Foundation Workshop III, June 1990.
- [15] Mayer, F.L., "Security Controls for an Automated Command and Control Information System (ACCIS): Baseline Definition," Report 201, Trusted Information Systems, Glenwood, MD, May 1989.
- [16] McCollum, C.J., Messing, J.R., Notargiacomo, L., "Beyond the Pale of MAC and DAC - Defining New Forms of Access Control," Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May 1990.
- [17] Miller, D.V., Baldwin, R.W., "Access Control by Boolean Expression Evaluation," Proceedings of the Fifth Annual Computer Security Applications Conference, Tucson, AZ, December 1989.
- [18] Murray, W.H., "Data Integrity in a Business Data Processing System," Report of the Workshop on Integrity Policy in Computer Information Systems (WIPCIS), Waltham, MA, October 1987.
- [19] Nash, M.J., Poland, K.R., "Some Conundrums Concerning Separation of Duty," Proceedings of the 1990 IEEE Symposium on Security and Privacy, Oakland, CA, May 1990.
- [20] National Computer Security Center, *Department of Defense Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD, December 1985.
- [21] O'Neal, S.A., "CALS: Enabling Process Improvement," Signal, September 1989, pp. 41-44.
- [22] Pfleeger, C., *Security in Computing*, Prentice-Hall International, Inc., Englewood Cliffs, NJ, 1989.
- [23] Sandhu, R., "Transaction Control Expressions For Separation of Duties," Proceedings of the Fourth Annual Computer Security Applications Conference, 1988.
- [24] Sterne, D., Branstad, M., Hubbard, B., Mayer, B., Badger, L., Wolcott, D., "Security Policies for Tactical Environments: A Research Study," Report No. 353, Trusted Information Systems, Glenwood, MD, November 1990.

- [25] Sterne, D., "On The Bussword 'Security Policy'," Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May, 1991.**
- [26] Thomsen, D.J., "Role-based Application Design and Enforcement," Proceedings of the Fourth IFIP Workshop on Database Security, Halifax, England, September 1990.**
- [27] Unisys, Camarillo, CA., and Trusted Information Systems, Inc., Glenwood, MD., SDI Battle Management Security Issues: A Preliminary View, TM-8361/000/00, February 1987.**

ANOTHER FACTOR IN DETERMINING SECURITY REQUIREMENTS FOR TRUSTED COMPUTER APPLICATIONS

David Ferraiolo
NIST
Building 224/A241
Gaithersburg, MD 20899

Karen Ferraiolo
Grumman Data Systems
2411 Dulles Corner Park
Herndon, VA 22071

Computer systems take on security requirements that are unique to the operational characteristics and needs of their application. These requirements can be applied on an individual basis in reducing operational risk. Methods exist to determine security requirements per DoD 5200.28-STD [3] by calculation of a risk index [4], [6]. This risk index is used to determine an appropriate level of trust (criteria class) per DoD 5200.28-STD, which is then used to define a set of security requirements. However, the resulting security requirements imposed on some systems by DoD 5200.28-STD can be overly restrictive, in need of interpretation, or in many cases, non-applicable. The purpose of this paper is to provide insights into determining appropriate security requirements for applications within a specified criteria class. Observations depend to a great extent on the system's user interface, considered as an additional environmental condition.

Introduction

The intent of a computer application is to provide an organization with information processing capabilities in support of its specific mission or goals. It has become apparent that many of the security concepts defined by DoD 5200.28-STD do not directly apply in a general manner to all trusted computer applications. Some of the security features and assurances of DoD 5200.28-STD may be overly restrictive, others in need of interpretation and in some cases, are not applicable at all. This is because the six criteria classes that make up DoD 5200.28-STD, at least in part, assume a user environment with all the risks associated with that of a full-capability general purpose operating system. For many applications however, user capabilities are more restrictive than that of an operating system. Associated with these capabilities is a lower relative risk that coincides with the constrained ability for the user to influence the underlying processing environment.

Associated with each system is a User Interface Set (UIS). A UIS is a collection of processing capabilities provided to the users of the system. These capabilities include system prompts, menus, transactions, utilities, privileges, and operations. The UIS can provide for or preclude users from the following capabilities: execution of programs and transactions; creating and editing messages, documents, and files; creating, compiling and linking application or system programs. At a higher abstraction, a UIS can support an organization's security policy such as restricting individual users or groups of users to specific capabilities, functions and resources. For example, within a hospital system, a doctor may be provided with the capability to perform diagnoses, order tests, and prescribe medicine, while at the same time be prevented from directly performing updates or queries within the financial database.

For a large class of applications, the UIS may define a finite set of possible data accesses. The system's UIS constrains users by enforcing a template of capabilities. This template restricts users to the extent that the system can be viewed as a set of predefined resources (applications, communication links and user groups having specific capabilities). The security attributes which need to be associated with these resources can be defined by design specification. Because of these fixed resource attributes and the absence of a programming environment, security design techniques that are normally not acceptable for general purpose systems can be applied to meet specific security feature and assurance needs. For example, a peculiarity (with respect to DoD 5200.28-STD) that results from the stable functionality of many embedded computer systems, is the ability to allow access control decisions to be unambiguously established during system design time rather than having to be computed at run time. This is known as

the binding of processes and data accesses, or simply early binding, a concept that is described in [7] and further exemplified by an access control triple described in [2]. In the context of DoD 5200.28-STD mandatory and discretionary access controls, subjects are thought of as representing people or the programs that act on their behalf, and objects as representing data or their files. However, in many embedded applications some subset of all the objects are not accessible to human users but are accessible only to the system hardware and software processes - these processes do not act as surrogates for users.

A good example is represented by a Regency Net (RN) terminal. The RN terminal is part of a tactical command and control system developed during the mid and late eighties. Although all users would be cleared for all information and belong to a single user group, security requirements were defined in respect to 1) the flow of data among multi-level resources, 2) the preservation of the integrity of critical data, and 3) the denial and delay of the delivery of critical messages. The concept of a reference monitor and security kernel was interpreted in order to ensure a high level of trust. The RN security kernel consists of an Initializer, to establish the CPU's initial secure state, and the Virtual Machine Monitor (VMM). Because the RN functionality is severely restricted, all subject-object access configurations are bound in the CPU Kernel code such that only those configurations which are both secure and functionally required are possible. The VMM is an extraordinary reference monitor in that it does not compute secure states, as a conventional reference monitor. It implements secure data flows directly rather than acting as an intermediate computational abstraction.

Another application dependent concept is captured by the Clark-Wilson [2] integrity model. The Clark-Wilson model defines an access control triple as the binding of a userID, transaction procedure (TP), and a set of constrained data items (CDIs). This binding indicates not only the ability to specify which users can access which executable program images (as is natural to normal DoD 5200.28-STD discretionary access control), but also implies that these executable program images (TPs) possess privileges in isolation from their invoking user.

Determining Environmental Risk

Defining meaningful computer security requirements for applications has not been a straightforward process. To help improve this situation, the National Computer Security Center (NCSC) published two documents, Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments and its associated Technical Rationale [4]. These two documents provide guidance for choosing an appropriate criteria class per DoD 5200.28-STD, by calculating a risk index based on the system's operating environment. The risk index is partially formulated by comparing the clearance of the least cleared user of the system to the highest classification of information to be processed by the system. The greater the difference between the clearance of the users and the classification of the information on the system, the greater the risk index and the greater the degree of trust that is required. Another environmental condition considers whether the personnel developing the application are authorized access to Secret information (or to the highest level of information to be processed by the system if the information classification is less than Secret). If not, the requirement is for a higher criteria class in order to compensate for the additional environmental risk developers impose on the delivered system.

It has been observed that not all potential risk associated with a system is due to the difference of the clearances of system users and the classification of information and the development environment. Risk may also result from other environmental factors. Another method, using other environmental factors to calculate potential risk has been developed by the Naval Research Laboratory (NRL) [6]. This report provides a more sophisticated approach for calculating risk, taking into account the environmental conditions of CSC-STD-003-85 as well as user processing capabilities and communication paths.

The Need for More Guidance

Both [4] and [6] define security requirements to the granularity of a predefined DoD 5200.28-STD criteria class. In the world of trusted computer applications, seldom have the calculated features and assurances of a criteria class of DoD 5200.28-STD defined a complete and essential set of applicable security requirements. Although this granularity may be at a reasonable level for products that are developed to be general purpose in nature (with no specific application in mind), for many applications minimal user capabilities can be ensured. Applicable security requirements can be defined (at least in part) in terms of the way a human user is intended to interact with the processing environment.

The premise of this paper is that, even though two systems may be defined as having the same risk index, and subsequently would require the same criteria class, applicable security features and assurances associated with these systems can vary significantly.

The Range of the Flexibility of User Interface Sets (R-FUIS)

It is suggested here that there is another significant environmental element that should be considered in determining information security requirements: the Flexibility of the User Interface Set (FUIS). As the flexibility provided through these interfaces increases, so does the risk that a user can influence and undermine the security preserving flow of information. This is regardless of whether the objective of an organization is to maintain the confidentiality of classified information, protect the privacy of individuals, ensure human safety, prevent fraud, or prevent unauthorized modification of educational records.

In order to consider the FUIS in the calculation of security requirements for applications, the FUIS must be measured in some way. The concept of a range in the flexibility of user interface sets (R-FUIS) is introduced. In theory, all systems fall somewhere on the R-FUIS. The relationship between these systems is such, that as systems progress on the range from left to right, applicable security features and assurances (requirements) appear that were not present prior to that point, until a point is reached where all features and assurances of DoD 5200.28-STD are present for a defined level of trust. Systems that fall to the extreme left have the most restrictive interface sets, and have the smallest subset of DoD 5200.28-STD requirements, while systems that fall to the extreme right are considered to have the most flexible interface sets and the most DoD 5200.28-STD requirements. Unlike the Risk Index, The R-FUIS represents a continuum where there is potentially an infinite number of possible points at which a system can be plotted. What is significant about the plotting of a system is where it falls relative to where other systems would fall. All systems can be plotted at some point on the R-FUIS. Depending on where systems fall, observations can be made as to security characteristics and requirements associated with that point. Moving from left to right along the continuum, the R-FUIS accounts for an extreme with no user interface; further along it accounts for a single user system, still further, multiple users but of a homogeneous nature (same role). Beyond the mid point, there are considerations for multiple users each belonging to a specific user role, while at the extreme right individual users with individual needs and privileges to access information are taken into account. (Instances of a role can include: a Doctor or Nurse within a hospital system; a Loan Officer or a Teller within a banking system; or a Traffic Analyst or Cryptanalyst within an intelligence system.) The concept of the R-FUIS is illustrated in Figure 1 below.

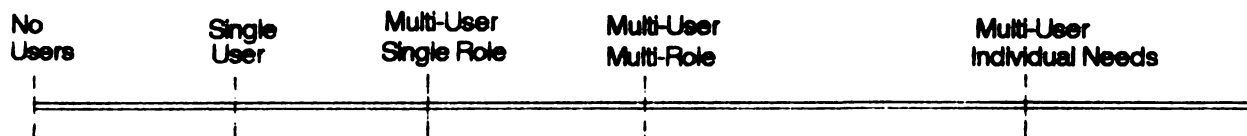


Figure 1. The Range in the Flexibility of User Interface Sets

A R-FUIS can be associated with each criteria class of DoD 5200.28-STD. The result is a two dimensional view of DoD 5200.28-STD, where there are 6 rows each representing a criteria class with the associated R-FUIS representing the range of applicable security requirements for that criteria class. An appropriate criteria class can first be determined through the use of current environmental guidelines [4], [6]. The position of the system on the R-FUIS for the criteria class can then provide insight as to applicable security requirements for the system.

The R-FUIS ranges from the most primitive or restrictive interface set, such as that of a black box with no user interface, to the most flexible interface, such as the full capabilities of a general purpose operating system. Both of these systems may process the same type and classification of information but because of the extreme differences in the FUIS, security requirements will differ greatly. The black box can be thought to have inherent security protection such as the inability of a human to alter its processing (except by physically removing its chassis and reprogramming it). However, a programming environment does not come as part of the system. It would need to be reprogrammed on another environment and down-loaded to the black box. On the other hand, the operating system supports the ability to create executable programs and alter existing ones. With the operating system interface, the following risks exist: the potential for introduction of a trojan horse, trap door, or virus; a program that mimics the operating system software and steals passwords; or the alteration of security relevant software. All these risks are a result of an operating system's natural user interfaces, while none of these risks are associated with the black box.

In order to reduce operational risk, security requirements are imposed throughout the system development cycle. These requirements must then be evaluated to ensure a secure operating environment. When a certification is performed for an application to operate in a specific environment, the certification should be an evaluation of the applicable security requirements associated with that system type. Because this evaluation would be conducted against some subset of requirements of a specified criteria class, it may not be appropriate to assign a criteria rating to the system, but instead indicate that the system mitigates known security risk, and is known to implement some list of security features and some level of assurances.

Defining Applicable Security Requirements

The R-FUIS can be subdivided in several ways depending on the UIS associated with the various types or categories of systems. By subdividing the R-FUIS into various types of systems and defining the security characteristics belonging to each of the types, the R-FUIS can be used to provide insight in the definition of security requirements. It is acknowledged that the use of the R-FUIS does not provide an absolute solution to defining security requirements for trusted applications. However, a widely agreed upon definition of the R-FUIS could provide guidance and establish precedence as to applicable security requirements that could be used from project to project, making the definition of applicable security requirements less of a subjective process.

By continuously subdividing the R-FUIS into smaller and more numerous pieces, the R-FUIS will be more helpful in the definition of security requirements. However, it is not the intent here to define an extensive list of possible types of computer systems. Instead, four types of systems are described and plotted on the R-FUIS to demonstrate how the R-FUIS can be used. By plotting a system on an even sparsely defined R-FUIS, guidance can be provided as to the system's applicable security requirements.

Observations on the R-FUIS

In the examples presented in this section, security features of DoD 5200.28-STD are described as they apply for each type of application. Figure 2 below summarizes these observations, providing one view of the R-FUIS.

No Users	Single User Single Role	Multi-User Single Role	Multi-User Multi-Role	Multi-User Individual Needs	
<ul style="list-style-type: none"> - Security Policy - Resource Labeling - Direct Data Flow Controls 		<ul style="list-style-type: none"> - I & A - User Accountability (interpreted) - Object Reuse (implementation Dependent) 	<ul style="list-style-type: none"> - Discretionary Execution (Administrative) - Object Reuse - User Accountability (interpreted) 	<ul style="list-style-type: none"> - Isolation (interpreted) - User Label (Single Session) 	<ul style="list-style-type: none"> - DAC (User Specified) - Isolation - User Accountability - User Label (Multi-Session)
Black Box		Limited Transaction Based System	Role Enforcing Transaction Based System	Full Capability Operating System	

Figure 2. Example Systems Mapped onto the R-FUIS

Black Box Systems

For the most restrictive systems, which will be typed Black Box, many of the security features and assurances of DoD 5200.28-STD are not applicable. For the Black Box system, no humans have the ability to directly influence (read, or write) its objects. These systems are usually components incorporated to perform one or more specific control functions within a larger system. A Black Box system can be thought of as a "closed" system that contains only embedded processes where none of these processors contain a direct man-machine interface. In fact, in many applications a Black Box provides specialized services to a larger system which is totally transparent to human users. Although a Black Box system does not support the direct needs of human users it still may be trusted to perform a vital processing function. Military Black Box applications are numerous but they can include civil and commercial applications as well. For example, the routing of mail, aircraft avionics, robot control, and transportation switching devices. What is significant is that the execution of the controls of the device can be assumed to be free of human interaction.

Obviously for Black Box systems direct user related features such as identification and authentication, and user accountability are not applicable. In addition, making access control decisions based on the identity of or an attribute associated with a direct user does not make sense within a Black Box environment -- no Discretionary or Mandatory Access Controls with respect to the UIS. Also, there is not a requirement for a trusted path between a system user and the TCB.

The applicable security features can be viewed as the smallest subset of DoD 5200.28-STD requirements. These security features are relevant for all systems of this specified R-FUIS and the specified criteria class. All systems that fall to the right of the black box will include these fundamental features in their list of security requirements. Probably the most fundamental of all security requirements is that of a security policy. It is the security policy that defines what it means for a system to be secure. All other security features and assurances are present only in support of that policy. This policy may ensure that information of varying levels does not get mixed while in the local system. Because there exists a security policy there must be an associated mechanism to implement the policy. For many Black Box applications, controls are flow-oriented where the policy is preserved through flow decisions that could be considered at design time rather than at run time. Object reuse more than likely would not be applicable. Pools of previously used memory are not available for subsequent scavenging.

Because there is no concept of application software as opposed to system software, there is a de-emphasis on the need for isolation techniques, such as domains of execution. Strong physical and procedural controls can be applied during system development to ensure an execution environment that is free of malicious code. Tools can be applied to ensure all flows are security preserving. Lastly, the absence of a user interface goes a long way in ensuring that the secure environment stays secure.

Limited Transaction Based Systems

The next type of system that will be described is a Limited Transaction based system. A good example of a Limited Transaction based system is an Automatic Teller Machine (ATM). For these systems there is the presence of a man-machine interface, although it is quite limited. All users generally belong to one user group. Although this group may potentially be quite large, the users are constrained to a narrow set of processing capabilities and all perform the same functions.

For example, there may be a menu where selections can be made via a simple interface device such as a numeric key pad. For a Limited Transaction based systems, users are precluded from accessing information other than through well defined inter-related sets of processes known as transactions. For systems of this type, subject-object access configurations can be pre-specified and bound in such a way that only those access configurations which are both secure and functionally required are possible. Authorized users are first identified and then given "select" access to a limited set of transactions, which in turn have access privileges to information. By making a selection on a menu, a transaction is started and a specified and controlled set of activities occur. This transaction will access and manipulate specific files based on the type of transaction being invoked. The only access to information is defined by specification and determined during the system design.

For many Limited Transaction Based Systems most of the objects are not accessible to human users but are accessible only to the system hardware or software processes. It is the data and the flow of information associated with these processes that are security relevant rather than humans accessing information. There may be some number of secure data communications links where the information may be multilevel in nature. The system must be trusted not to mix information of a higher level with that of a lower level where it would then be perceived as being of the lower level (this is the mandatory policy). While supporting secure links, this type system could also support a link that is not secure (unencrypted) which would have to be considered unclassified. Although there exists multiple users each belongs to the same role and would possess the same security clearance. The users security attributes would be considered fixed for which data would flow accordingly.

Identification and authentication mechanisms are generally used for accountability purposes alone. Discretionary access controls (per DoD 5200.28-STD) do not apply in the sense that user's can specify what other users have access to the files. For the ATM example, the user's Personal Identification Number (PIN) may be used as a parameter within a transaction for the purposes of retrieving the correct account record. No capability exists for users to grant or revoke privileges other than through disallowing access to the system.

Role Enforcing Transaction Based Systems

A Role Enforcing Transaction based system is similar in many ways to a Limited Transaction based system in that the access to information is granted or configured in terms of a process or transaction ID. For this type system all users have a proper clearance and need-to-access within their role. Therefore a user security level can be assumed (no need to specify security session level) and as with the Limited Transaction based system the flow of data can be considered accordingly.

The access control mechanisms principally enforce the rigid concept of least privilege and not the richer mechanisms implied by discretionary access control. Role Enforcing systems restrict access to information based on the role a user chooses. A given user may have the ability to move from role to role if he is authorized, but the user can only take on one role at a time. A user would choose a role via a menu selection, at which point a validation would be conducted to ensure the user can take on that role. The user's identity is critical for both validating that his role is legal and accounting for his actions. The method of enforcing need-to-access is not implemented through a strict discretionary access control mechanism as defined in DoD 5200.28-STD, but rather through a series of mechanisms and characteristics of the system. First, a check is made as to whether a user can take on a selected role. If the user is

granted access to the role, he is given execute access to a series of transactions that are presented to him. The user is presented only with a list of transactions that has been specified to support his role. After the selection of a transaction, the user specifies parameters associated with the transaction and hits a return key or clicks a mouse to effectively start the transaction. The transaction is deterministic, performing activities in support of the user's role. Individual users can be added and deleted to each role. Role membership is most likely centrally administered rather than at the discretion of the individual users. Role membership may be altered through administrative and procedural controls. The capabilities represented by each role would be static in nature and could not easily be changed.

A role enforcing system in many cases supports a type of mandatory (non-discretionary) access control policy. Consider the hospital example described above. The system may provide a physician with the capability to perform a diagnosis, prescribe medication, and add to a record of treatments performed on a patient. Here roles would be created to preclude the physician from giving away the capability to perform a diagnosis or prescribe medicine to a non-physician. It is also a mandatory policy that users are prevented from modifying the record of treatments maintained for each patient.

The deterministic characteristics of the system is an important consideration in maintaining an audit trail. UserID, time, transactionID, and transaction parameter entries, in many cases are sufficient in holding users accountable for their actions. However, any one transaction may invoke numerous processes across several platforms and access countless data items. To audit each successful access would provide an overwhelming amount of information.

Full Capability Operating System

The most complex of all human interface sets is associated with an operating system. A Full Capability Operating system supports many users simultaneously while at the same time enforces both a mandatory and discretionary access control policy with respect to users and information. Discretionary access control mechanisms allow users to specify, using their discretion, the access privileges other users have to the objects they own. Although discretionary access controls are intended to be the principal means of enforcing need-to-access, these controls are inherently insecure. Because of a real possibility of users introducing malicious software, a more reliable mechanism than a discretionary access control mechanism must be provided, namely mandatory label-based access controls. These mandatory access controls are typically provided through the enforcement of the rules of the Bell & LaPadula security model [1].

Within an operating system environment, mandatory security rules must consider fixed resources, the assumption of malicious software, users with different security clearances, and data of multiple security levels. Here a run time access control intermediary must be provided that enforces the rules of the Bell & LaPadula security model. This access control intermediary is based on the concept of a reference monitor and may be implemented as a security kernel, depending on the risk index calculated for the application. Before a subject is permitted to have access to an object, a run time check is performed to ensure that the proposed access conforms to the set of underlying security rules governing the system. The theory of security in an operating system is induced from an initial secure state and a demonstration of the preservation of security for every operation subsequently allowed by the reference monitor. In essence, the purpose of a reference monitor is to compute security states for the system.

To preclude the ability of a subject from having simultaneous read access to an object of a higher classification and write access to an object of a lower classification (where there is the potential for an illicit flow of information) a rule similar to the *-property must be enforced. The *-property requires the subject's security level to be equal to or lower than that of an object for which the subject is attempting to gain write access. Because the classification of the object can be lower than the highest security clearance of the user, a method must be provided to allow the user to establish a session level lower than that of his or her highest security clearance. If that user needs to read an object of a higher classification then the object to which the user just wrote, the user's session level must be raised to a level at least as high as the level of the object to be read.

Application software can be added or modified at any time during the operational life cycle of the system, and often is created by the very subjects to which the rules of the security policy apply. In order to provide a reasonable degree of assurance that applications software cannot by-pass or alter the security policy enforcing mechanisms, security mechanisms must reside in a separate and more privileged execution domain than that of the applications software.

Further, to preclude a malicious user from stealing an unsuspecting user's password through the creation of a program that mimics a legitimate password request, a trusted path must be provided. A trusted path would ensure a reliable communication channel between system users and security relevant software.

Conclusion

Computer applications range from a black box which has no direct system user, to a very flexible system supporting many human users simultaneously, where these users have the ability to create executable images of programs and share information on a discretionary basis. It is reasonable to believe that although these systems may have the same calculated risk index per [4], they should implement only security mechanisms that are applicable to their operational environment.

The R-FUIS (Range in Flexibility of the User Interface Set) has been introduced to provide insights to determining security requirements for a system based on characteristics of the application as well as other environmental conditions identified in [4] and [6]. It is acknowledged that the use of the R-FUIS will not provide an absolute solution to determining security requirements, but it is our hope that the determination of applicable security requirements may become more of a methodology.

By further defining the R-FUIS innovative security design techniques can be uniformly applied across new secure application development efforts. This definition can be provided through the consideration of existing and future secure application development cases studies. The result would be new and increasing numbers of innovative security techniques uniformly applied within appropriate (better defined) security environments. Through a peer review process new security techniques can be accepted as legitimate methods in combating environmental risk. The existence of criteria, criteria interpretation, and guidelines should never result in the stifling of new and innovative approaches for applying security within our systems.

References

- [1] Bell, D.E., LaPadula, L.J., Secure Computer Systems: Mathematical Foundations, ESD-TR-73-278, Vol. I, Electronic Systems Division, Air Force Systems Command, November 1973.
- [2] Clark, D.D., Wilson D.R., "A Comparison of Commercial and Military Computer Security Policies," Proc. on Security & Privacy, Oakland, CA, pp. 184-194, IEEE Computer Society, April 27-29, 1987.
- [3] Department of Defense, Department of Defense Trusted Computer System Evaluation Criteria, Department of Defense 5200.28-STD, December 1985.
- [4] Department of Defense Computer Security Center, Computer Security Requirements - Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments, CSC-STD-003-85, National Computer Security Center, June 1985.
- [5] Goguen, J.A., Meseguer, J., "Security Policies & Models," Proceedings 1982 Symposium on Security & Privacy, Oakland, CA, pp. 11-20, IEEE Computer Society, April 1982.
- [6] Landwehr, C.E., Lubbes, H.O., "An Approach to Determining Computer Security Requirements for Navy Systems," NRL Report 8897, Naval Research Laboratory, May 1985.
- [7] Norton, W., Reider, L., "Computer Security in Regency Net," DRAFT.

APPARENT DIFFERENCES BETWEEN THE U.S. TCSEC AND THE EUROPEAN ITSEC

Dr. Martha A. Branstad
Dr. Charles P. Pfleeger
Trusted Information Systems, Inc.
3060 Glenwood, MD 21738

Dr. David Brewer
Gamma Secure Systems, Ltd.
Diamond House
149 Frimley Road

Mr. Christian Jahl
Mr. Helmut Kurth
IAGB Software Technology
EinsteinstraBe20
D-8012 Ottobrunn

The U.S. *Trusted Computer System Evaluation Criteria*, called the TCSEC [TCS85] which was first published in 1983 and revised in 1985, has become an accepted standard for the evaluation of trusted systems. Not only is it used in the U.S. for evaluations by the National Computer Security Center (NCSC), it has also been adopted by NATO for the evaluation of systems for use in NATO installations. More recently, in May 1990, a group of four nations, France, Germany, the Netherlands, and the United Kingdom, produced a first draft of its *Information Technology Security Evaluation Criteria*, called the ITSEC [ITS90]. The ITSEC shows clearly that the thinking of the computer security community has been heavily influenced by the TCSEC, but the ITSEC also addresses some issues in ways that are very different from the TCSEC. A meeting was held under the sponsorship of the European Commission in Brussels on September 25-26, 1990, at which members of the four nations discussed their reactions to comments received since the publication of the ITSEC and presented their opinions of changes that should be made to the ITSEC.

As a method of understanding the ITSEC more completely, it was analyzed to determine the impact that compliance with an F5/E5 rating would have upon a B3 targeted system that is under development. This analysis led to a discussion with ITSEC authors from both Germany and the United Kingdom that helped to clarify many questions concerning specific wording and concepts of the ITSEC and its relationship with the TCSEC. It should be noted that the views presented here are the authors' and not official statements from the various organizations with which they are affiliated.

BACKGROUND

TCSEC Overview

Briefly, the TCSEC establishes six levels of evaluation: C1 and C2 provide discretionary access control only, B1, B2, B3, and A1 provide both discretionary and mandatory access control. Beginning at B2 and progressing to B3 and A1, the requirements for assurance — measures that inspire confidence that the implementation of the system truly and rigorously enforces its stated security policy — play a very significant part in the evaluation. Each TCSEC rating, called a digraph, is thus a combination of a particular set of features and a necessary minimum degree

Copyright © 1990 Trusted Information Systems, Inc.

Portions of the work reported in this paper were performed on DARPA contract #F30602-89-C-0125

of assurance. Since publication of the TCSEC, there has been discussion in the computer security community over the advisability of this bundling of features and assurance. There has also been considerable discussion regarding the predetermined collections of features represented by each digraph class; little room is available for the development and evaluation of a trusted system that had goals other than maintaining confidentiality.

ITSEC Overview

In consideration of these two concerns, the authors of the ITSEC chose to separate the functionality of a trusted system¹ from ratings of its assurance², and to expand the range of functionality. Each evaluated trusted system would be awarded two descriptors: one denoting the functionality the trusted system presents, and the second, denoting the assurance of correct implementation of that functionality. Currently, there are ten exemplary predefined functionality classes, F1–F5 and F6, F7, F8, F9, and F10. The classes F1–F5 correspond closely with functionality required at the TCSEC classes C1, C2, B1, B2, and B3³, respectively. The five remaining predefined classes represent integrity, availability, data communications integrity, data communications confidentiality, and data communications integrity and confidentiality, respectively. A trusted system can be evaluated against more than one of these classes of functionality, if appropriate. There is also the potential for a developer to define a new class of functionality, if these classes fail to describe a particular trusted system adequately. Assurance is recognized as a combination of correctness and effectiveness. Six correctness ratings were defined as E1–E6; these combine with the judgement of effectiveness of the security functions and mechanisms. These assurance ratings were intended to correspond generally to the TCSEC assurance requirements for C1, C2, B1, B2, B3, and A1 trusted systems, respectively. Thus, given trusted systems might achieve ratings of F3/E2 or F4–F7/E4, for example.

Because the requirements for the F1–F5 and E1–E6 classes so closely resemble the TCSEC requirements, it is reasonable to try to identify points where ITSEC and TCSEC ratings coincide. The annex to Appendix A of the ITSEC lists the intended correspondences from the ITSEC to the TCSEC, that is, functionality/assurance combinations that are at least as strong as TCSEC digraphs. Table 1 shows these intended correspondences. The ITSEC criteria contain a number of requirements that do not appear in the TCSEC explicitly, and thus, according to the ITSEC, direct equivalence of evaluation levels is inappropriate.

¹ The ITSEC distinguishes between a "product" which is intended to be useful in a wide range of application environments, and "system" which is designed and built for the needs of a specific type of environment. The term "trusted system" is used in this paper to denote either a product or a system that is being evaluated under one of the criteria.

² The separate evaluation of functionality and assurance was first documented in the German II Security Evaluation Criteria [GISA89].

³ TCSEC class A1 was omitted from this list because its functionality requirements are identical to those of class B3.

Table 1 Intended correspondence from ITSEC to TCSEC

<u>ITSEC Class</u>	<u>TCSEC Class</u>
F1/E2	C1
F2/E2	C2
F3/E3	B1
F4/E4	B2
F5/F5	B3
F5/E6	A1

However, it is also true that there are requirements in the TCSEC that were not replicated in the ITSEC. Thus, the correspondence of Table 1 does not work in either direction. Still, it is a good starting point for analyzing the differences between the two evaluation criteria.

TCSEC/ITSEC CORRESPONDENCE

As a method of understanding the ITSEC more completely, it was analyzed to determine what was involved in achieving compliance with an F5/E5 rating and what the impact would be upon a B3 targeted trusted system that is under development. First the ITSEC was examined to determine (a) how a trusted system could be evaluated as F5/E5 yet fail to meet B3, and (b) how a system could be evaluated as B3 yet fail to meet F5/E5. The intention of this analysis was first, to understand better the nuances of the requirement language, and second, to determine what additional work a developer would need to do in order to produce a system that met both criteria. After the identification of apparent differences, some of the TCSEC authors, some of the ITSEC authors, and some others met to determine if the apparent differences were really intended. Among the ITSEC authors, there were representatives both from Germany and the United Kingdom. The remainder of this report describes the outcome of that meeting. It should be noted that the participants at the meeting were presenting their own views of the sense of the groups of which they are a part.

Attributes of Trusted Systems that could Pass F5/E5 but Fail B3

The following sections present statements from the TCSEC and the ITSEC, followed by a brief statement of intention from the authors. Section or page number references are included. **Bold face type** is reproduced from the original; underlining is used to draw attention, but is an addition to the original text.

1. No DAC or DAC does not apply to all named objects.

TCSEC: §3.3.1.1 ...These access controls shall be capable of specifying, for each named object, a list of named individuals, ...

ITSEC: F5, p. 104 (§ Administration of rights) The system shall be able to distinguish and administer access rights between each user and/or user group and the objects which are subject to the administration of rights.

Discussion: The ITSEC drafters indicated that it was their intention to have this F5 requirement correspond to B3. Rework of the ITSEC wording could make the equivalency more evident.

The ITSEC drafters wanted to avoid the term "named object" since there has been some controversy about its meaning in the TCSEC. There is ambiguity in the phrasing of the ITSEC, however. The ITSEC drafters intended for this requirement to apply to all objects defined by and visible to users. In any system there are three classes of objects that might be subject to administration of rights: i) those that are defined by and visible to users, ii) those that are defined by the system and may be directly or indirectly visible to users, and iii) those that are defined by the system but used at a level below that at which access control policy is enforced. The objects of class iii) are not subject to the administration of rights but must be considered in covert channel analysis. Those of classes i) and ii) are subject to mandatory access controls. The objects of class i) are subject to discretionary access controls.

The ITSEC drafters acknowledge that they would like less restrictive language to apply to lower assurance levels. Progressively more stringent requirements for applicability of access control were desired as the assurance level rose within a given functionality class, but given the separation between functionality and assurance, it is very difficult for the ITSEC authors to impose such progressive requirements within one functionality class. The ITSEC authors are searching for a way to delineate those objects that must be subject to administration of rights; it may be that the categorization of class i, class ii), and class iii) above is a way to achieve this.

2. MAC does not apply to all resources directly or indirectly accessible by subjects external to the TCB.

TCSEC: §3.3.1.4 The TCB shall enforce a mandatory access control policy over all resources...

ITSEC: F5, p. 106 (§ Verification of rights) With each attempt by users or user groups to access objects which are subject to the administration of rights, the system shall...

Discussion: This is the same problem as above. The wording of the TCSEC is open to some interpretation (e.g., whether or not it is intended to apply to a system console). The intention of the ITSEC authors was to be equivalent to their perception of the meaning intended in the TCSEC.

3. Encrypted storage is not cleared before reuse.

TCSEC: §3.3.1.2 No information, including encrypted representations of information, ... is to be available to any subject that obtains access to an object that has been released back to the system.

ITSEC: F5, p. 107 (§ Object Reuse) All storage objects returned to the system shall be treated before reuse by other subjects, in such a way that no conclusions can be drawn regarding the preceding content.

Discussion: The distinction between the TCSEC and the ITSEC was intended. If encryption is judged adequate to protect data in transmission or storage, then it should also be adequate to prevent any determination of plaintext from ciphertext that may be obtained from a reused object.

4. Human readable labels are provided, but not at places specified.

TCSEC: §3.3.1.3.2.3 The TCB shall mark the beginning and end of all human-readable, hardcopy output.

ITSEC: F5, p. 106 (§ Administration of rights) The system shall mark human readable output with attribute values. The values of the attributes shall be determined according to the rules laid down in the system. Authorized users shall be able to specify the printable name of each attribute value and the location of the corresponding marking.

Discussion: The distinction between the TCSEC and the ITSEC was intended. It should be a matter of agreement between system user, system designer, and system security administrator precisely where the labels are placed.

5. The trusted path mechanism is not available for the user to change security level or to query the system about security level.

TCSEC: §3.3.2.1.1 The TCB shall support a trusted communication path between itself and users for use **when a positive TCB-to-user connection is required (e.g., login, change of subject security level)**.

ITSEC: F5, p. 106 (§ Administration of rights) A user shall be notified immediately of any change in the security level associated with that user during an interactive session. The user shall be able at all times to display all the subject's attributes.

Discussion: The authors of the ITSEC have consciously tried to separate functionality requirements from mechanisms by which those requirements are implemented. They do not wish to be prescriptive of specific mechanisms in their requirements. However, without a trusted path in an F5/E5 trusted system,

there is a possibility that an untrusted process could masquerade as the login process, thereby capturing a user's login and authentication data. This presents a security threat which, if included in the trusted system's Security Target (the actual baseline against which the system is evaluated, see Chapter 2 and page 63 of the ITSEC) would need to be identified and countered, for an E5 rating.

The ultimate difference here is that the TCSEC authors felt strongly enough about the need for a trusted path at the B3 level to mention it explicitly. In the ITSEC the issue is handled through the suitability of functionality and strength of mechanism requirements of assurance - effectiveness (§4.2.1 and §4.2.4). The trusted path is not an explicit requirement of the ITSEC, but it, or a similarly effective mechanism, would be needed to counter the threat of a masquerade of the login procedure. Explicit specification of implicit effectiveness requirements would lead to greater clarity of actual ITSEC requirements. This is another instance in which a low assurance class might not necessitate such a strong mechanism as the trusted path, which would be very appropriate at the higher assurance levels.

6. No identifiable reference monitor exists.

TCSEC: §3.3.4.4 Documentation shall describe how the TCB implements the reference monitor concept and give an explanation why it is tamper resistant cannot be bypassed, and correctly implemented.

ITSEC: no such explicit requirement exists

Discussion: The identification of a TCB and implementation of protection through the reference monitor concept was seen by the ITSEC authors as being associated with specific security policies and prescriptive of particular mechanisms. On the other hand, the ITSEC authors recognize the desirability of the reference monitor concept and TCB in many instances. They intend to use the effectiveness component of assurance to exclude systems that fail to use the reference monitor concept when it would have been more appropriate than whatever approach the developers used. A need for greater specificity of the effectiveness requirements is recognized, but such specificity is difficult to achieve while maintaining the goal of policy generality. It is of course open for the person defining the security target for an ITSEC F5/E5 evaluation to mandate the use of particular types of mechanism, for example a reference validation mechanism implementing the concept of a reference monitor. Clearly, this then constrains the developer to follow a TCSEC-like approach.

7. TCB not appropriately structured

TCSEC: §3.3.3.1.1 The TCB modules shall be designed such that the principle of least privilege is enforced... It shall make effective use of available hardware

to separate those elements that are protection critical from those that are not. The TCB shall be designed and structured to use a complete, conceptually simple protection mechanism with precisely defined semantics. This mechanism shall play a central role in enforcing the internal structure of the TCB and the system. The TCB shall incorporate significant use of layering, abstraction, and data hiding. Significant system engineering shall be directed toward minimizing the complexity of the TCB and excluding from the TCB modules that are not protection critical.

ITSEC: no such explicit requirement exists

Discussion: This issue is essentially the same as the trusted path issue explored above. All of these structuring requirements were seen by the ITSEC authors as prescribing mechanisms that would be very appropriate in many situations but might not be appropriate in all. Their intention is to treat this issue in the effectiveness section. It is likely that this issue will be addressed in a manual for evaluators, by way of example.

Observation: The TCSEC and ITSEC authors recognize these last two points as definite differences between the TCSEC and the ITSEC. If a developer wants to achieve both F5/E5 and B3 evaluations, the developer will want to plan to meet both sets of requirements. The ITSEC authors recognized that exact correspondence with the TCSEC was impossible within the ITSEC scheme. They have indicated that their intention was that trusted systems evaluated at the F5/E5 or the B3 level should yield equivalent assurance of enforcement of the defined security policy.

Attributes of systems that could pass B3 but fail F5/E5

1. Fail to provide a read-only access mode.

TCSEC: §3.3.1.1 These access controls shall be capable of specifying, for each named object, a list of named individuals... with their respective modes of access to that object.

ITSEC: F5 p. 104, (§ Administration of rights) It shall also be possible to restrict a user's access to an object to those operations which do not modify it.

Discussion: Since many commercial clients are concerned with controlling the ability of a user to modify information but are not concerned with whether the user can read the data, the existence of read-only access mode is deemed important.

2. Fail to provide labels for subjects and objects internal to the TCB.

TCSEC: §3.3.1.4 The TCB shall enforce a mandatory access control policy over all resources... that are directly or indirectly accessible by subjects external to the TCB. These subjects and objects shall be assigned sensitivity labels...

ITSEC: F5, p 105 (§ Administration of rights) In addition, the system shall provide all subjects and objects... with attributes.

Discussion: The words as currently written do not convey the intended meaning of the ITSEC authors.

3. Fail to provide design documentation for non-TCB elements.

TCSEC: no requirement

ITSEC: §3.6.1.1.1 The sponsor shall provide the following documentation... structured description of the detailed design.

Discussion: The ITSEC authors indicated that their intent was for design documentation to be required only for parts of the system critical to the enforcement of security.

4. Use a non-validated compiler.

TCSEC: no requirement

ITSEC: §3.6.1.2.2 b) The used compilers shall be validated e.g., approved by an appropriate body.

Discussion: This requirement was discussed at the Brussels conference; it is expected that the requirement will be reworded.

5. Use a non-rigorous notation for the architectural design.

TCSEC: no requirement

ITSEC: §3.6.1.1.3 c) The architectural description shall use some form of rigorous approach and notation.

Discussion: The concept is appropriate for reconsideration by ITSEC authors. At the Brussels conference, a number of inconsistencies were reported in Chapter 3 of the ITSEC. Rewording of requirements is likely.

6. Provide no mathematical analysis of design refinements.

TCSEC: no requirement

ITSEC: §3.6.1.1.4 c) **Mathematical reasoning shall be used to show that each hierarchical level is a refinement of the previous level.**

Discussion: The intention of the ITSEC authors was to support traceability between levels of the design. The term "logical" is perhaps a better choice than "mathematical" to express the ITSEC authors' intent of supporting traceability.

7. Include functions with side-effects.

TCSEC: no requirement

ITSEC: §3.6.1.1.4 c) An analysis of the detailed design for side effects shall indicate that none exist and that no additional functionality is present which would allow the security mechanisms to be bypassed.

Discussion: This distinction is both semantic and substantive. In some European evaluation circles, a "side effect" is something that a trusted function does which is security-relevant and which the function is not intended to do. In the U.S., "side effect" is used more broadly to mean any effect beyond the defined functionality. The narrower usage is consistent with the intention of the TCSEC as described under Security Testing (§3.3.3.2.1) as "their [testers'] objectives shall be: to uncover all design and implementation flaws..." The intention of the ITSEC authors was to prohibit side effects that could undermine the security policy enforcement. The difficulty in designing a completely side-effect free product is acknowledged. The ITSEC wording could be clarified.

8. Fail to map security functions to mechanisms.

TCSEC: no requirement

ITSEC: §3.6.1.1.4 b) It [the specification document] shall also map security functions to mechanisms and functional units.

§3.6.1.1.4 c) It shall be shown that the security mechanisms provide the security functions stated in the security target.

Discussion: This requirement was intentionally included by the ITSEC authors; however, it is anticipated that this requirement might be met by a level-by-level analysis as part of the philosophy of protection required by the TCSEC.

9. Fail to identify all non-local variables.

TCSEC: no explicit requirement

ITSEC: §3.6.1.1.4 b) **All variables used by more than one functional unit shall be defined at the lowest level of the specification and their purpose shall be explained.**

Discussion: This requirement was intentionally included by the ITSEC authors as an extension of the TCSEC.

10. Provide inadequate configuration management tools by (a) failing to illustrate item relationships or (b) failing to identify security relevant changes.

TCSEC: no explicit requirement

ITSEC: §3.6.1.2.2 b) **All objects created during the development process, such as design documents, source code, and other dependent data shall be subject to configuration control. ... In the event of a change of any of these objects, the tools shall be able to identify all objects affected by this change. The tools shall support the determination of whether a change is security relevant.**

Discussion: This requirement was intentionally included by the ITSEC authors as an extension of the TCSEC. Part (b) was not intended to be extreme; its intention was to force the developer to separate the code into a part that was security relevant and a part that was not. Changes to only the security relevant code were to be tracked; and any change to security relevant code was to be tracked.

11. Provide inadequate vulnerability analysis.

TCSEC: No general vulnerability analysis requirement exists, but sections 3.3.3.1.3 and 3.3.3.2.1 require covert channel analysis and penetration testing, respectively.

ITSEC: §3.6.1.1.4 b) **The design vulnerability analysis shall determine any ways in which it is possible for a user of the TOE to deactivate, bypass, corrupt, or otherwise circumvent the security afforded by the TOE as configured by a security administrator.**

§3.6.1.1.5 b) **The implementation vulnerability analysis shall determine any ways in which it is possible for a user of the TOE to deactivate, bypass, corrupt, or otherwise circumvent the security afforded by the TOE as configured by a security administrator, based on the source code. It shall identify covert channels.**

Discussion: The ITSEC authors recognize that defining what constitutes an adequate vulnerability analysis is difficult, especially for systems and products that span a collection of varying security policies. The authors intend to include more specific guidance in the manual for evaluators. For the present, in confidentiality-preserving systems, the authors' intent was that penetration testing and covert channel analysis suffice for a vulnerability analysis.

Comment: With respect to penetration testing, the ITSEC authors expect that the developer and the evaluators will be in a cooperative, not an adversarial, relationship. The developer will undoubtedly perform some amount of penetration testing; notes on the analysis required to hypothesize penetrations and the tests performed to validate the hypotheses will reduce the amount of work the evaluators need to perform for penetration testing.

Moreover, covert channel analysis is only applicable under certain circumstances, i.e., where the security policy concerns confidentiality and the threat of covert channel attack is included in the Security Target. Thus it may be better to move the covert channel analysis requirement (pages 57, 65, and 73 of the ITSEC) from Chapter 3 to the predefined functionality classes F4, F5, and F6.

12. Fail to use test coverage tools.

TCSEC: no requirement

ITSEC: §3.6.1.1.5 b) The test documentation shall contain plan, purpose, procedures and results of the tests, the extent of test coverage, the metric used for calculating extent, and a justification why the coverage is sufficient.

Discussion: The intent of the ITSEC authors was to require evidence of degree of test coverage by developers for individual functional units and for the trusted system as a whole. Because of the size and complex functionality of some trusted systems, extensive, let alone complete, test coverage is difficult to achieve. The developer and evaluator should know and be able to document what has been achieved through testing.

13. Fail to provide trusted distribution.

TCSEC: no requirement

ITSEC: §3.6.2.2.2 b) A procedure approved by the certification authority for this assurance level shall be followed, which guarantees the authenticity of the delivered TOE.

Discussion: This is an intentional requirement that extends the TCSEC.

14. Fail to provide checks against maintenance without agreement of the security administrator.

TCSEC: no requirement

ITSEC: §3.6.2.2.3 b) No maintenance shall be possible without the agreement of the administrator.

Discussion: This is an intentional requirement that extends the TCSEC. Constraints in the trusted system are required so that the agreement of the administrator is assured before on-line maintenance is performed.

15. Fail to identify all security mechanisms and their interrelationships.

TCSEC: no explicit requirement

ITSEC: §3.6.1.1.4 b) It [the specification document] shall explain the realization of all security functions through all levels of design hierarchy, and identify all security mechanisms.

Discussion: This requirement was intentionally included by the ITSEC authors. The requirement should be met by the philosophy of protection required by the TCSEC.

16. Fail to provide security functions that are adequately easy to use.

TCSEC: no requirement

ITSEC: §4.3.1 a) Under this aspect of assessment, the security functions and mechanisms of the TOE are assessed for their practicality of use in actual live operation.

Discussion: This requirement was discussed in Brussels. It is likely to be reworded to make it more objective.

SUMMARY

As indicated by the previous sections, although they are similar, the F5/E5 and B3 requirements are not identical. Without explicit effort to meet additional requirements, a system targeted at one rating would not meet the other. An F5/E5 targeted system must meet additional or more constrained requirements on system structure, trusted path, labels on printed output, and object reuse. A B3 targeted system must take additional effort with system development practices, trusted distribution, and maintenance controls. Expressed another way, an F5/E5 system has more architectural freedom than B3 in achieving high assurance of confidentiality while a B3 system is less constrained in its development practices. For a B3 targeted system to achieve an F5/E5 rating, the following additional requirements must be met:

- * Provide detailed design specifications with mappings between design levels.
- * Use more elaborate configuration management tools.
- * Use test coverage tools for unit testing.
- * Develop trusted distribution procedures.
- * Incorporate security administrator authorization for maintenance.

Analyzing the TCSEC to determine the impact of compliance with F5/E5 requirements upon a B3 system proved to be a very useful technique for determining the relationship between the ITSEC and the TCSEC. It caused the questions to become specific enough so that productive dialogue could take place with ITSEC authors to clarify the meaning of particular requirements. This resulted in better understanding of the document as a whole by those more familiar with the TCSEC and realization of the implications of ITSEC wording by its authors. In thirteen cases, specific intentional differences between the TCSEC and ITSEC were identified. In two instances, the participating authors felt that changes in the ITSEC were likely. Wording changes to clarify intent were deemed essential in nine cases. In two instances, the authors felt that clarification would occur in the manual for evaluation that is anticipated in the future.

Although the analysis of F5/E5 and B3 requirements does not provide a general comparison of the ITSEC with the TCSEC, it does serve to clarify some of the intended similarities and differences in the two documents. As such, the dialogue that ensued cannot but lead to the development of more precise and understandable criteria.

Since its first publication in 1983, there have been at least two broad types of criticism levied at the TCSEC. The first is that parts of it are ambiguous and imprecise. The TCSEC authors freely admit that there are inadequacies in the document. The ITSEC authors have tried to eliminate some of the difficulties of the TCSEC. Many of these points where the authors of the ITSEC have intentionally varied with the written or interpreted TCSEC lead to points where the ITSEC is stronger than the TCSEC. Being human, however, the ITSEC authors in their own writing have introduced ambiguity and imprecision which, ideally, will be clarified in future drafts. This paper has identified both points of intentional variation of the ITSEC from the TCSEC, and points of ambiguity in the current draft of the ITSEC.

A second major criticism of the TCSEC is that its binding of functionality and assurance into a single digraph class is too restrictive. The authors of the ITSEC have chosen to separate functionality and assurance completely, so that for example, evaluation of a high assurance-limited functionality trusted system becomes a possibility. Also, the authors of the ITSEC have decided to broaden its applicability by allowing the evaluation of trusted systems whose policy is other than confidentiality. These goals extend the applicability of the ITSEC beyond the range of trusted systems for which the TCSEC is appropriate. These goals also have the unfortunate side effect of allowing only minimal requirements to be posed for either functionality or assurance. To mandate specific mechanisms would be inappropriate since different policies may require different mechanisms. At low assurance levels, one might be willing to accept modest

functionality, but one would want more stringent functionality requirements as the assurance level rises. Given the absolute separation of features from assurance, it was impossible for the ITSEC authors to impose such progressive requirements. While the ITSEC authors have addressed the excessive restrictiveness in the TCSEC, they have also become susceptible to the problems of generality.

The authors of the ITSEC used different premises and language than the TCSEC and thereby created an evaluation document that is close but not identical to the TCSEC. As has been identified in this analysis, some of the variations between the ITSEC and the TCSEC were intentional, while others were not. A goal of this analysis has been to clarify the differences so that as the authors of the ITSEC refine their criteria, only the intentional differences will remain. However, ignoring the predefined functionality classes (which are in any case only exemplary), the ITSEC represents a catalogue of evaluation criteria, whereas the TCSEC is a mixture of evaluation criteria and security requirements. The ITSEC does not (nor was it intended to) tell anyone what to build, only how to evaluate what has been built.

REFERENCES

[ITS90] Information Technology Security Evaluation Criteria, Draft version 1, May 1990.

[GISA89] German Information Security Agency, IT-Security Criteria, version 1, 1989.

[TCS85] National Computer Security Center, Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, December 1985.

AUDITING OF DISTRIBUTED SYSTEMS

D. Banning, G. Ellingwood, C. Franklin, C. Muckenhirn, D. Price
SPARTA, Inc.
7926 Jones Branch Drive
Suite 900
McLean, VA 22102

Security auditing systems are used to detect and assess unauthorized or abusive system usage. Until recently, security audits have been confined to a single computer system. Current work examines ways of extending auditing to include heterogeneous groups of computers (distributed systems). This paper examines the issues involved in auditing distributed systems, presents the framework for a Distributed Auditing System (DAS), and proposes a design for the audit reporting elements of the DAS.

INTRODUCTION

Security auditing for computer systems is the collection and analysis of computer system usage information used to ascertain the security posture of a computer system. Until recently, auditing has been performed only on a local basis, that is, information collected was logged on the system under audit. While this is a reasonable approach in an environment where there are few hosts that require auditing, as the number of hosts requiring audit increases, it becomes difficult to 1) examine the audit trails, 2) analyze the information and correlate events on one host to events on others, and 3) maintain consistent levels of audit collection. A further complication in large networks is the probable use of a variety of computer systems, each potentially having a different auditing mechanism, reporting syntax, and audit trail.

This paper presents an architecture for the collection of audit data in a distributed environment. One of the goals of this document is to relate the Distributed Audit System (DAS) architecture to the large body of work currently being done in the area of intrusion detection.

We are providing a method for presenting system-independent audit information and transportation of the information for analysis by a security officer or intrusion detection system at a central node in a distributed network. Our approach is expected to complement intrusion detection systems, not to compete with them.

Overview

The primary purpose of this paper is to describe a concept for auditing security-relevant events in a distributed environment. To accomplish this goal we defined the relevant audit issues, outlined the specific goals of auditing, and put our work in perspective with ongoing intrusion detection projects. These are briefly outlined here in order to accomplish the main focus as described above. An in-depth discussion of these issues can be found in the draft report delivered to Lawrence Livermore National Laboratory in September of 1990 and referenced in the bibliography.

The issues relevant to distributed auditing include: what data should be collected, how to transport audit data from a collection point to an analysis point, the system-independent audit data representation, the user interface and user invoked functions and the control of audit functions from a remote location. These are all issues that have been addressed in the concept and design of the DAS architecture as presented in Figure 1.

Other issues that are more appropriate for research by developers of intrusion detection systems include: data storage for subsequent retrieval and damage assessment, formulation of audit records into "security events" and anomaly detection from a set of events. What constitutes a good intrusion detection algorithm for network use is being addressed by projects such as Intrusion Detection Expert System (IDES), Haystack and the Network Security Monitor (NSM).

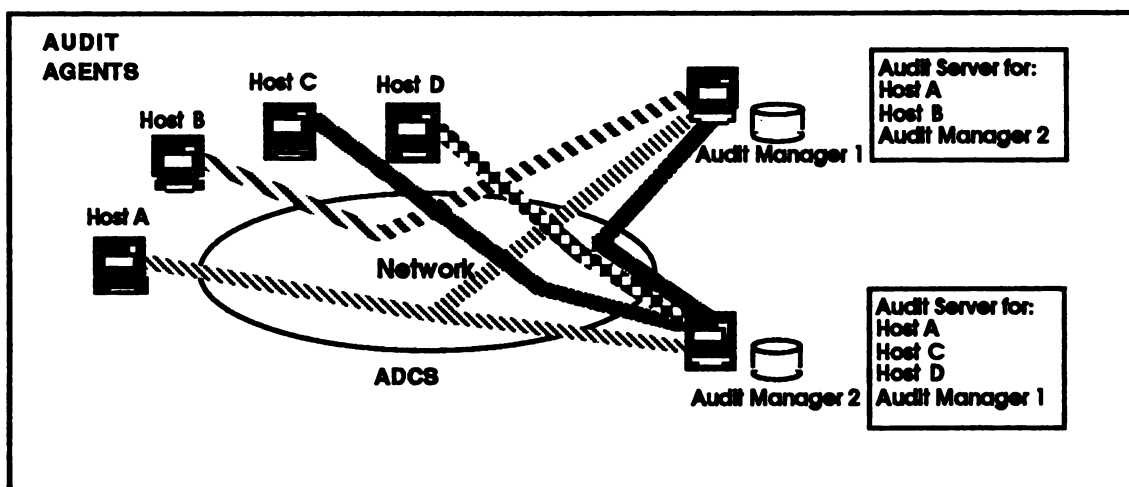


Figure 1 Distributed Auditing System Conceptual Architecture

GOALS OF AUDITING

This section briefly summarizes the goals of auditing and serves to establish the requirements for the design.

Security auditing is a broad function that can include the definition of security events, the creation of audit records, the real time analysis of these records for indications of anomalous activity, the archiving of these records for subsequent analysis, and the postmortem analysis of these archived records for various purposes.

From a security objectives standpoint, one of the more important goals of security auditing is that of providing for individual accountability, such that an individual knows with certainty that he is to be held accountable for his actions. This alone may serve as a major deterrent to abusive behavior.

Other related requirements for audit records are summarized below:

Intrusion Detection: the ability to detect suspicious activity through the use of user profiles

Real Time Monitoring: the ability to monitor activity on a system in order to detect unauthorized activity

Damage Assessment: the ability to determine what was compromised

Attack Reconstruction: the ability to understand how an attack was carried out (i.e., in order to design effective countermeasures to guard against future attacks of the same type)

Damage Recovery: the ability to recover from whatever damage may have occurred

Each application may require an additional set of information that the security auditing system should collect. A good auditing system should address all applications that have a requirement for audit records.

The approach has been to define an overall security auditing architecture with functions and mechanisms for the collection and management of audit-related data, and allowing for the future refinement of these mechanisms to serve advanced requirements such as computer analysis.

DAS CONCEPT EVOLUTION

The history of the DAS began in 1988 with the initial concept of a "virtual audit trail". It has evolved into the current definition of a set of network management protocols to control the collection of audit data

Initial Concept

The original concept defined a standard representation of a canonical audit trail that could be used by the current audit analysis tools. Standard form records collected on multiple machines could be analyzed by a single security monitor for an entire network of systems. This concept evolved into the notion of a "virtual audit trail" and a related set of protocols for transporting data in a common format. In considering complex situations where multiple "audit messages" are needed to compose a single "network virtual audit message", a method was considered where the "translation" would be performed at the application level and the presentation protocol would be used for local data-representation translation.

Different types of transport protocols such as, TCP, UDP and VMTP were considered with respect to the selection of transport reliability, duration of calls and use of network resources. The main difference between the different transport protocols is in how they move data (e.g., as independent blocks of data or as a continuous stream of bytes) and how reliability is achieved.

Evolution of the Concept

An architecture for distributed auditing developed as mechanisms were described for collecting data from multiple host systems in a network for a multitude of purposes (e.g., real time intrusion detection or after the fact analysis resulting in damage assessment). The architecture provides a framework for a set of application/transport level protocols for transmission of auditing data, and a management protocol for controlling the local host (e.g., setting thresholds and synchronizing clocks) from a remote location.

A top level outline of a DAS was developed and documented in a report delivered to LLNL in September 1989. Later, the notion of an auditing protocol was extended to address both the transmission of data from the Audit Agent (AA) to the Audit Manager (AM) and the control the operation of the AA from the AM. The names of these components have evolved to allow association with terms more commonly used in network management.

The belief is that the AM can send commands to the AA (via the Audit Data Communication Service (ADCS)) to increase granularity of monitoring, to audit specific users in detail, to audit accesses to specific files, to audit specific system calls, log all traffic to/from a specific node/terminal, take a snapshot core image, etc. Thus the security officer, sitting at a workstation connected to the AM, can control the auditing throughout the entire network and can respond quickly to newly discovered attacks (e.g., as those reported on the networks by CERT and LLNL's CIAC).

The machine that supports the AM can also have a back-end connection to a system that interprets the audit information for real time detection of anomalous events. An extension is to allow such a system to signal the AM to increase the fidelity of monitoring, etc., much as a human security officer would respond to detected anomalous events. The concept can be further extended to the idea of multiple AMs, where each community of interest can have its own AM, allowing logical subnets for which each AM collects audit data.

Network Management as a Model for Auditing

SPARTA's Networking Research group is heavily involved in the design and development of network management protocols. Struck by the similarity between collecting audit data and collecting performance data, they suggested that we examine the work in the network R&D community that is leading to the definition of network management protocols providing mechanisms for collecting data from various nodes in a network. They observed that, since the mechanisms for controlling the collection process and for reporting it to a central site are similar, the same protocols might be used for both purposes.

A review of the evolving specifications for the upcoming Common Management Information Protocol (CMIP), the related CMI Service (CMIS) specification, and the Management Information Base (MIB) which defines the data elements showed the similarity between collecting data in a network for network management and collecting security-relevant data elements.

A copy of CMOT (CMIP running over TCP) was obtained from the University of Wisconsin and was evaluated on the company's internal LAN to determine whether it could be easily extended to collect the network security data elements.

We concluded that network management protocols provide a good model for our DAS design and the network auditing architecture and design presented in this document is based on the premise of extending the network management protocols currently being defined to incorporate provisions for the collection of security events.

The DAS concept now includes the following:

1. An application for collecting data and transforming it to a network virtual representation. This requires a format and semantic meaning for audit records
2. A transport protocol for transmitting the audit records.
3. A management protocol for dispatching commands from the central site to the remote node that requires a format and semantic meaning for the commands (i.e., to instruct the remote node how to behave upon receipt of each command).

AUDITING AS A NETWORK MANAGEMENT FUNCTION

Network management protocols provide a mechanism for transmitting network performance information from remote nodes to a central collection point. As mentioned earlier, the collection and reporting process for performance data and audit data are very similar. Therefore, the network management protocols can serve as a "model" for collecting, reporting, and transmitting audit information in a distributed network. Below is a brief description of network management protocols and their applicability to audit functions.

Introduction to Network Management

Network management is accomplished by managers at local management stations and agents at remote managed nodes exchanging monitoring and control information via protocols and shared conceptual schema about a network and its components. The shared conceptual schema mentioned above is a priori knowledge about "managed objects" concerning which information is to be exchanged. Managed objects are abstractions of system and networking resources (e.g., a protocol entity, an IP routing table, or in this case, auditing resources) that are subject to management. Managed objects have attributes, operations, and notifications that are visible to managers. The internal functioning of the managed object is not visible to the manager. Currently, an agent is responsible for conversions between a managed system's internal format of managed objects and the external format of managed objects (i.e., the form expected by the manager).

Using management services and protocols, a manager can direct an agent to perform an operation on a managed object for which it is responsible. Such operations might be to return certain values associated with a managed object (i.e., get a variable), to change certain values associated with a managed object (i.e., set a variable), or perform an action, such as self-test, on a managed object. In addition, the agent may also forward to the manager notifications generated asynchronously by managed objects (e.g., send updates periodically).

Network Management Architecture

The network management architecture described here consists of a Management Information Base (MIB) containing a list of managed objects, the International Organization for Standardization (ISO) Common Management Information Services (CMIS)/Common Management Information Protocol (CMIP) Manager and Agents. The Managers and Agents exchange information based on the managed object definitions contained in the MIB, and the ISO network management protocols that facilitate the exchange of this information.

Management Information Base (MIB)

A MIB is a list of managed objects, described in external format, which are considered useful for a particular application. The Internet MIB contains managed objects that are read-only (since current management protocols are not sufficiently secure to exert control, as would be the case with writable objects), and help a manager determine the status of the network elements. Using the Internet MIB as a model, it should be possible to develop an audit MIB.

CMIS/CMIP Manager and Agents

The Common Management Information Services (CMIS) are provided by the Common Management Information Service Element (CMISE). The Common Management Information Protocol (CMIP) supports these services. An invoking CMISE-service-user, or "manager", may invoke a management operation. A performing CMISE-service-user, or "agent", is the process that performs a management operation invoked by a "manager." A CMIS/CMIP manager and agent applications could use adaptations of the ISO Common Management Information Service Element (CMISE) to exchange information and commands for the purpose of auditing.

CMISE provides facilities for a managed "agent" to send multiple linked responses to a manager. An audit agent could use this type of service to send detailed information to an audit manager.

CMISE also provides to managers the ability to "multicast" operations to be performed on a group of managed objects. Through CMISE services, a manager can perform a single operation on a group of managed objects. A distributed audit mechanism could use such a service to assist in responding interactively to network attacks.

4.3 Uses of CMIP in Distributed Auditing

CMIP offers a mechanism to transmit information between agents and managers in a distributed network. The components of the auditing system could use the network communication services offered by CMIP.

AMs located on remote network nodes can send messages to audit applications located on many different local nodes. Audit applications would use the same services to send audit information to the AMs. The advantage of using CMIP for such communication is that a rudimentary mechanism already exists through the CMISE services.

To implement a distributed audit capability using the CMIP protocol for communication, the CMIP protocol would have to be extended. Additions to CMIP would include definition of message types to transmit between manager and agent and specification of what information is expected of both the manager and the agent.

DISTRIBUTED AUDIT SYSTEM DESIGN

The design of the DAS consists of 4 major components, Virtual Audit Trail (VAT), Audit Agent (AA), Audit Manager (AM), and Audit Data Communication Service (ADCS) as depicted in Figure 2. The functions performed by each of these components are discussed below.

Security of the DAS is critical to a successful implementation of audit services. Without the implementation of security principles, a DAS may be attacked and rendered useless in either detecting an attack on other computing resources or in assessing the cause and extent of any resulting loss.

The DAS Architecture incorporates three security principles: access control, data integrity, and assured delivery of messages. In light of this: 1) only specifically authorized individuals (usually a security officer) may change the selection of audited events on a system or cause the audit reporting mechanism/process to stop; 2) audit reports must not be modified while in storage or in transit to storage (over the network); and 3) audit reports that are generated and transmitted to a manager must be received.

Virtual Audit Trail (VAT)

The VAT is formulated from audit information sent from the AA to the AM. A virtual audit record is distinct from what is recorded on a particular host. It is O/S independent and reflects security relevant events and must be inclusive enough to fulfill any of the goals outlined in Section 2. The virtual audit record is unrestricted by what the local site security policy defines as security relevant.

To determine what constitutes such an audit record we should examine several areas:

- 1) look at the auditing done by particular O/Ss, determine the security relevant events and include these in the virtual audit record,

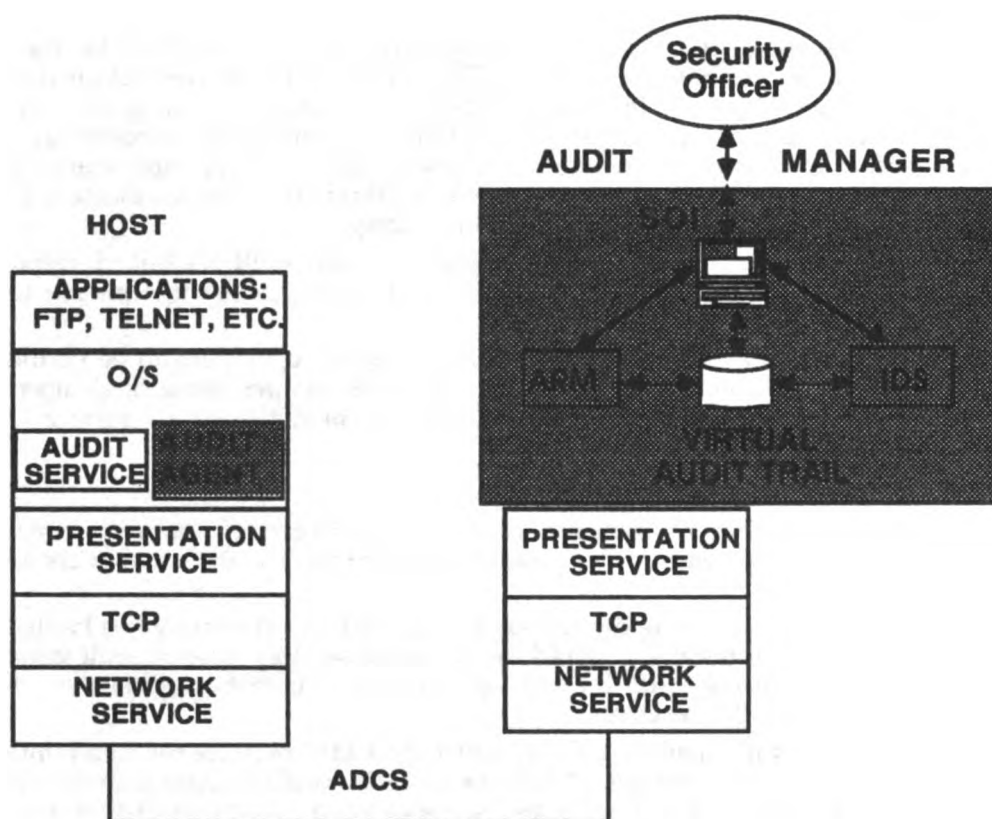


Figure 2. Distributed Audit System Design

- 2) look at the current audit analysis tools and catalogue what events are needed by each of them for their particular analysis, and
- 3) look at what events have triggered discovery of incidents in a real situation (e.g., Cliff Stoll's incident, etc.).

Using the above information, a set of record types that represent different types of events can be defined. For each type of record, the variables that define that event are determined. These audit records constitute the VAT at the AM.

Once the contents of the audit records have been defined, a MIB of audited elements is specified for use with the network management protocols. An audit MIB contains managed objects considered essential for auditing.

Audit Agent (AA)

The AA consists of a process running on each network host and has three principle functions: selecting audit events for forwarding to the AM, translating host-specific information into a "virtual" format, and responding to commands from the AM.

The AA sends selected audit information from the host to the remote AM. In a distributed network, each host would have an AA and would report to a number of AMs. The AA examines the audit records generated by the host's O/S and determines what information to forward by examining an audit table.

Forwarding Audit Events

This audit table, depicted in Figure 3 is maintained and updated in response to commands from the AM. The use of the audit table allows each site to send audit reports based upon the site's individual security policy.

The audit table tells the AA which events to send as event reports, which to send as event summaries, and which to ignore. An event report is a detailed record containing information such as the userid, command invoked, network address, and any related fields specific to a particular event. An event summary reports the frequency and number of a particular event per some unit time. The event summary could be useful in a real-time situation where limited specific information is needed quickly (e.g., when an intrusion is suspected and more information is needed).

The audit table is read by the AA upon initialization. The audit table has the structure of username, event, report, and summary. The username field indicates which user's activities are to be audited. The event field indicates what event to audit. The report field is a boolean value that indicates if an event report is to be sent to a Audit Manager. The summary field is also a boolean value that indicates if an event summary is to be sent to the Audit Manager. Usually, the event, and report fields are mutually exclusive, i.e., you either send an event report or an event summary but not both. Finally the AM field indicates to which audit manager(s) this event should be reported.

Translating Host-Specific Information

The AA will use a language tailored to each O/S to perform translation of host-specific information to a "virtual" format. The language will consist of a set of verbs and nouns which express all the audit events to be used in the DAS. It is expected that this language will be extensive in order to express all the required information with the desired level of granularity.

Using this approach, logon reports could be as simple as "Joe logged on at 1:30" or as complex as "Sam, aliased to Joe, logged onto host Euler from host Kepler, whose internet address is 192.48.111.1, via the Internet gateway 192.5.8.1, on 26 June 1989 at 1:30 pm." Each of these reports is optimal for the information they contain. Each report relays all the information available from their respective O/Ss without loss or overhead.

Responding to Commands from the AM

The audit information collected by a particular AA is determined by local security policy. What subset of this information is sent to the AM is predetermined by the AA's audit table. Though this information would be periodically updated by the AM, it would be useful to have the ability to request further information from the AA.

The DAS provides the AM the capability of controlling the operation of the AA through a series of commands sent via the ADCS. These commands allow the AM to request increased granularity of audit information on specific: users, files, system calls, resources, and node/terminal traffic. Upon receipt of the commands the AA processes them, performs the necessary action and provides a response. If the necessary action cannot be performed (e.g., user has logged off and no further information can be obtained), a response indicating the inability to complete the task is formulated.

Audit Manager (AM)

The AM consists of three components: Audit Record Manager (ARM), Security Officer Interface (SOI) and the Intrusion Detection System (IDS). The AM acts as a centralized control center for audit information transmitted from distributed hosts. The three components of the AM work closely together to provide these services: collection/correlation of audit information, interpretation of audit information, and notification of the AA to take further action. Figure 4 shows the logical interrelationships between the AM components.

Audit Record Manager (ARM)

Upon receiving audit records from the AA, the ARM updates the audit database with the new information. Some maintenance functions are provided automatically (e.g., archiving and deletion of duplicate entries). Other functions are provided through a set of security officer queries entered

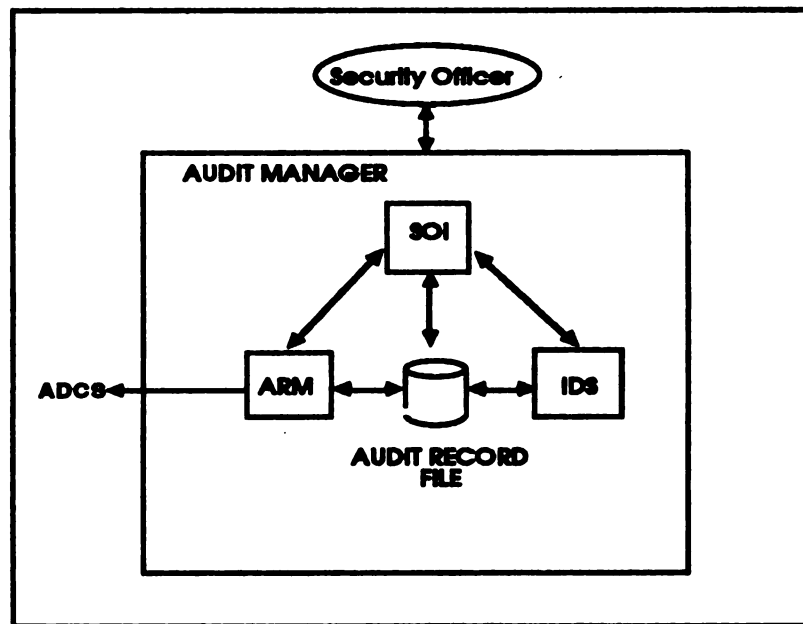


Figure 4. Audit Manager Architecture

through the SOI (e.g.,) deletion by record, correlation of audit entries and record retrieval. The ARM initiates transmission of audit table updates at specified time intervals. The ARM also has the capability of sending audit table updates upon instruction from the SOI.

A primary purpose of the audit database is to provide the necessary information for the IDS for identifying suspicious activity. Querying of the audit database can be done through the IDS or via the SOI and controlled by the ARM.

The ARM also provides correlation of incoming audit information from different hosts. This correlated information is then given to the IDS for analysis. Correlation of information is important for those networks where the same user utilizes different hosts such that a complete set of audit information can be given to the IDS for analysis.

Security Officer Interface (SOI)

The SOI provides an information display and command processing capability. The SOI display will use a window structure to provide graphical display of detected anomalies, security of the network and status of AM functions. A command capability will be provided for issuing commands to the AA for additional audit information. A menu of frequently used commands will be provided as well as a command line option.

Intrusion Detection System (IDS)

The IDS to be used with the Distributed Audit System is not specified in this design, but treated as a "black box" that uses the audit records maintained by the ARM to detect suspicious activity. The IDS used for this function can be any of the current systems available. The configuration or function of the IDS is independent of its use for this DAS design.

The DAS will provide audit records to the IDS for analysis of user activity. The security officer will then be able to send a command to the AA requesting an additional granularity of information on a particular user. For example, if the AM receives an event summary that user Joe has used the telnet command 50 times in the past hour and this activity is outside of Joe's user profile (according to the IDS), the AM can send a message to the AA asking to see all the commands

issued by Joe. When the AA receives this request, it would modify the audit table to reflect the request to monitor Joe more closely.

However, if an IDS is used that does not perform real-time monitoring, the additional information available will be limited to that already in the AA audit trail since the user will most likely not be active.

Audit Data Communication Service (ADCS)

The ADCS provides the necessary communication services for transporting messages between AA and AM. To enable an AM to control the functions of an AA, services currently defined by CMIS could be adapted for use in the ADCS. Using the network management services provided by the ADCS, the AM could request the AA to provide additional audit records on a particular user or event, change the events being audited, set/reset audit thresholds, and provide event reporting at specified intervals.

The automatic reporting of audit events to the AM from the AA could be accomplished using the M-EVENT-REPORT service which is invoked by the AA at specified intervals.

Using the CMIS management services and CMIP protocol, the manager can direct the agent to perform an operation on a managed object for which it is responsible. The following services would be invoked by the AM to make requests of an AA:

M-GET: Used to request additional audit records from the AA for increased granularity from existing audit records.

M-SET: Used for setting/resetting AA audit thresholds from the AM.

M-ACTION: Used to increase collection of data by modifying an existing parameter (e.g., change the system files to be audited).

M-CREATE: Used to request an AA to audit new events for a particular user.

M-DELETE: Used to request AA delete audit records, audit events or an audited user due to changes in operation.

The ADCS must also provide the security services of data confidentiality, data integrity during transmission and assured service of messages.

OTHER ISSUES AND FUTURE DIRECTIONS

As indicated in the overview section, many issues were considered in the DAS design and not all of them can be thoroughly discussed here. To fully define the DAS design, it is necessary to resolve some additional audit issues that are currently being researched. These include, but are not limited to the security and technology issues outlined briefly below.

Security issues related to the building of a DAS include:

- Assurance - both in the case of being assured that the AA is performing as it should and in the case of being assured that the AM is secure from penetration;
- Transmission security - the information flow from the AAs to the AMs and vice versa must be secure; and
- Network Management Protocol Security - while work is ongoing in this area, the idea is still fairly new.

Technology issues facing the successful implementation of a DAS include:

- Commercial Marketability;
- Anomaly Detection Capability - the testing of; and
- Time Stamping - addressing the delays related with heterogeneous hosts.

All of these issues can be addressed via prototyping, which is the next step in the process.

BIBLIOGRAPHY

- [1] DOD5200.28-STD, DOD Trusted Computer System Evaluation Criteria (TCSEC), December 1985.
- [2] International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 9595, CMIS, 6 December 1989.
- [3] International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 9596, CMIP, 6 December 1989.
- [4] Internet Request For Comments (RFC) 1156, MIB, May 1990.
- [5] Internet RFC 1155, Structure and Identification of Management Information for TCP/IP-based Internet (SMI), May 1990.
- [6] Internet RFC 1095, The Common Management Information Services and Protocol over TCP/IP (CMOT), April 1989.
- [7] NCSC-TG-001, Version 2, A Guide to Understanding Audit in Trusted Systems, 1 June 1988.
- [8] NCSC-TG-005, Version 1, Trusted Network Interpretation, 31 July 1987.
- [9] SPD1003.6, Draft 2, Security Interface for the Portable Operating System Interface for Computer Environments, 4 June 1989.
- [10] Auditing of Distributed Systems (Draft), SPARTA, Inc., 28 September 1990
- [11] Survey of Audit Trails and Audit Analysis Tools, SPARTA, Inc., 3 March 1989

BUILDING A MULTI-LEVEL APPLICATION
ON AN UNTRUSTED DBMS
IN A UNIX SYSTEM V/MLS ENVIRONMENT
- A PROJECT'S EXPERIENCE

David S. Crawford
Directorate of Security Operations
Department of National Defence
National Defence Headquarters
101 Colonel By Drive
Ottawa, Ontario, Canada K1A 0K2
(613) 993-6775

ABSTRACT The procurement option of using an untrusted DBMS on a TCB where both trusted system and DBMS functionality is required is briefly discussed in Appendix B of the Trusted Database Interpretation. This paper discusses an approach proposed for a Canadian Department of National Defence project to design and implement several multilevel multiuser DBMS-based applications using an untrusted DBMS on a B1 UNIX® TCB, and the design and operational constraints imposed by this solution.

PART I - INTRODUCTION

1.0 BACKGROUND

Adequate segregation of sensitive information has historically been a serious impediment to the provision of Information Technology services in support of defence activities. One Canadian Department of National Defence project had concerns about the ability to provide a secure environment for applications and data on Base level computer systems due to presence of both UNCLASSIFIED and CONFIDENTIAL information. Data analysis had determined that information processed on these applications was, in certain instances, classified in isolation and in aggregation. In addition, the number of sites involved resulted in significant cost implications if all equipment at all sites was required to meet TEMPEST standards, since current Canadian standards require TEMPEST protection for any classified processing.

These concerns led to the project to plan to operate in a Controlled (restricted form of multilevel) Security Mode of Operation and the statement of a requirement for a B1 Trusted Computer Base, which was subsequently specified as AT&T UNIX® System V/MLS (SV/MLS)¹. By specifying a B Division TCB, the project intended to address confidentiality concerns and to minimize the number of TEMPEST equipment required, since device labelling could be used to restrict classified processing to only the limited number of TEMPEST devices attached to the TEMPEST host computer. Other integrity concerns would be addressed by traditional software engineering practices.

The other early concern for the project was establishing the application software environment. Procurement of a

¹ UNIX is a registered trade mark of AT&T

DBMS and 4GL environment was initiated and resulted in the procurement of ZIM^{®2}, a DBMS and 4GL from Sterling Software, for this project prior to determination of the TCB requirements.

The framework of untrusted DBMS and secure UNIX was established without considering whether or not the DBMS could be effectively used on a multilevel operating system and how the DBMS based applications could be designed. It now remained to determine how to design and implement DBMS based applications that would meet security requirements without violating the TCB.

This paper discusses major design issues necessary to build multilevel applications within the project constraints and additional considerations employed to provide additional protection.

2.0 PROJECT FRAMEWORK AND CONSTRAINTS

The nature of a multilevel application is that it more closely models an actual defence-related environment, where information exists at various levels of sensitivity. More traditional data processing approaches, such as operating separate systems for various levels of sensitive information or treating all information at the highest level of sensitivity held, are expensive both in terms of capital procurement costs and administrative overhead. From the project perspective, operating with UNCLASSIFIED and CONFIDENTIAL information would require all project equipment to meet TEMPEST requirements unless an acceptable multilevel solution could be implemented.

The project, as part of the requirements definition, had conducted extensive data modelling. Analysis of the information model from a security perspective established that any tuple, in isolation, was UNCLASSIFIED. However, specific tables were identified that were, in the aggregate, CONFIDENTIAL. These tables were relatively static and managed in isolation by a central authority.

In addition, specific joined tables, in the aggregate, were CONFIDENTIAL. Project personnel were able to identify specific views, application screens and reports that contained classified information.

One area of considerable concern dealing with aggregation concerned the quantity at which the aggregate became classified. The classic example on the project was the aggregation of persons, where an individual tuple was UNCLASSIFIED and all persons belonging to a unit reveal operational capability and thus was CONFIDENTIAL. The project solution was to establish an overly restrictive de-facto business rule that any set containing more than one tuple was classified.

Project applications would be developed and maintained by a central authority. Each application would be released to sites as a turnkey application or subsequently as an update to an application. No capability to modify the applications was to be provided to the field.

PART II - DESIGN FRAMEWORK

The specific conditions within the project and the features available in the TCB and the DBMS led to the formulation of two general problems and the associated approaches in building multilevel applications that relied on TCB controls and the identification of additional controls that would compensate for acknowledged weaknesses.

² ZIM is a registered trade mark of Sterling Software

3.0 SECURING DBMS TABLES - A FILE BASED APPROACH

The first general problem was controlling access to any data within a given DBMS table and was based on the existence of tables that contained CONFIDENTIAL information. The general approach taken to address this problem was based on the use of the TCB mandatory access controls to control access to DBMS tables. This approach involved the labelling of the O/S files containing the DBMS tables according to the highest level of sensitivity of the DBMS table, thus controlling access to data through TCB controls.

The extensive data analysis on the project supported this approach since it was readily apparent that tuples within tables could be assumed to be of a uniform sensitivity and table level sensitivity labelling would be sufficient. This approach would have not been appropriate had tuples within tables been required to reflect differing levels of sensitivity.

This approach was technically possible in the target environment since the ZIM DBMS managed each table as a separate O/S file. The DBMS only opened those tables required and opened tables as READ-ONLY unless the table was being updated. Errors in opening tables for WRITE access, such as are caused by opening files labelled at a lower level, resulted in the SELECT operation returning a null set and a warning message issued by the DBMS.

One concern with this approach is that it may impose significant restrictions on functionality if update activities spanning classification levels are necessary. In the case of this project, most tables were UNCLASSIFIED. The few CONFIDENTIAL tables were relatively static, were not closely linked to its related UNCLASSIFIED tables and could be maintained independent of the UNCLASSIFIED tables.

4.0 SECURING DBMS VIEWS - A PROCESS BASED APPROACH

The second general problem was controlling access to CONFIDENTIAL views of data that was UNCLASSIFIED in isolation. The approach to address this problem was made somewhat obtuse since the ZIM DBMS did not directly support a view mechanism. However the view mechanism was represented through each Selection and Projection operation in each ZIM program.

A means to address this problem was needed. A view was represented as the retrieval statement, such as a SELECT statement, within a program. Each ZIM program existed as an O/S file and the DBMS required READ access to the file in order to execute the program. By labelling each program with a sensitivity label corresponding to the highest level of sensitivity of the views or aggregations being manipulated, the TCB was employed to control access to views and aggregations. Access would be based on the sensitivity label of the user's process that invoked the program, hence the term "process based" control. This meant that users operating at a level dominating the program label could execute the program whereas users operating at a lower level would be unable to execute the program. Based on this approach, it was accepted that labelling the means of producing views or aggregations would represent a comparable functionality to labelling views.

This approach was supported by earlier work on the project to define screen and report formats and contents. This work had included review for security relevant issues, such as display of classified information.

An additional refinement to this approach sought to employ mandatory controls to enforce some integrity issues by using confidentiality labels as *de-facto* integrity labels. The labelling scheme was modified so that application programs would be labelled at a <level - 1> in order to isolate the programs from the user processes. In the project example, UNCLASSIFIED was established as level 30 and CONFIDENTIAL was level 180. Level 29 was created for UNCLASSIFIED programs and level 179 was created for CONFIDENTIAL programs.

<u>Level Name</u>	<u>Level Number (numeric level)</u>	<u>Suffix</u>	<u>Prompt</u>	<u>Hardcopy</u>
Secret	210	(S)	SECRET	SECRET
Application S	209	(Appl S)	Appl (S)	Appl (S)
Confidential	180	(C)	CONFIDENTIAL	CONFIDENTIAL
Application C	179	(Appl C)	APPL (C)	APPL (C)
Protected A	60	(PA)	PROTECTED A	PROTECTED A
Application PA	59	(Appl PA)	APPL (PA)	APPL (PA)
Unclassified	30	(U)	UNCLASSIFIED	UNCLASSIFIED
Application U	29	(Appl U)	APPL (U)	APPL (U)
System	0	(TCB)	SYSTEM	SYSTEM

Figure 1: Project Labelling Hierarchy

5.0 ADDITIONAL CONTROLS

In considering a process based approach to managing access to data, the software engineer must consider both controlled access and uncontrolled access to sensitive information. Controlled access to information is the access that a user has through the application functionality and is a direct result of system design and implementation. This type of access is defined in terms of application screens, reports and query facilities. Uncontrolled access is the access that a user may have if free to specify how and what to retrieve. This type of access is typified by the use of ad-hoc query languages or through the use of other software, such as system utilities. In addition, access to information also includes device level considerations as there must be mechanisms in place to ensure that classified information is routed to the appropriate devices and labelled appropriately.

Uncontrolled access to information poses the most immediate threat in the use of a process based approach to building a multilevel application. This is primarily due to the potential for uncontrolled aggregations permitted through ad-hoc query facilities. The ability of a user to extemporaneously, repetitively and interactively define and retrieve any possible combination or permutation of data existing on a system poses a horrendous burden of proof on the software engineer that all possible data retrievals will be at the same level of sensitivity as the base data. In the case of this project, it was already known that some aggregations of data were CONFIDENTIAL, even though these combinations are based on data which is UNCLASSIFIED in isolation. This implies that access to the ad-hoc query capability, if permitted, be restricted to known users with the appropriate clearances and permissions, to users operating at the appropriate security level and to TEMPEST devices, if classified aggregates are possible.

There are two aspects to the ad-hoc query threat. There is the possible surreptitious access to underlying query capability. This is represented by users who circumvent controls and use software they are otherwise unauthorized to use. The second and more plausible threat is that of legitimate access to an ad-hoc query capability. Since the designer cannot control what the end user specifies as retrieval criteria, there are legitimate concerns that users could intentionally or inadvertently retrieve sensitive aggregations while operating at inappropriate security levels, while operating without the appropriate clearances or while using inappropriate (or non-TEMPEST) devices.

The requirement to permit ad-hoc query can be very real for the applications designer as it will add considerable functionality in terms of addressing unforeseen information requirements and may significantly reduce the number of report generators required to be developed. The problem of controlling the contents of a query can only realistically

be addressed by application software to pre-screen each query, a daunting software development task. However if the designer cannot control the contents of the query, he can control access to the means to query by denying unauthorized users and devices access to the query engine.

The problem of actual or potential access to an ad-hoc query facility, if sensitive aggregations can exist, will require that all applications be executed on a runtime engine. This requirement is necessary since it is imperative to guarantee that inappropriate end users or devices cannot, either intentionally or inadvertently, access any ad-hoc query facility. This assurance cannot be provided by application software. However, the operating system, since it is a TCB, can provide this assurance.

The use of the TCB mechanism of mandatory access controls can be employed to permit selective access to the ad-hoc query facility. The design of the ZIM engine assisted in that it did not use a client/server architecture but was separately invoked by each process executing the file. It was therefore possible to restrict access to the ad-hoc query facility by labelling the query runtime engine (executable file) at the CONFIDENTIAL level, which will make it inaccessible to users operating at levels lower than CONFIDENTIAL. Provided that the UNCLASSIFIED components of an application use the ZIM runtime engine, it was then possible to provide the functionality of ad-hoc query for users operating at a CONFIDENTIAL level without compromising access to the means to create classified aggregations.

One problem with this approach was that there were several smaller sites where more than one application would be hosted on the same CPU. In order to address this problem, the use of mandatory access controls in the form of application specific categories was established to enforce mandatory need to know separation of incompatible communities of interest. Since an application that does not hold information which is sensitive in the aggregate should not have restrictions placed on access to an ad-hoc query capability, such applications co-resident with a second application holding information which is sensitive in the aggregate could employ mandatory access controls in the form of application specific categories to differentiate between applications. Separate copies of the query runtime engine, each labelled with the appropriate security level and application specific category, would exist on the system. Users belonging to the second application would not be able to access the query runtime engine labelled for the first application and would, provided that they are restricted to a runtime engine, be unable to gain access to an ad-hoc query capability.

The requirement to ensure all classified or potentially classified information is routed to TEMPEST devices can be effectively addressed if access to classified aggregations is restricted to users operating at an appropriate classified level. SV/MLS supported device labelling whereby minimum and maximum clearance levels are assigned to devices, such as terminals and printers, by the system or security administrator. Labelling all non-TEMPEST terminals and printers with a maximum level of UNCLASSIFIED and all TEMPEST devices with the maximum level of CONFIDENTIAL provided assurances that potentially sensitive information could only be displayed or printed at appropriate devices.

The need to label screens with appropriate sensitivity labels was identified as a requirement. Label processing on SV/MLS required privileged system calls. The DBMS had a feature that enabled reading and writing to UNIX pipes. This feature enabled a very small, simple piece of untrusted application code to be developed to read the *stdout* output from the SV/MLS *labels -u* command, a trusted program that was part of the TCB, and extract the sensitivity label of the process knowing that the label was correct. Actual screen labelling was not trusted but was to be considered part of the normal software activity and subject to independent verification and validation.

Additional controls that were felt to be required included removing all access to the operating system interface ("prompt"). All project applications would deny access to the O/S prompt through the development of application specific menus installed as default shell. In addition, the removal, imposition of restrictive labelling or restricted file permissions on O/S shells and utilities would be carefully considered prior to system implementation. This was not

seen as being detrimental to the project since systems were to be considered turnkey application specific systems and not general purpose ADP equipment.

PART III - DBMS CONSIDERATIONS

A number of DBMS related issues were encountered in building a prototype multilevel application on ZIM under SV/MLS. There are a number of areas where the SV/MLS environment impacts the use of ZIM and the application design. These areas do not, in general, represent problems which cannot be addressed but have approaches that solve, avoid or work around the difficulties.

One SV/MLS feature that proved key to the ability to implement an untrusted DBMS on a secure UNIX platform was secured, or multilevel directories. This feature was developed to address the problem caused by the widespread use of common directories with global read/write access, such as */tmp*, in UNIX. This feature enables the user to reference the same directory from more than one level, while the operating system transparently redirects the user into an appropriate subdirectory for the user's privilege (security) level. Untrusted subjects can reference the same directory and be transparently redirected to a directory which is appropriate for the subject's privilege. Trusted subjects, on the other hand, are not subject to this redirection and the entire directory structure is both visible and accessible [1].

6.0 DBMS WORK FILES

ZIM uses several working files (*zimsetd*, *zimsett*) on a per user session basis and which require READ/WRITE access. The SV/MLS environment impacts this DBMS requirement in that each user will require read/write access to their ZIM working files at all times and ZIM creates and maintains these files in the defined work directory. The immediate problem is that these files will exist at the same security level as the process creating them, which poses a problem in the case of an application requiring two or more security levels.

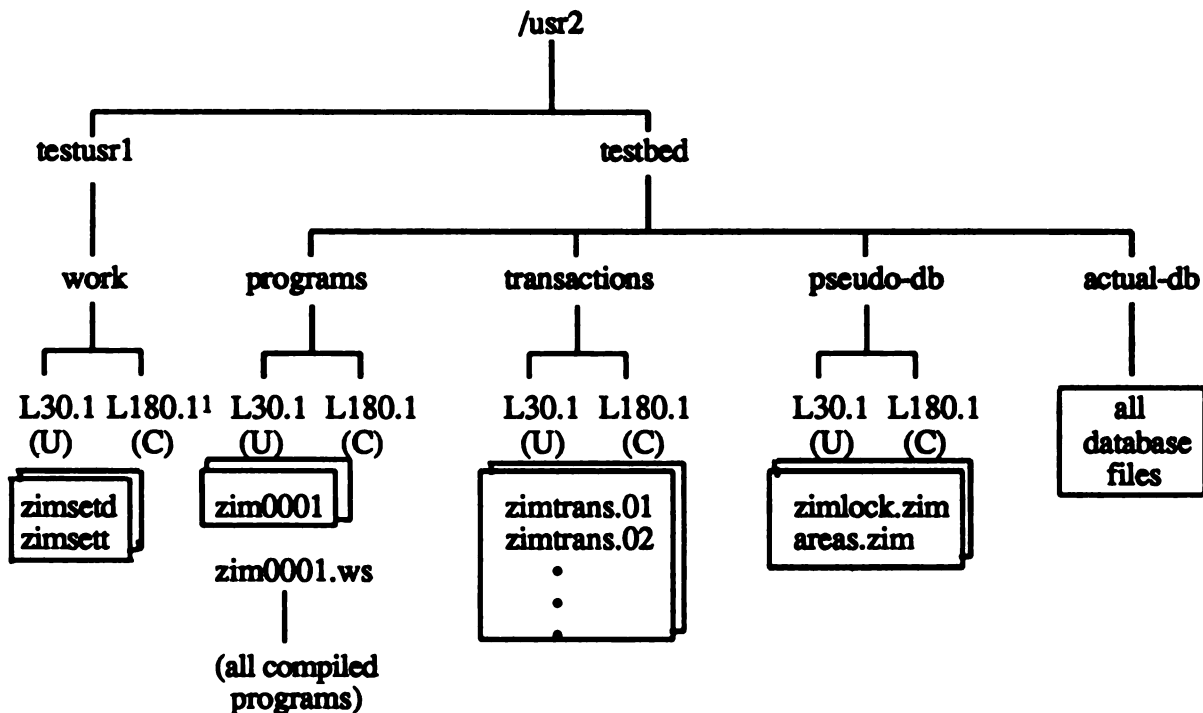
It was possible to set the working directory to a specific directory through the "*work path <pathname>*" entry in the ZIM configuration file (*config.zim*). By creating the working directory as a multilevel directory, a user account could be set up that would permit multilevel use of the DBMS.

7.0 DATA DICTIONARY

The ZIM data dictionary points to the location of all interpreted ZIM program files and all compiled ZIM programs are located in the *zim0001.ws* directory. ZIM, as of Release 3.03, required read/write access to the ZIM data dictionary (*zim0001*). This was subsequently modified to READ/ONLY access as a result of the DBMS port to the UNIX System V/MLS platform. The DBMS data dictionary did not now pose a problem in a SV/MLS environment since this file was now accessible to any process existing at a dominating level.

8.0 TRANSACTION FILES

ZIM used, in support of database journalling, a pool of transaction files that are used by all users of a database. Since these files are reused by all users of the database, the label associated with all journal files must be identical with that of all users. If users operating at a range of more than one label access the database, the DBMS will fail to function since journal files may not have identical labels. The location of zim transaction files (*zimtrans.n*) posed a possible problem as these files normally reside in the database directory. However, the transaction files can be redirected to another directory through the "*audit path <pathname>*" entry in the *config.zim* file. Once again, the creative use of multi-level directories addressed this problem. By creating the transaction journalling directory as a multilevel directory, a user account could be set up that would permit multilevel use of the DBMS.



Note 1: This represents a multi-level directory to support 2 labels. These labels would be an UNCLASSIFIED label (L30.1) and a CONFIDENTIAL label (L180.1)

Home directory: /usr2/testusr1

config.zim: database path /usr2/testbed/pseudo-db
 work path /usr2/testusr1/work
 audit path /usr2/testbed/transactions
 audit updates yes
 .
 .
 .

areas.zim: 0001 /usr2/testbed/actual-db
 0002 /usr2/testbed/actual-db
 .
 .
 .

Figure 2: Sample Directory Structure

9.0 CONCURRENCY CONTROLS - THE MULTIUSER LOCK FILE

A review of multi-user ZIM under UNIX System V indicated that the concurrency control mechanism would be a problem under SV/MLS. This is a multi-user locking scheme based on all processes having read/write access to the *zimlock.zim* file, which is resident in the database directory. A suitable mechanism within the DBMS was needed but this was not a problem which could be addressed within the scope of the project.

The problem associated with multi-user ZIM and the *zimlock.zim* file could be addressed, in terms of a "work-around", through the use of a multilevel database directory. This directory would contain subdirectories for each level associated with the application and each subdirectory would contain a separate copy of the lock file. A ZIM configuration file, the *areas.zim* file, can be used to point to specific directories for specific tables on a table by table basis. This file would be used in this scenario to point to each actual ZIM database table file, which would be located in conventional directories.

The issue of a lack of a guaranteed rereadability was tested. The only problem that was encountered was when the following scenario occurred:

- a. a set was selected by a CONFIDENTIAL process;
- b. an UNCLASSIFIED process updated a table that was part of the set selected by the CONFIDENTIAL process; and
- c. the CONFIDENTIAL process attempted to process the previously selected set.

The ZIM DBMS issued several error messages related to pointer and read errors to the CONFIDENTIAL process since file pointers in the temporary working file were invalid. These error messages could be trapped in the application program and the program could be reexecuted.

This approach is not a solution as it will not guarantee rereadability and will not ensure integrity across security levels since separate copies of the lock file exist for each level. The ideal solution to this problem would be the procurement of a true secure DBMS but this course of action would require new procurement actions and would cause significant project delays. It was accepted by project management that the loss of guaranteed rereadability for processes reading tables from lower sensitivity levels was an acceptable loss of functionality, given the predominantly read-only nature of the of the classified components of the applications within the project.

10.0 ZIM PROGRAMS

There are two aspects to the manner in which ZIM uses programs which assist in the building of a multilevel application. The first, the labelling of specific program files, has already been discussed. In the context of the ZIM DBMS, the inability of the DBMS to read a program will result in a warning message, which can be disable, and continued processing.

The second aspect is the possibility of building separate applications based on the sensitivity level of a given user. The ZIM data dictionary points to the location of all interpreted ZIM program files. All compiled ZIM programs are located in the *zim0001.ws* directory. It is possible to put the application programs in a multilevel directory so that a complete application is present for each level of sensitivity of the application. To the ordinary user, there will appear to be only one program directory. The filenames referenced in the ZIM data dictionary will, if they refer to multilevel directories, be interpreted by the operating system to point to the appropriate directory for the user's current security level. Document filenames, defined as absolute path references, will always point to the appropriate directory since the ZIM data dictionary will reference the appropriate multilevel directory under SV/MLS.

PART IV - CONCLUSIONS

In conclusion, this paper outlines, in terms of a specific project, how multilevel multiuser applications can be developed for an untrusted DBMS on a TCB and use the controls implicit in the TCB. The use of features implicit in UNIX SV/MLS can assist in the use of an untrusted DBMS. The specific case of the ZIM DBMS and its constraints, within the operational context of a project, demonstrate a specific means of implementing a multiuser multilevel application using untrusted DBMS on a TCB.

REFERENCES

- [1] "System V/MLS 1.1.1 Trusted Facility Manual", AT&T, 13 June 1989.**

BUILDING A MULTI-LEVEL SECURE TCP/IP

Deborah A. Fletcher
The Wollongong Group
2010 Corporate Ridge Dr
Suite 550, McLean, VA 22102

Ron L. Sharp
AT&T Bell Laboratories
Rm 14E-214, 1 Whippany Rd
Whippany, NJ 07981

Brian K. Yasaki
The Wollongong Group
2010 Corporate Ridge Dr
Suite 550, McLean, VA 22102

ABSTRACT

This paper describes changes made to a networking protocol in order to make it "*trusted*" in a multi-level secure operating system. The protocols are the standards used by the Internet; the Transmission Control Protocol and the Internet Protocol (TCP/IP). These protocols are currently used in many heterogeneous networking environments. This paper is based on actual work being done by AT&T Bell Laboratories and The Wollongong Group in the joint design and development of a secure TCP/IP.

INTRODUCTION

The Transmission Control Protocol (TCP) and the Internet Protocol (IP) were originally developed for the ARPANET. Together they comprise one of the most popular transport and network layer protocol suites in use today, particularly within the U.S. Department of Defense (DOD). Since TCP is always run on top of IP the two are commonly referred to as TCP/IP. Initially TCP/IP provided no security services except for reliable delivery and integrity checksums. A sensitivity label was added as a possible option in the IP datagram header to enhance security. Since Multi-Level Secure (MLS) systems and networks are just now becoming available, most implementations of TCP/IP do not include this IP option.

Just adding an IP security label to each IP datagram does not provide enough security information for an MLS system. Many conditions must be met when importing information into an MLS system. Is the data labeled? Can the label be trusted to be correct? Is the host authorized to handle the level of sensitivity represented by the label? These questions and others must be answered prior to bringing networking data (i.e., IP datagrams) into an MLS host or passing it on to another network.

AT&T Bell Laboratories and Wollongong have teamed up to develop a security enhanced TCP/IP. This new TCP/IP, referred to as MLS/TCP, is fully compatible with existing TCP/IP implementations. Additional features have been added to provide network labeling and other security services in concert with System V/MLS.^[1] System V/MLS is a multi-level secure enhancement to AT&T's System V UNIX® operating system. System V/MLS received a B1 rating from the National Computer Security Center in September 1989.

In addressing the problem of how to add security to a TCP/IP protocol stack, we were concerned with three non-security requirements. The first was that the specifications for the networking protocols could not be modified. This would ensure that the multi-level host would still be interoperable with all the other TCP/IP implementations. Second was that the MLS/TCP host should be able to remain trusted in an environment where both non-secure and multi-level secure hosts were part of the network. This would provide a transition path from a partially secure network (mixture of trusted and non-trusted hosts) to a completely multi-level secure network. The third requirement was that we wanted current applications to be reused without any changes (i.e., be binary

compatible). This would allow *"commercial off the shelf"* (COTS) software to still be used. This requirement was later limited to those applications that did not require *"root"* privileges.¹ Since *"root"* privilege implies trust, we did not believe that having to modify a trusted application to recognize the security policy was excessive.

This paper provides some of the insights gained and lessons learned while enhancing TCP/IP to work in an MLS environment. Enhancements to the TCP/IP implementation are described. Two types of IP labels are supported and discussed in the Packet Labeling section. Changes to the route selection mechanism are also discussed. A decision was made to support trusted and untrusted application level servers and the impact to these servers is shown. The Network Interface section discusses the changes required to interface to trusted and untrusted networks. As stated earlier, some changes were required to support trusted applications. A section is included which describes some of these changes. Finally, the auditing requirements for a multi-level secure TCP/IP are reviewed.

MLS REQUIREMENTS

Introducing TCP/IP into an MLS environment places additional requirements on the implementation. Modifications are needed to provide the additional security features required to protect the data from compromise or corruption. In addition, a careful examination of the TCP/IP software must be performed to ensure that it meets the assurance requirements for an MLS system.

One of the most important requirements is the added trust that is required. Most TCP/IPs are implemented in the kernel² and thus have access to all of the kernel data structures. A malicious implementation of TCP/IP could violate the security policy by manipulating critical operating system data. Of course this threat is not unique to MLS hosts or even to UNIX hosts. Untrusted software in an operating system can render any security control useless; however, on an MLS host the potential damage posed by such a threat is even greater.

All data in an MLS system must be labeled. Without a label the host can not make access control decisions. There must be a strong link between the data and its associated label. The Trusted Network Interpretation^[2] ("Red Book") has the following requirement concerning network labeling:

"When the TCB exports or imports an object over a multilevel communications channel, the protocol used on that channel shall provide for the unambiguous pairing between the sensitivity labels and the associated information that is sent or received."

There is no one standard format for a sensitivity label. In addition, there are many different representations of the fields within a label. Therefore a robust implementation of an MLS TCP/IP must understand and be able to map between these multiple formats and representations.

Most implementations of TCP/IP do not handle labels. They are used on single-level networks where there is no need for labeling. Backward compatibility requirements dictate that the MLS host should be able to connect to such a single level network, accept data and associate the proper label with this data.

Networks connected to an MLS host may be accredited to handle multiple labels or only one label. The TCP/IP must ensure that no data is sent to a network that is not authorized for that data. In addition, all incoming data must be within the sensitivity range authorized for the host.

As with any protocol, TCP/IP buffers data until the receiving host can receive it or until the user is ready to read it. It is critical that the MLS TCP/IP maintain strict separation of this data inside the kernel allowing no accidental mixing of data of two different sensitivities.

All security relevant events must be audited. This includes successful and failed connections as well as any change in security parameters. Since the operating system may never see a failed connection, such auditing must be performed within TCP/IP.

1. The concept of *"root"* privilege in the UNIX environment means that the process has the capability to bypass most security checks.

2. The kernel is the part of the UNIX operating system that is separated from the user application by a distinct address space. It handles access requests to all system resources such as terminals, disks, printers, and networks.

PACKET LABELING

IP implements part of the network layer of the Open Systems Interconnection (OSI) Reference Model. IP is based on the datagram model. In this model, each data unit is treated as an isolated entity. All the information, such as a sensitivity label, necessary to transmit the data unit through the network is contained within the packet. IP datagrams contain a header which includes the source and destination addresses for the datagram and any other information that the network may require in order to transport the datagram from source to destination. Additional information can be included in the header in the form of IP options. The total amount of space that can be used by *all* the IP options sent in a datagram is limited to forty octets.

It is easy to see that a sensitivity label represented by human-readable ASCII characters could exceed forty octets in length. Thus security related information that is transmitted as an IP option is usually represented by numbers and not letters. Another reason for using numbers instead of letters is that label comparing is less costly. The computer resources required to compare two numbers is significantly less than that used when comparing two character strings.

Current IP Security Options

The Military Standard 1777 (MIL-STD 1777) specifies the Internet Protocol. Included as part of that document is a section on the defined IP options. There is a definition for an IP Security Option which includes fields for a security level, compartments, handling restrictions and transmission control code. Request For Comment 1038 (RFC 1038), currently in draft form, specifies changes to MIL-STD 1777 regarding two IP security options. The options are referred to as the "Basic Security Option" (BSO) and the "Extended Security Option" (ESO).

Basic Security Option

RFC 1038 has the following to say about the purpose of the DOD Basic Security Option.

"This option identifies the U.S. security level to which the datagram is to be protected, and the accrediting authorities whose protection rules apply to each datagram."

The BSO defines four security levels: "Top Secret", "Secret", "Confidential" and "Unclassified." It also identifies four accrediting authorities. The BSO option reuses the option type 130 which changes the definition of the option as defined by MIL-STD 1777. MLS/TCP supports the BSO and allows the security administrator to define the meanings of the security levels.

Extended Security Option

There were concerns that the BSO did not provide all of the label information that was needed. In response to this concern a flexible security option was created that allows a recognized authority to define the contents of the option. RFC 1038 specifies the DOD Extended Security Option as follows:

"This option permits additional security related information, beyond that present in the Basic Security Option, to be supplied in an IP datagram to meet the needs of registered authorities. If this option is required by an authority for a specific system, it must be specified explicitly in any Request for Proposal".

The ESO uses IP option type 133. See reference ^[3] for a detailed definition of each option. Due to the largely undefined nature of the ESO, we have chosen not to implement this option in the first release of our product.

Commercial IP Security Option

The Trusted Systems Interoperability Group (TSIG) ³ has proposed a new IP security option that better meets the

3. TSIG is composed of a group of vendors developing secure operating systems. They are working together to solve interoperability issues with respect to MLS networking.

requirements of transmitting security related information in an IP option in an open systems environment. The BSO and ESO are administered by the U.S. Department of Defense and meet defense department requirements. These requirements do not always satisfy those found in the commercial or open systems environments.

The Commercial IP Security Option (CIPSO) permits security related information to be passed between systems within a single Domain of Interpretation (DOI). A DOI is a collection of systems which agree on the meaning of particular values in the security option and which have a common security policy. The format of the CIPSO option is shown below.

8 bits	8 bits	32 bits	8 bits	8 bits	? bits		8 bits	8 bits	? bits
134	6 - 40	1 - 0xmmmm	1-255	1-34	?	...	1-255	1-34	?
option number	option length	DOI	tag id	tag length	info field		tag id	tag length	info field

The option length is the total length of the CIPSO option including the number and length fields. The Domain of Interpretation field is 4 octets in length. The remainder of the option is variable in length and contains a stream of tags. These tags are used to transmit additional security information associated with the datagram. TSIG has currently defined two tag types.

The first tag type is referred to as the "*bit-mapped*" tag type. Its format is shown below.

8 bits	8 bits	8 bits	0 - 248 bits
1	3 - 34	0 - 255	bit 1 : ... : bit 248
tag type	tag length	level	bit map of categories

The tag type is equal to 1. The tag length is the total number of octets including the tag type and length fields. The bit map can range from 0 to 31 octets in length. If bit N is a 1, then category N (as defined by the DOI) is part of the sensitivity label for the datagram. If bit N is a 0, then that category is not part of the label.

The second tag type is referred to as the "*enumerated*" tag type. It is used to describe large but sparsely populated sets of categories. Its format is shown below.

8 bits	8 bits	8 bits	8 bits	16 bits	16 bits
2	4 - 34	0 - 255	0 - 255	cat 1	... cat 15
tag type	tag length	level	flags	list of categories	

The tag type is equal to 2. The tag length includes the tag type and length fields. The flags field is interpreted as follows. If the least significant bit is a 0, then all the enumerated categories are part of the sensitivity label. If the bit is a 1, then all categories defined by the DOI are set excluding the ones listed. All other bits in the flag field are reserved for future use. Each enumerated category is 2 octets in length. This allows from 0 to 15 enumerated categories per CIPSO.

With the backwards compatibility requirement, MLS/TCP allows both BSO and CIPSO security options to be used. They can be used in any combination. The security administrator for the host determines the configuration of which IP security options to use for each network interface.

Label Mapping

The method of converting a human-readable sensitivity label to machine representation is a local issue. Each host is free to use any conversion it wants. Most implementations just create a mapping table where the human-readable security attribute is converted to a number. An entry is made in the table for every legal value for each security attribute defined in the host.

The use of numbers to represent security attributes introduces a new problem when used in the environment of networked computers. It is now necessary for each host that communicates with another host to use the same security attribute to number mapping conversion. One solution is that each host has a mapping table for every host it wishes to communicate with. This introduces the problem of maintaining a large number of mapping tables when the number of hosts grows large. Another solution is to have each host connected to the network use the same global mapping table. But this solution implies that all the hosts belong to the same security domain. These solutions represent the two extreme cases.

The CIPSO option avoids this problem through the use of a flexible yet manageable solution. In most situations, when a host joins a network, it will communicate with a set of hosts with which it has the requirement to share information. Since the set of hosts will be sharing information, the security policy regarding the protection of the information should be the same. Thus for each different group of hosts sharing information, a new Domain of Interpretation (DOI) is created. If all the groups share the same security policy, only one DOI is required. The DOI in the CIPSO option is then used to point to a mapping table that is common to all the hosts using the same DOI or within the same security domain of interpretation. MLS/TCP can support multiple DOIs for hosts that belong to more than one security domain such as gateways.

ROUTING

When a host is connected to a network, the security policy may state that data labeled at a certain security level is restricted to a particular path it takes through the network. IP normally chooses the least cost path, where cost is the number of hops that an IP datagram would traverse. TCP uses the datagram service provided by IP. TCP provides for the reliable delivery of a stream of data from source to destination. By using the services of IP, TCP will gain some of the datagram capabilities. One such capability is that IP will chose the path that an IP datagram takes dependent upon the current conditions in the underlying network. Thus if one gateway along a path goes down, IP could detect the problem and choose to route IP datagrams through a different path. Figure 1 depicts this situation. If host A wishes to communicate with Host B Secret information then it must use Net 1 or Net 3. If the routing policy does not take labels into account then the connection could be set up through Net 2. TCP will have provided the service requested but the security policy will be violated.

Thus, the algorithm that IP uses to determine the path that the datagram takes required modifications to make it cognizant of sensitivity labels. This change required that each physical network interface connected to the host be assigned a range of sensitivity labels. IP compares the label of the packet to be sent to the network label range. If the packet label is not within this range then that path will not be chosen.

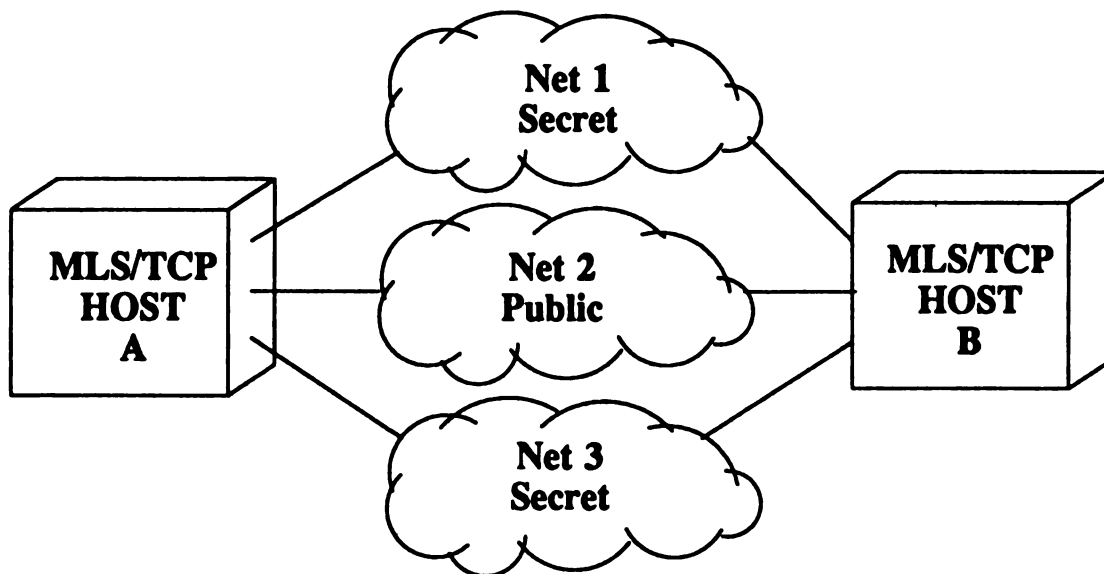


Figure 1: Multi-Level Secure Routing

NETWORK SERVERS

Network applications are sometimes described by a client/server model. The client and server together implement a defined application layer protocol. The client is the application that is requesting some service while the server is the application providing that service. The three most widely known network applications are the File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) and the TELNET Protocol (TELNET).

Servers normally accept connections from any host. Each service has assigned to it a unique "*well known*" port number. Using this port number, the server will notify TCP that it is willing to accept any connection requests to its port number. This is commonly called a "*passive open*." When a server posts a passive open, the TCP state for that connection is in the "LISTEN" state. Thus servers may also be called "*listeners*." The client application knows what service it is requesting on behalf of a user. With this information, it can look up the corresponding well known port number for that service. The client then makes an "*active open*" to the server's well known port number. The server gets notification from TCP that a client is requesting a network connection. The server can accept or reject the request. If accepted, the network connection is established and the client and server can then communicate. Server processes will normally spawn⁴ a child process and it is the child process that will perform the work requested by the client. The parent process is then free to go back to listening for new connection requests.

Untrusted Servers

When any server process requests that a passive open be performed, the networking software checks to see if the process has "*root*" privileges.⁵ If it does not, the networking software stores the sensitivity label of the server process as the "*session label*." This action is taken without any assistance from the server process. The session label is used to restrict all incoming connections to the server to have the same sensitivity label as the untrusted server process.

When a client connection request comes in, the networking software checks to see if the session label is set. If so, then a label compare of the sensitivity label from the IP datagram of the incoming connection request is made against the session label. If the labels are equivalent, then the rest of the processing for connection establishment is performed. If not equivalent, the client's connection request is rejected. This allows untrusted servers to be supported without modification while restricting their operation to a single label.

There is a generic problem with untrusted servers; any user of the system has the capability to create an untrusted server. This allows the import/export of data, albeit at a single level, without any identification or authentication processing being performed. We solved this problem by restricting all server executables to be stored at the "*system low*" level, a level at which normal users can not create executables.

Trusted Servers

When a process with "*root*" privileges requests a passive open, the networking software does not fill in the session label. When a client connection request comes in, the networking software detects that there is no session label set for the associated listener. The networking software then checks the sensitivity label of the incoming IP datagram to make sure that it is within a range of values that the security administrator has set for the host. If within range, the networking software notifies the server of the connection request. If the server accepts the connection request, the session label for the new connection is set to the label of the incoming IP datagram. This allows the trusted server to know the sensitivity label of the client process.

The server process spawns a child process. This child process still has "*root*" privileges. Before the child execs a program that will provide the service, it must perform a few tasks. First it must retrieve the session label from

4. In the UNIX environment, a new process is "spawned" by executing the system call "fork." This creates a new process that is an exact copy of the original process. It has the same security privileges and has access to all the same open files. The new process can then use the system call "exec" which overlays the current running process with another program if it wants to run a different program.

5. In a MLS UNIX environment an "*untrusted server*," is any server process that does not have "*root*" privileges.

TCP and change the process sensitivity label to the session label. The child process must then change its User Identification (uid)⁶ from "root" to another uid thus removing its "trusted" capability. Only after the child process has removed its "trusted" capability can the child process exec the program that will provide the requested service. In this way the program that provides the actual service does not need to be trusted. The uid that is selected can be predetermined based on the service the server is providing. For example if the server is providing the Simple Mail Transfer Protocol service, the uid is that of the "mail" daemon. Other services require that some other authentication mechanism be performed. For example the TELNET server relies on the supplied /bin/login⁷ program after setting up a terminal environment.

There may be cases when a server needs to be trusted in order to gain access to other system resources but it only wants to accept network connections at a specific session label. A trusted server is allowed to make a call to TCP that will set the session label. Thus when a client connection request is received, the networking software detects that the session label is set and processes the request as if the server were untrusted.

MODIFIED NETWORK APPLICATIONS

It was previously mentioned that the three most widely known network applications are implementations of the TELNET, FTP and SMTP protocols. This section goes into more detail on how the implementation of each network application had to be modified to be supported under MLS/TCP.

TELNET

The TELNET client application required no changes. This is due to the fact that the kernel TCP software can obtain security relevant information about the user of a client TELNET without any assistance from the TELNET program. The TELNET server also required no changes. The reason for this is that server TELNET is implemented in the kernel and it ultimately depends upon the /bin/login trusted program to perform the UNIX login processing.

FTP

The FTP client application required no modification. In order to support the server FTP program, two changes were required. First a trusted front-end to server FTP was created. This trusted program performed the identification and authentication portion of the FTP protocol. The FTP protocol for identification and authentication requires a user name and the password associated with the user name. After checking that the user name and password are valid, the trusted front-end makes a call to TCP to retrieve the session label. A check is then made to see if the user name has been authorized to process information at that session label. If not, an error is returned to the client FTP and the network connection is closed. If allowed, the trusted front-end changes the security attributes of the process to match those of the session label. It then execs the "original" FTP server program. The original "untrusted" FTP server program has been modified to disable the user name and password commands. The original FTP server remains an untrusted program that responds to the commands requested by the FTP client. The untrusted server FTP process can only access correctly labeled data because of the MAC and DAC checks performed by System V/MLS.

SMTP

The SMTP protocol application presented a different set of security considerations due to the fact that it is most often implemented and accessed on behalf of the user via the general internet mail routing application known as "sendmail". As a stand-alone protocol specification, SMTP as its name implies provides for a very simple set of handshaking and etiquette requirements. In contrast, the sendmail application is a complicated program which integrates the SMTP protocol implementation with such functions as mail collection, routing and queuing.

6. UNIX assigns a user identification number to each user account. The "root" account has always used the uid of 0. Thus a non-zero uid implies some user that does not have the trust associated with "root."

7. /bin/login is a trusted program used by UNIX to perform identification and authentication.

On the client side, no real modifications were necessary for the main processing path. A user wishing to send mail to a remote system uses the MLS mail interface program which in turn invokes sendmail to route the message to the remote destination. If sendmail determines that the SMTP protocol should be used to accomplish this task, it attempts to establish the connection. The networking software will automatically set the session label to the user's current operating level. Assuming the connection can be established and there are no violations of the "simple" protocol requirements, the message is delivered to the remote system and stored in a user mailbox file whose label matches that of the sending user. The problem arises when something goes wrong in this scenario such as the inability to connect to the remote system. In this case, sendmail queues the message for later delivery attempts. Queue processing in an MLS environment adds an additional complication to the sendmail application. The solution was incorporated into the trusted server section of the program.

Most of the changes made to the server side of this application in support of the MLS/TCP environment are similar to those already described for the other trusted server applications. Specifically, when invoked as a server, sendmail first verifies that it is executing with "root" privileges. If so, it sets up a trusted SMTP listener without an associated session label set. Subsequent attempts by client SMTP applications to establish connections are handled by spawning new processes which set the label of the server to match that of the incoming connection and the uid of the process to the uid of the mail daemon before continuing with the normal SMTP transfer function. This differs somewhat from the "identification and authentication" process described for the FTP and TELNET protocol applications as the uid is automatically set to the uid of the mail daemon. However, since mail accepted by sendmail's SMTP server will be delivered to the local user at the level at which the sending user invoked SMTP, if the local user is not authorized to operate at that level, he will not have access to the message. In fact, the MLS mail interface program will not even notify him that it exists.

As mentioned above, a second change to the sendmail server software was necessary to handle the queue processing. Standard sendmail implementations spawn a new process which retains "root" privileges to periodically examine the mail queue and attempt to deliver any accumulated messages. The server was modified to create a separate process for each level at which mail capability has been authorized and set the label of each process to match. The user id of all of these processes is set to the mail user ID. This solution was chosen because it reduced the amount of processing that required root privileges while minimizing the changes to the existing sendmail implementation.

NETWORK INTERFACE

One of the strengths of TCP/IP is that it can connect to many different types of networks. Some of the common types are 802.3 (Ethernet), token ring, X.25, or even a serial RS232 line. A secure TCP/IP can protect the data only while it is in the host. Once it leaves the host it is the responsibility of the network to protect it. For many Local Area Networks (LANs) this protection is just physical control of the communications media (the copper wire). For other networks there are devices that provide special security services such as encryption or mandatory access control. Each of these types of network interfaces have unique requirements pertaining to security. A secure TCP/IP should be configurable to handle the needs or shortcomings of these networks.

Trusted Networks

Within the context of this paper, a trusted network is one in which the security parameters provided are guaranteed to be accurate. These parameters may be provided by a network device or by the host at the other end of the connection. It must not be possible for a non-trusted host or user to be able to interject false security parameters into a trusted network.

One example of a trusted network is the Verdex VSLAN network.⁸ The security parameters (i.e., the sensitivity label) for each 802.3 packet is provided by a network interface card. We have modified an 802.3 network driver to accept the VSLAN label and convert it to a CIPSO or BSO label. The label and the data packet is then passed up to IP. IP attaches the label to the packet and sends it up to TCP or out another network interface depending

8. The VSLAN network was evaluated by the National Computer Security Center and has received a B2 rating.

on the IP destination address.

Another example of a trusted network is Blacker. The Blacker Front End (BFE) is a device that provides data confidentiality through the use of high-grade encryption. Blacker uses BSO to obtain the security level of data and performs access control based on this label. No additional changes were required to support Blacker.

A simple Ethernet network can be a trusted network if all hosts on the network are trusted. The level of trust provided by this network is equal to the level of trust of the least secure host on the network. For this network, the security parameters are passed in the TCP/IP protocol and a separate security interface is not required.

Untrusted Networks

Most networks in use today offer no security services and any security parameters provided by these networks can not be trusted. We could require MLS hosts to only connect to MLS networks, however that would not be practical. Our solution was to assign a fixed set of security attributes to these networks. These attributes are provided by the security administrator of the MLS host and reflect the security attributes associated with the untrusted network.

As mentioned earlier, all data coming into an MLS host must have a label. Datagrams from untrusted networks should not contain a label. If a label is present in the datagram then it can not be trusted. Our solution is to insert a CIPSO or BSO label into the IP datagram as it enters the MLS host. If a sensitivity label is already present in the datagram it is overwritten with the new label. This sensitivity label is obtained from the fixed set of security attributes assigned to that network. Figure 2 illustrates the function of a MLS/TCP gateway between an untrusted, non-labeled network and a trusted, MLS labeled network. If the label was not added then hosts on the untrusted network could not communicate with hosts on the MLS network where a label is required.

For packets going from the MLS host to the untrusted network we provide the capability to "strip" out the CIPSO or BSO label from the datagram. Some of the hosts on the untrusted network may not be able to handle an unrecognized IP option and may crash the host.

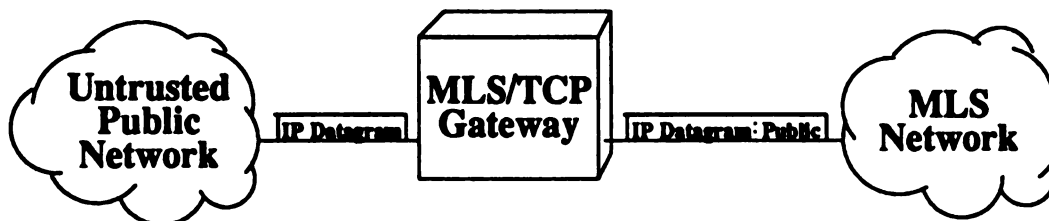


Figure 2: Label Insertion and Stripping

AUDITING

All security relevant events must be audited. A security relevant event is any action taken by a host or network which provides a user with access to a resource or that effects a change to security information. The information included in an audit record must be sufficient to determine the characteristics of the access or change. Below is a list of some of the events that are recorded.

1. All failed or successful connections to the host
2. Incoming packet with a label outside of the host label range
3. Outgoing connection refused due to no route found that meets security requirements for the level of the requested connection
4. Label on incoming packet contains a security level not recognized
5. CIPSO DOI on incoming connection not supported by the host
6. TCP connection closed

These new audit records are included in the host audit record. Some TCP/IP events could generate a large amount of audit records and overwhelm the system. For example, all audit events at the packet level could generate an audit record for every packet associated with a particular connection. For this reason we have included the capability to allow the security administrator to turn off the recording of any event that the administrator determines is not needed.

An argument could be made that all packets received should be audited. As mentioned above this would quickly consume all the disk space on the system. Since TCP is a connection oriented protocol, we feel that just auditing the success or failure of the connection is enough. The operation of connecting to a remote host using TCP is analogous to opening a file. The Orange Book^[4] requires the file open to be audited, but does not require auditing of the individual reads or writes. Likewise auditing the closing of a file is also required and TCP audits the closing of each connection. UDP (User Datagram Protocol) is a connectionless protocol and audit of each packet would probably be required. The networking software does not currently implement a trusted UDP but one is planned for a later release.

ASSOCIATION WITH THE TNI

The Trusted Network Interpretation (TNI), also known as the "Red Book" describes the requirements for MLS networks. It recognizes that most networks are made up of many components each of which may provide a different security service. For this reason the TNI breaks the requirements for a secure network into four distinct areas. These areas are Identification and Authentication (I&A), Mandatory Access Control (MAC), Discretionary Access Control (DAC), and Audit. The implementation described in this paper is designed and implemented to satisfy the MAC and Audit requirements. It is expected that the DAC and I&A requirements are satisfied at a higher layer in the protocol stack.

CONCLUSIONS

Despite the complicated nature of the MLS requirements, the design and implementation of this project went very smoothly. TCP/IP already embodied many important concepts such as data separation and integrity. The flexibility of the options in the IP header was a particularly critical ingredient. Many of the changes involved hooks in the TCP/IP that called new operating system routines. There were no major rewrites of TCP/IP or UNIX code.

Most of the new capabilities have been embedded in the internal workings of TCP/IP and can not be seen outside of the host or even by the user. The only change seen outside of the host is the newly supported IP security labels and those can be stripped if not needed. The user application interface to TCP/IP has not been changed so all applications should continue to operate. Some additional applications interface features were added to support trusted applications that understood labeling.

REFERENCES

1. C.W. Flink and J.D. Weiss, "System V/MLS Labeling and Mandatory Policy Alternatives", Proceedings of the 1989 Winter USENIX Conference, February, 1989.
2. National Computer Security Center, "Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria", NCSC-TG-005, 31 July 1987
3. M. St. Johns, "Draft Revised IP Security Option", Request For Comments 1038, January 1988.
4. Department of Defense Standard 5200.28-STD, "Department of Defense Trusted Computer System Evaluation Criteria", December 1985

THE CASCADE PROBLEM: GRAPH THEORY CAN HELP

John A. Fitch, III¹ and Lance J. Hoffman

Department of Electrical Engineering and Computer Science

George Washington University

Washington D.C. 20052

Abstract

This paper presents a new approach, based on finding shortest paths in a graph, for solving the cascade problem. The result is an efficient ($O(N^3)$) algorithm, where N is the number of security domains in the network. The paper provides background on the cascade problem, generalizes the problem from its traditional military roots, and then applies the shortest path technique to a military example. The shortest path approach appears quite general and provides a method based on established mathematics for evaluating network security.

Keywords: Cascade problem, graph theory, shortest path, network security, risk analysis.

1. The Cascade Problem

The cascade problem was first defined and discussed in [14]. The importance of the cascade problem is that it demonstrates how networking systems together may produce unacceptable risks even though the individual systems in the network are secure and reasonable interconnection rules are followed. "Reasonable interconnection rules" means that the network connections comply with security policy and are secure from external attacks such as wiretapping. This paper provides background information on the cascade problem, generalizes the problem from its traditional military roots, and applies a resource-constrained shortest path technique to a military example. The result is a new, efficient ($O(N^3)$) algorithm, where N is the number of security domains in the network, for determining if a network has a cascade problem. This graph-theoretic approach appears quite general and provides a method based on established mathematics for evaluating network security.

1.1. The General Cascade Problem

The cascade problem is described in [14, 8]. Both references focus on cascading in military networks where both security risk assessment and system security evaluation use Defense Department standards and guidelines. This section describes the cascade problem in more general terms, provides the background information to understand the types of networks in which cascading may be a concern, and presents a military example of the cascade problem.

1.1.1. General Cascade Problem Definition

The cascade problem belongs to a subspace of the problem set that asks, "If secure systems are connected together, is the resulting network secure?". This section partitions the problem set to place the cascade problem in perspective and then presents a more formal definition of cascading.

Before partitioning, it is first necessary to define *secure system*. This paper defines a secure system as a system that has undergone both a system security evaluation and a risk analysis evaluation that results in an acceptable risk of operating the system. A risk analysis considers the assets of a system and threats against it to determine how much security is sufficient. System security can be modeled as a

¹Mr. Fitch is also affiliated with GTE Government Systems

function of several parameters: physical security, personnel security, administrative security, communications security, and computer security [15]. These parameters can be represented by classes of countermeasures that reduce system risks. For example, physical security can be described by the class of countermeasures that includes locks, fences, and guards. A system security evaluation, therefore, measures the effectiveness of the countermeasures used in the system.

The first partitioning of the network security problem space is to divide the space into networks that (for security purposes) can be treated as a single system and networks that cannot be treated as a single system. Cascading is only a concern in the latter type of network. There are reasons why some networks cannot be or are not viewed as single systems. First, the network may be so large that a single system security evaluation is not feasible, so a divide-and-conquer approach must be taken. Second, the network may be made up of systems that are owned or operated by differing administrative entities or systems that use different system security evaluation or risk assessment methods.

Having limited the problem space to networks that either are not or cannot be evaluated as single systems, the next step is to reduce the problem space by examining the conditions under which two systems would interconnect. As a minimum, the administrators of the two systems have to mutually agree that the other system is secure in its own environment; that is, they need to understand and accept the risk assessment and security evaluation methods used by the other and believe that the analysis was done correctly. This does not imply that all the systems on the network are equally secure: it means that each system recognizes that the other's security is good enough as a stand-alone system (that is, before the interconnection is considered). If one system does not believe that the other is secure, then there is a clear risk to sharing data with that system. For example, two systems that implement completely different security policies or conduct very different evaluation methods are unlikely to share sensitive data. For the cascade problem, only mutually recognized secure systems are interconnected and each system provides the other with its system security evaluation metrics.

Having mutually recognized that the other system is independently secure, the next step is to decide which assets (or classes of assets) are to be shared with the other system. This step is closely related to mutually accepting the other system's security evaluation because each system must identify a subset of assets for export that it believes the other system will protect accordingly. This does not imply that the two systems must have identical export sets: the exchange may be one way, with one system acting only as an exporter and the other acting only as an importer.

Because each exporting system believes that the importer will properly protect the exported asset, it implicitly believes that the importer will share the asset with third-party systems only if those systems are also secure. This means that a system needs to consider only the security of the system directly involved in the interconnection and not the security of all the systems in the network in order to be assured that the exported asset is properly protected. This "nearest neighbor" approach thus creates an implied transitive property of protection.

Because the systems agree to share assets via a network connection, the security of the connection itself must be addressed. The cascade problem assumes that the interconnection mechanism itself is secure; (that is, assets are not threatened when on the connection) and that the threats are only at the two systems involved in the connection.

In summary, the following type of network is being considered:

- The network consists of independent secure systems; that is, each system in the network, based on its own risk analysis and system security evaluation, is secure before considering network connections.
- For size or political reasons the network cannot be treated as a single system and undergo a security evaluation similar to that of the component systems in the network.

- Before agreeing to an interconnection, each system mutually recognizes the security of the other.
- The systems involved in a connection only share assets that the exporting system believes the importing system will protect properly.
- The connection itself is secure; that is, there is no threat posed against data while in transit between the systems.

Limiting the discussion to these types of networks, it is now possible to define when a cascade problem exists:

A cascade problem exists when independent, mutually recognized secure systems are interconnected by secure channels to create a network system that is not secure.

1.1.2. Why Cascade Problems Occur

The existence of the cascade problem results from several factors. The decision to allow an interconnection between systems was based only on assuring the protection of the assets being shared; it was not based on all the assets in the source and destination systems. This at first appears adequate because the two systems are independently secure, but the fact of the interconnection means that the two systems are no longer truly independent. The cascade problem exploits these two facts in a subtle fashion based on risk analysis principles.

One purpose of a risk analysis is to determine how much security is needed to protect an asset. Because the asset has some determined value, there is a threshold on the amount one is willing to spend on protection. For example, one may not be willing to spend \$75 on a safe to protect a \$100 watch, but may be willing to spend \$20 to buy better locks for the door: there is a limit at which one accepts the residual risk to an asset rather than pay more for security. Another way to view this concept is that security is measured by the amount of effort required to steal the watch. The watch owner wants the thief to have to spend the effort to defeat a \$20 lock in order to steal the watch. In a computer system, there is an analogous threshold where one is willing to accept the residual risk to the asset (such as compromise or destruction of data) rather than incur the cost of additional protection (see [16]). The definition of a secure system in the previous section is consistent with this cost/reward observation.

A penetration (either by a human or by "nature") of one of the systems may cause other systems' assets to propagate to an interconnected system. While a stand-alone secure system that suffers a penetration is, by definition, willing to accept the local penetration as within acceptable risk, that system does not necessarily accept the export of other assets as within acceptable risk. (This was the point of identifying import and export sets.) Thus the cascade problem is essentially a risk assessment problem that measures network risk based on local risk metrics of an export of data not in the export set. The problem is called cascading because the links between the systems act as conduits that cascade assets along a path between systems. If the assets arrive at a system that does not adequately protect them, then a cascade problem exists.

Thus, determining if a network has a cascade problem requires identifying if the network is of the type identified in the previous section, stating the acceptable level of risk against loss by cascading, calculating the actual cascade risk based on the network configuration, and assessing if the cascade risk exceeds the acceptable level. As in the example of the thief and the watch, security from cascading can be measured by the amount of effort required to defeat the protection mechanisms. Security from cascading can be measured by requiring a penetrator to expend a stated quantity of resources to affect the penetration(s) necessary to cause a loss via cascading. From a penetrator's perspective, cascading can be viewed as an accumulation of costs as the penetrator creates a path of penetrations through the network.

1.2. Military Cascade Example

To derive a specific cascade problem from the general cascade problem requires indicating the risk assessment and system security evaluation methods used by the systems in the network. This section briefly reviews the risk analysis and evaluation methods used in the military cascade problem as defined in [14, 8] and presents an example of a network with a military cascade problem.

The risk assessment method used in [14, 8] is based on the environment guidelines given in [12, 13] where assets values are measured by the security classifications of the data in the system and the threats are measured by the minimum user clearance in the system. The risk analysis method uses the maximum data classification and minimum user clearance as indices into a table to determine a recommended amount of computer security for the system. The amount of computer security is measured by a specific rating defined in the Orange Book [10, 11]. Figure 1-1 shows a table from [13] that maps a (minimum user clearance, maximum data sensitivity) pair to a required Orange Book level of computer security. The Orange Book computer security ratings are ordered as $D < C1 < C2 < B1 < B2 < B3 < A1$.

		Maximum Data Sensitivity						
Minimum Clearance or Authorization of System Users		U	N	C	S	TS	1C	MC
	U	C1	B1	B2	B3	•	•	•
	N	C1	C2	B2	B2	A1	•	•
	C	C1	C2	C2	B1	B3	A1	•
	S	C1	C2	C2	C2	B2	B3	A1
	TS(BI)	C1	C2	C2	C2	C2	B2	B3
	TS(SBI)	C1	C2	C2	C2	C2	B1	B2
	1C	C1	C2	C2	C2	C2	C2	B1
	MC	C1	C2	C2	C2	C2	C2	C2

Figure 1-1: Security Index Matrix For Open Environments (adapted from [13])

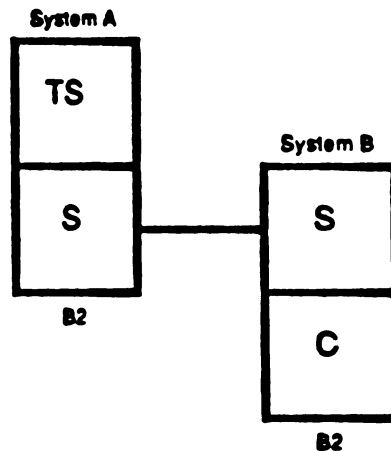
The Orange Book rating is used as the computer security portion of a system security evaluation that also includes other factors, such as physical and procedural security. The cascade problem in [14, 8] considers only the computer security portion of a system security evaluation. To simplify the mutual recognition of each system's security and to follow the example from [14, 8], only the computer security portion of the system security evaluation is considered here as well.

The next step is to define the acceptable import and export sets between systems. This is done by requiring that the interconnection between systems obeys the military multilevel security policy of "no read up" and "no write down" between data at different classification levels. The classifications in the example are ordered as CONFIDENTIAL < SECRET < TOP SECRET. See [2] for details on the multilevel security policy and [3] for a general lattice-based model of secure information flow.

Having reviewed the military risk assessment and security evaluation methods, the military cascade problem can now be discussed. The cascade problem for the military multilevel system is informally

defined in [14] as when a penetrator can take advantage of the network connections to compromise data over a range of sensitivity levels that is greater than the accreditation range of any of the systems that must be defeated to do so. (An accreditation range is the set of security levels a system is trusted to process and separate correctly according to the information flow policy).

The example shown in Figure 1-2 from [14] demonstrates the military multilevel cascade problem. System A has an accreditation range of (SECRET, TOP SECRET) and the minimum user clearance is SECRET. System B has an accreditation range of (CONFIDENTIAL, SECRET) and the minimum user clearance is CONFIDENTIAL. Based on the guidelines in [13] and shown in Figure 1-1, System A requires at least B2 computer security and System B requires at least B1 (System B's rating of B2 in Figure 1-2 satisfies this constraint).



TS - TOP SECRET, S - SECRET, C - CONFIDENTIAL

Figure 1-2: A Network With A Cascade Problem

Each of the systems agrees to export only SECRET information to the other. This interconnection conforms to the military information flow policy and thus defines the allowed export set.

The cascading in this network occurs by assuming a penetration of the operating system protection mechanisms at both end systems. If a penetrator compromises System A, TOP SECRET information may be leaked via the SECRET connection to system B. If system B is compromised, then this TOP SECRET information may be leaked to a user who is only cleared CONFIDENTIAL. Thus the network has a cascade problem because the penetrator has compromised three levels of data by defeating two systems with accreditation ranges consisting of two levels of data.

To determine if a network has a cascade problem, the next section formulates the multilevel military cascade problem as a resource-constrained shortest path problem.

2. Shortest Path Formulation of the Military Cascade Problem

Formulating the cascade problem as a resource-constrained shortest path problem provides an efficient algorithm for determining if a network has a cascading problem and thus improves greatly on the heuristic presented in Appendix C of [14]. The resource-constrained shortest path algorithm is also superior to the algorithm designed by Millen [9] based on matrix multiplication. There are several

motivations for performing a cascade analysis. For example, a system administrator may make a decision to join or not to join a network based on the risk posed by cascading. In a network where there is additional cooperation between the system administrators, the network can possibly be re-architected to eliminate the cascade so that all parties may securely use the net.

The resource-constrained shortest path approach to determine whether or not a network has a cascade problem is based on three phases: Preprocessing, Shortest Path Calculation, and Postprocessing. The details of each of these steps is provided in the following sections.

2.1. Preprocessing Step

The preprocessing step consists of three actions:

- Defining the cascade problem as a graph by identifying nodes, edges, and weights;
- Viewing the problem from the penetrator's perspective by allocating the penetrator a set of resources; and
- Defining the resource consumption function that determines how the network consumes the penetrator's resources.

The formulation of the cascade problem as a graph begins with the definition of *protection domains*. Appendix C of the Trusted Network Interpretation [14] defines a protection domain as a (system, level) pair. The protection domains in Figure 1-2 are (A, TOP SECRET), (A, SECRET), (B, SECRET), and (B, CONFIDENTIAL). The protection domains are the nodes of the graph in the shortest path formulation of the cascade problem.

The edges in the graph are the flows between protection domains. Viewing the problem from the penetrator's perspective, edges are assigned as follows:

1. An edge between nodes (protection domains) is created if it represents a network interconnection. This edge is weighted 0 because it is an allowed flow under the military flow policy and, therefore, represents no cost to the penetrator.
2. An edge between nodes internal to the same host system is created if it represents an allowed information flow. This edge is weighted 0 because it conforms to the military flow policy and therefore represents no cost to the penetrator.
3. An edge between nodes internal to the same host system is created if the flow represents a downgrade; that is, if the flow is not allowed by the military flow policy. This edge is weighted by the Orange Book computer security rating of the host system because it represents having to defeat the computer protection mechanisms in order to achieve the information flow.
4. For mathematical completeness, flows from a node to itself cost 0 and all other node pairs receive an edge weight of infinity.

The path a penetrator can follow through the network thus consists of steps consisting of penetrations internal to a host system or a legitimate network link.

Whether or not to consider allowed flows internal to a system depends on whether the objective is to locate the core paths that actually cause the cascades (achieved by not considering flow 2 above) or to locate all information flows that may be threatened by the cascade via legitimate flows into the core cascading paths (achieved by including the type 2 flows). This paper will not apply the type 2 flows to the example problem and will thus search for core cascading paths.

Having defined the nodes, edges, and weights, the next step is to allocate a set of resources to the penetrator and define a resource consumption function. The cascade problem in [14] treats the source and destination protection domains as requiring the same level of protection as a stand-alone system;

that is, any network path of protection domains must meet the computer security protection given in the Environments Guidelines [12, 13] (see Figure 1-1). For a specific source and destination, one could use the matrix lookup to determine the required path protection and allocate that quantity of resource to the penetrator. Because the concern here is to find all cascading paths, it eases analysis to calculate the cost of all paths and then test the path cost against the required path protection as part of the postprocessing stage rather than preallocate a fixed resource to be used for path pruning during the shortest path calculation.

The consumption function used here is similar to what Millen calls the *path resistance* [9]: the cost of a path is the cost to the penetrator of achieving the information flow from source to destination protection domain. According to [14] and [8], the example of Figure 1-2 has a cost of B2 for the cascading path between the (A, TOP SECRET) and (B, CONFIDENTIAL) protection domains. This cost of a path is found by taking the maximum of the costs of the edges in the cascading path. This results in a consumption function that states that for the military cascade problem, the amount of penetrator resources consumed on a path between protection domains is equal to the largest edge cost in the path. Naturally, the penetrator wants to minimize the path cost between source and destination domains because it represents the level of effort required to effect a cascade. This objective (minimizing the consumption function) has now mapped the cascade problem to a resource-constrained shortest path problem.

The consumption function for the military cascade problem implicitly assumes that if a penetrator can defeat a system with a specific security rating (B2 in the example), the penetrator can defeat other systems with the same rating with no significant additional effort. By viewing the problem from a shortest path perspective, other consumption functions can be easily defined and tested. For example, if one assumes that all system penetrations are independent, then the corresponding consumption function simply sums the cost of all the edges along the path. A corporate cascade example that uses summation as the consumption function is in [4].

The network shown in Figure 2-1 is used to demonstrate the shortest path method for finding cascading paths. The results of the preprocessing step are shown in Figure 2-2. The security levels in the circles are the graph nodes and the dashed boxes indicate the domain in which the nodes belong. Note that the example network includes both one-way and bidirectional flows. The adjacency matrix for the preprocessed system is also shown in Figure 2-2.

2.2. Shortest Path Calculation

Having defined the nodes, weights, edges, and the consumption function, it is now possible to apply the shortest path algorithm. Because the objective is to first determine if the network has a cascading problem, an all-pairs algorithm is used to calculate the shortest path costs between all pairs of security domains. Should the all-pairs algorithm indicate a cascade problem, a specific source-destination shortest path algorithm can be used to yield the actual path involved. (The act of determining all the edges in the shortest path can actually be incorporated into the all-pairs algorithm, but the two steps are kept separate here for clarity.) The all-pairs algorithm presented here is similar to that of finding the transitive closure of a graph. In fact, as long as the relationship between the edge weights and the consumption function forms a closed semiring, an N^3 algorithm can be used [1] to find all paths. This is indeed the case because the computer security evaluations can be ordered as $D < C1 < C2 < B1 < B2 < B3 < A1$ and, therefore, mapped to integer values; and because the operations minimum and maximum needed to optimize and express the consumption function can be shown to be valid $+$ and \circ operations, respectively, on the closed semiring of integers [5].

As a consequence, the all-pairs algorithm shown in Figure 2-3, which is modified from [6], is used. The graph is stored as an N -by- N adjacency matrix named *cost*, where N is the number of protection domains. The array *a* is the resulting N -by- N matrix of least path costs under the military consumption function defined in the previous section. The line

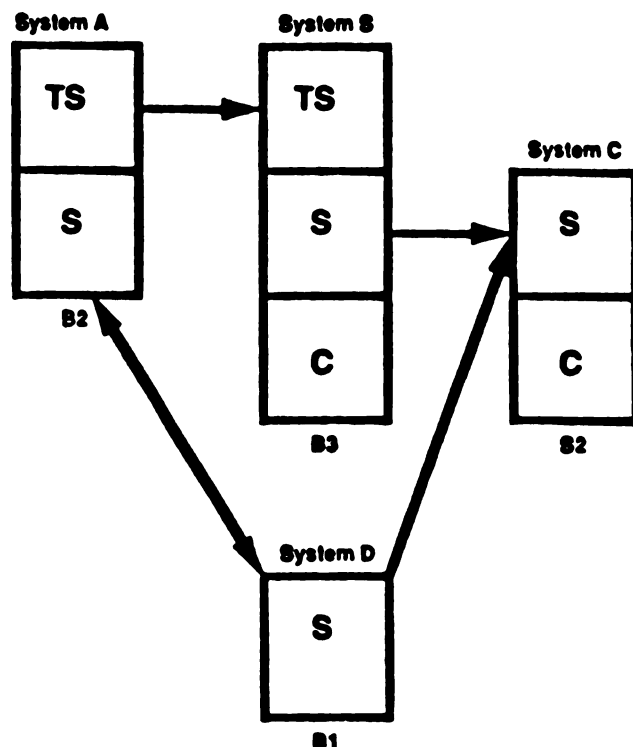


Figure 2-1: Military Network With A Potential Cascade Problem

$$a[i, j] = \min(a[i, j], \max(a[i, k], a[k, j])) \quad (1)$$

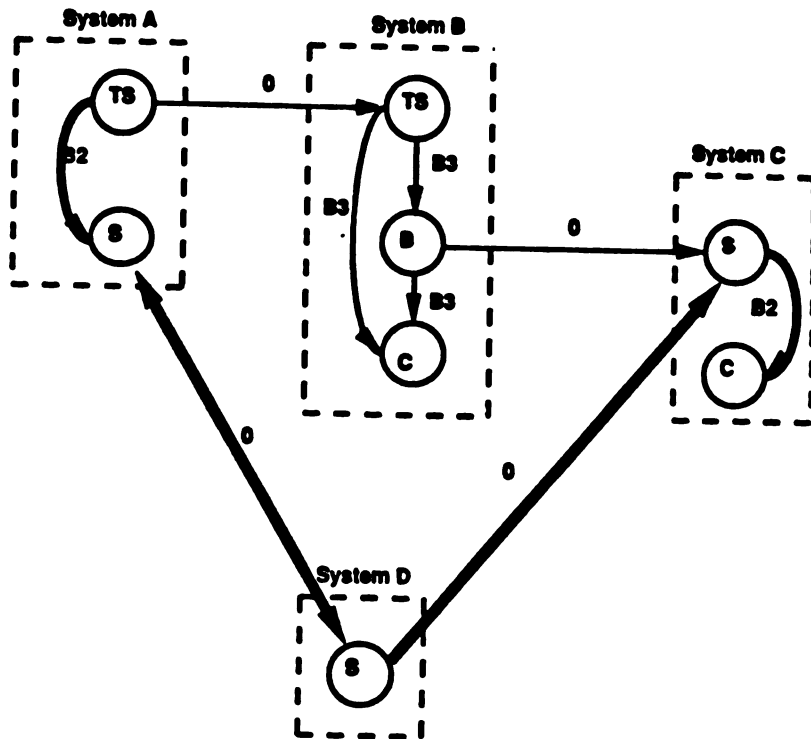
in the algorithm represents minimizing the military consumption function.

The all-pairs algorithm provides a shortest path solution in $O(N^3)$ time, although the postprocessing phase has not been considered yet. The shortest path cost results table is shown in Figure 2-4. As will be shown, the postprocessing is $O(N^2)$ so the computation complexity to determine if the network has a cascade problem remains as $O(N^3)$ where N is the number of protection domains. For comparison, Millen's work [9] is not quite as efficient. He calculates the resistance of all paths in the network by a matrix computation requiring $O(N^3 \log_2(N))$ steps.

2.3. Postprocessing Step

Figure 2-4 shows the cost of the shortest path between all pairs of (source, destination) security domains under the consumption function defined in equation (1). This path cost represents the minimal effort required by a penetrator to effect an information flow from the source to destination domain. The Postprocessing step determines whether the cost of the paths is within acceptable risk. This is done by considering the minimum user security clearance at the destination domain for all pairs of security domains. The real system risk is not that a penetrator has simply achieved a flow from one source security domain to another at an unacceptable cost; the risk is that a user who is not cleared for the information (as it was protected at the source domain) may actually obtain this information at the destination domain.

The risk acceptance test can be done as a set of table look ups for each security domain pair as follows:



Src/Dest	(Sys A,TS)	(Sys A,S)	(Sys B,TS)	(Sys B,S)	(Sys B,C)	(Sys C,S)	(Sys C,C)	(Sys D,S)
(Sys A,TS)	0	B2	0					
(Sys A,S)		0						0
(Sys B,TS)			0	B3	B3			
(Sys B,S)				0	B3	0		
(Sys B, C)					0			
(Sys C,S)						0	B2	
(Sys C,C)							0	
(Sys D,S)		0				0		0

TS = TOP SECRET, S = SECRET, C = CONFIDENTIAL

Figure 2-2: Military Network After Shortest Path Preprocessing

```

procedure allpairs(
    cost : adjacencymatrix, {initial edge cost matrix}
    var a:adjacencymatrix,  {all pairs shortest path costs}
    n :integer)              {number of security domains}
{ Computes the shortest path cost between all pairs of domains}
{ cost[1..n,1..n] is the initial graph cost adjacency matrix }
{ a[1..n,1..n] is the cost of shortest path between nodes }
var i : integer; {loop control for source nodes}
    j : integer; {loop control for destination nodes}
    k : integer; {loop control for intermediate nodes}

begin
    {copy the array cost into the array a}
    for i := 1 to n do
        for j = 1 to n do begin
            a[i,j] = cost[i,j];
        end;

        {Calculate shortest path cost for all domain pairs}
        for k = 1 to n do {for path with highest node index k}
            for i = 1 to n do {for all possible source nodes}
                for j = 1 to n do {for all possible destinations}
                    {if i->k->j cost is smaller than current i->j cost}
                    {then update the i->j path cost. In other words, }
                    {minimize the consumption function}
                    a[i,j] = min(a[i,j], max(a[i,k], a[k,j]))
                end;
            end;
        end;
    end.

```

Figure 2-3: All-Pairs Shortest Path Algorithm (adapted from [6])

Src/Dest	(Sys A,TS)	(Sys A,S)	(Sys B,TS)	(Sys B,S)	(Sys B,C)	(Sys C,S)	(Sys C,C)	(Sys D,S)
(Sys A,TS)	0	B2	0	B3	B3	B2	B2 **	B2
(Sys A,S)		0				0	B2	0
(Sys B,TS)			0	B3	B3	B3	B3	
(Sys B,S)				0	B3	0	B2	
(Sys B, C)					0			
(Sys C,S)						0	B2	
(Sys C,C)							0	
(Sys D,S)		0				0	B2	0

TS = TOP SECRET, S = SECRET, C = CONFIDENTIAL, ** = CASCADE

Figure 2-4: Military Shortest Path Cost Results

1. Look up the minimum user clearance for the destination security domain's system and determine the larger of the security level of the destination domain and the security level of the minimum user clearance.
2. Use the results of step 1 and the security level of the source security domain as an index into Figure 1-1 to determine the required amount of computer security.
3. From Figure 2-4, look up the actual cost to the penetrator to achieve the information flow between the source and destination domains. If the actual cost to the penetrator is less than the amount of security required by step 2 above, a cascade problem exists.

For the network in Figure 2-1, recall that the minimum user clearance at System A was SECRET, at System B CONFIDENTIAL, at System C CONFIDENTIAL, and at System D SECRET. As an example of the risk acceptance test, consider the flow from (System A, TOP SECRET) to (System C, CONFIDENTIAL). Step 1 of the postprocessing results in a value of CONFIDENTIAL because both the minimum user clearance at System C and the security level of the destination domain are CONFIDENTIAL. Step 2 consults Figure 1-1 using TOP SECRET as the source data sensitivity and CONFIDENTIAL as the minimum user clearance to obtain a recommended computer security rating of B3. In Step 3, referencing Figure 2-4 shows that the actual cost to the penetrator to achieve the flow from (System A, TOP SECRET) to (System C, CONFIDENTIAL) is B2. Since B2 is less than B3, a cascade condition exists for this path. The entry marked ** in Figure 2-4 shows the source and destination security domains that make up the cascade in Figure 2-1. The core cascade path is from (A, TOP SECRET) to (A, SECRET) to (D, SECRET) to (C, SECRET) to (C, CONFIDENTIAL) with a total cost of B2. This path is shown in bold in the upper half of Figure 2-2.

The postprocessing step to test all security domain pairs for a cascade problem can be done in $O(N^2)$ time. There is a total of N^2 domain pairs and the processing for each domain pair requires performing a table lookup from a table of minimum user security levels, from the shortest path results table, and from the table shown in Figure 1-1. The table references plus the comparison of recommended security to the penetrator's actual cost can be done in constant time, resulting in a total $O(N^2)$ complexity for the postprocessing step. Thus the complexity of the overall cascade problem is dominated by the $O(N^3)$ shortest path calculation. Should the actual path causing the cascade be desired, either the all-pairs algorithm in Figure 2-3 should be modified to save the paths as the costs are calculated, or a specific source-destination algorithm should be run on the domain pairs found to have a cascade problem. An algorithm to find the shortest path between a specific pair of nodes is $O(N^2)$ [1], so locating the actual cascading paths can be done without changing the $O(N^3)$ complexity for the overall problem.

2.4. Interpreting the Military Consumption Function

One way to view the military consumption function is that it makes a network risk assessment policy decision that once a computer system with a particular security rating (say B2) is defeated, the defeat of another system with the same level of protection does not cost the penetrator any significant amount of effort. This implies that no matter how many B2 systems are connected in series, a network system created from them will never afford the protection of a B3 system. This consumption function makes sense if one is looking for a worst case analysis of the problem or if it is realistic to assume that the systems in the network suffer from identical or similar flaws so that once one system is defeated, all similar systems are easy to defeat. However, it is easy to postulate other consumption functions based on different assumptions about the (lack of) interdependence of defeating individual systems in a network. For example, assuming that system penetrations are independent events results in a consumption function that is identical to the "normal" shortest path calculation; that is, it minimizes the sum of the edge costs in the path. A corporate cascade example that uses this consumption function is in [4]. As long as the consumption function forms a closed semiring, an $O(N^3)$ algorithm exists for solving the cascade problem under that function.

3. Conclusions

This paper has presented a new method, based on the resource-constrained shortest path, for solving the cascade problem. There are several conclusions:

1. The generalization of the cascade problem and its formulation as a resource-constrained shortest path problem point out the underlying security issues in interconnecting independently evaluated systems; this process leads to a broader understanding of network security risks.
2. Requiring a consumption function to be defined forces a clear policy statement about the (lack of) interdependence of defeating individual systems in a network.
3. A broad set of consumption functions can be defined that allows for a network risk function to reflect a given system's dependence or independence from its peers.
4. The shortest path formulation can detect and locate cascades in $O(N^3)$ time as long as the consumption function and minimization form a closed semiring operating on the graph. The military consumption function presented here and a corporate consumption function in [4] are well-behaved and demonstrate that the semiring requirement is not overly strict.

The resource-constrained shortest path approach and the concept of a consumption function appear quite general. Potential extensions to the basic approach presented here and suggested applications include the following activities:

- Analyze the computational complexity of the algorithm when the consumption function is not as well behaved as in the examples presented here.
- Investigate the effects on the approach when a vector of resources rather than a scalar is involved.
- Explore path-pruning algorithms that incorporate the penetrator resource set into the shortest path calculation step. Compare the path-pruning approach to the method presented here that uses the resource set as a threshold during the postprocessing step.
- Investigate techniques for reducing the number of security domains that must be considered in the cascade problem.
- Compare the ability of the shortest path consumption function to reflect the interdependence of a system from its peers with a statistical analysis of the cascade problem, such as that done by Ted Lee [7].
- Develop precise methods for systems to mutually acknowledge each other's security.

References

1. A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Mass., 1974.
2. D.E. Bell and L.J. LaPadula. *Secure Computer Systems: Mathematical Foundations*. Tech. Rept. ESD-TR-73-278, Volume 1, The MITRE Corporation, Bedford, Mass., March, 1973.
3. Dorothy E. Denning. *Cryptography and Data Security*. Addison-Wesley, Reading, Mass., 1982.
4. J.A. Fitch and L.J. Hoffman. *A Network Shortest Path Security Model*. Tech. Rept. GWU-IIST-90-32, George Washington University, Washington, D.C., September, 1990.
5. John A. Fitch, III. *A Network Security Model Based on the Resource Constrained Shortest Path*. Ph.D. Th., George Washington University, Washington, D.C., 1991. To appear..
6. E. Horowitz and S. Sahni. *Fundamentals of Data Structures in PASCAL*. Computer Science Press, Rockville, Maryland, 1984.
7. Theodore M.P. Lee. *Statistical Models of Trust: TCBs vs. People*. Proceedings of the IEEE Symposium on Security and Privacy, April, 1989, pp. 10-19.
8. J.K. Millen. *The Cascading Problem for Interconnected Networks*. Fourth Aerospace Computer Security Applications Conference, December, 1988, pp. 269-273.
9. J.K. Millen. *Algorithm for the Cascading Problem*. In *Internet IEEE Cipher News Group*, J. P. Anderson, Ed., June 25 IEEE Cipher forum on DOCKMASTER.NCSC.MIL, 1990.
10. Computer Security Center. *Department of Defense Trusted Computer System Evaluation Criteria*. CSC-STD-001-83, Department of Defense, Computer Security Center, Fort G.G. Meade, Maryland, August, 1983.
11. National Computer Security Center. *Department of Defense Trusted Computer System Evaluation Criteria*. DoD 5200-28.STD, Department of Defense, Fort G.G. Meade, Maryland, December, 1985.
12. National Computer Security Center. *Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments*. CSC-STD-003-85, National Computer Security Center, Fort G.G. Meade, Maryland, June, 1985.
13. National Computer Security Center. *Technical Rationale Behind CSC-STD-003-85: Computer Security Requirements*. CSC-STD-004-85, National Computer Security Center, Fort G.G. Meade, Maryland, June, 1985.
14. National Computer Security Center. *Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria*. NCSC-TG-005, National Computer Security Center, Fort G.G. Meade, Maryland, July, 1987.
15. T.A. Rullo. *Advances in Computer Network Security Management*. Heyden & Sons, Inc., 1980.
16. Rein Turn and Norman Z. Shapiro. *Privacy and Security in Databank Systems: Measures of Effectiveness, Costs, and Protector-Intruder Interactions*. In *Security and Privacy in Computer Systems*, Lance J. Hoffman, Ed., John Wiley & Sons, Los Angeles, CA, 1973. Originally published as Rand Corporation Report P-4871, 1972.

A CASE STUDY FOR THE APPROACH TO DEVELOPING A MULTILEVEL SECURE COMMAND AND CONTROL INFORMATION SYSTEM

**James Obal
Supreme Allied Commander Atlantic
U.S. Naval Base
Norfolk, Virginia 23511-6696**

**William Grogan
Contel Federal Systems
15000 Conference Center Drive
P.O. Box 10814
Chantilly, Virginia 22021-3808**

ABSTRACT

This paper presents a case study of two NATO Command and Control Information Systems (CCIS) projects with stringent computer security requirements. These projects were conceived and initiated at a time when trusted products were not readily available and the concepts of trusted system development and evaluation were not well understood. These circumstances have necessitated the Government and the contractor to seek a unified approach to integrating security into the development process; to ensuring that security requirements are satisfied; and to performing the security evaluation. That approach has been adopted and is now permitting the development of the CCISs to advance. This paper outlines the history of the problems and decisions which culminated in their definition.

INTRODUCTION

This paper provides a description of the lessons learned from the early stages of the two multilevel secure (MLS) CCIS projects. Included are the managerial and engineering decisions which have been taken to help ensure that the project will continue to move forward to completion and satisfy the requirement for B3 certifiability. The importance of demonstrating a sound trusted engineering methodology as well as the role of prototyping in trusted system development is discussed. Particular attention is given to the definition and development of the security documentation which is essential to support both the engineering aspects and the security certification needs of the projects.

BACKGROUND

A fixed price contract to build a high assurance (B3) CCIS for the Supreme Allied Commander Atlantic (SACLANT) was awarded to Contel Federal Systems in October 1984. A second fixed price contract to build a high assurance (B3) CCIS for the

Commander-in- Chief Iberian Atlantic Area (CINCIBERLANT) which has similar functional requirements was also awarded to Contel in October 1987. Both projects are one hundred per cent funded by NATO, and were initially managed independently. Each project had specified unique documentation standards, different engineering design methodologies and separate certification requirements. This duplication of effort soon proved to be extremely expensive and time consuming for both the Government and Contel. The operational needs of both systems were closely analyzed and with concessions being made by all parties the notion of a single system design emerged. Both projects have since adopted a unified security policy, a single set of security requirements, and have been placed under the direction of a single project management office. This joint project is entitled Alpha CCIS, and will be referenced hereinafter as the ACCIS.

The ACCIS is required to process automated messages received from multiple telecommunications lines; to maintain a myriad of databases which contain plain text formatted messages, parametric (record) data, and geographic representations; to provide the capability to create and release formal messages; and to retrieve and display formatted information from its databases.

The ACCIS will combine a suite of alphanumeric and graphical terminals, communication processors, central hosts, and database machines to form the hardware architecture. The system will be highly redundant in order to provide the continuous service requirements mandated by the performance specifications.

Woven into the ACCIS functional requirements is a dominating requirement that the system provide a specified level of computer and communications security. This pervasive requirement for security is principally defined in terms of the so called Orange Book [1]. The ACCIS must be certifiable to class B3 in accordance with the criteria established in the Orange Book. The basic requirement for a B3 system was formed by applying the guidance contained in the Yellow Book series [2]. The B3 requirement is augmented by stringent performance requirements which mandate a high assurance and responsive architecture. The Gemini Multiprocessing Secure Operating System (GEMSOS) developed by Gemini Computers Incorporated was proposed by Contel as the commercial off-the-shelf (COTS) system for the ACCIS Trusted Computing Base (TCB). As another high assurance TCB has subsequently become available (pre-endorsed), the COTS portion of the ACCIS TCB has been re-evaluated. Currently, the HFSI XTS-200 has been identified as the best choice for the ACCIS.

SECURITY POLICY DEVELOPMENT

The ACCIS Security Policy was developed by the Government with the cooperation of Contel. The ACCIS Security Policy contains the administrative, personnel, physical, emanations, communications, and processing security requirements. It is intended to be used both as an operational policy document and as a definition of the requirements for the TCB. The development of the policy was a unique process in that the contractor was reviewing a government originated document for accuracy of content. These reviews were conducted first to determine the consistency, correctness and completeness of the policy and secondly to determine if specific aspects of the policy might pose implementation problems.

The first review process discovered inconsistencies in the policy, mostly due to semantics. However slight, the terminology differences highlighted the need for a

glossary of security terms. There were also policy statements that Contel felt were inconsistent with accepted interpretations of the Orange Book but had to remain because of operational needs. An example is the policy's requirement that allows a user to delete a file that is classified at a security level below the user's sign-on security level. There is a potential covert channel associated with this requirement, but it was determined by the Government that the operational need for the feature exceeded the threat of compromise posed by a covert channel.

The second level of review focused on evaluating the impact that implementing the policy would have upon the target TCB. A goal of the ACCIS design philosophy is to produce a system that does not require extensive modifications to the COTS TCB. The selected approach is to layer the additional ACCIS TCB functionality on top of the COTS TCB. The security policy was reviewed with this approach in mind, identifying those requirements that could fundamentally affect the COTS TCB. Additionally, the review produced suggestions for specific policy amendments and recommended design approaches that could be employed to implement the functionality and mitigate the impact on the COTS TCB. An example of a policy impact on the COTS TCB was in the area of auditing. The policy explicitly stated how audit data were to be protected. The method of protecting audit data used by the COTS TCB is equally as strong as the stated policy for audit data protection but it uses a different approach. The policy was modified to allow different approaches for protecting audit data, provided those approaches meet a minimum level of assurance.

Interrelationships between the ACCIS Security Policy, the ACCIS security requirements and TCB design became apparent. A detailed effort to align the ACCIS Security Policy and system security requirements was initiated.

CAPTURING SECURITY REQUIREMENTS

The ACCIS security requirements are a mixture of the standard computer security features specified by the Orange Book (e.g., mandatory access controls, discretionary access controls, identification and authentication, and audit). The ACCIS also has unique security features that are needed to support specific system applications. These features include a Two-Designated-Man-Rule, Trusted Turnover, and trusted message handling functions.

Specific security requirements were identified in the ACCIS Security Policy, the SACLANT Request For Proposals, the CINCIBERLANT Invitation For Bid, and Contel's proposals. Additional ADP security specifications were defined in specific NATO standards and guidelines referenced by these basic requirements' documents. In the process of identifying the security requirements, a basic dichotomy was discovered. The requirement for a B3 system imposed a structure dictated by the evaluation process that did not directly align with the standard systems engineering process. The basis of an Orange Book evaluation is the system's security policy. All certification evidence is derived to some extent from it. This is in contrast to a systems engineering approach which traces the system's development back to the requirements.

To resolve this difference, it was decided that the ACCIS Security Policy would contain all of the security requirements; that is, it would incorporate the security requirements from all of the sources mentioned above. This had the effect of blending the Orange Book evaluation and the systems engineering processes. The

ACCIS Security Policy statements and the ACCIS system security requirements are in complete correspondence.

CERTIFIABILITY, EVALUATION AND OVERSIGHT

Significant emphasis has been placed on the tasks of evaluation, certification and accreditation of the ACCIS. Initially, there existed very broad interpretations of what the term "certifiability" meant to the Government and to Contel. The differences between the evaluation process normally applied to trusted products and an evaluation of an application system that integrated a trusted product had to be sorted out. The issues that arose concerning the question of certifiability illustrate the conflict between developing and evaluating a B3 system and developing a useful system that satisfies its requirements and performs its mission. The initial discussions regarding the nature of the evaluation process revolved around issues concerning the latitude of the evaluators in defining the scope of the evaluation. The prevailing government position was that the ACCIS was a "system" and not a "product" and that a standard National Computer Security Center (NCSC) type evaluation was not sufficient. It was discussed whether the evaluators could mandate additional evaluation requirements regardless of contractual requirements. There were also questions as to how theoretical (vs. pragmatic) the evaluation should be.

The certification evidence document and Contract Data Requirement List (CDRL) item that brought all of these issues to the forefront was the Formal Model of the Security Policy (FMSP). Contel's FMSP was based upon the Bell and LaPadula Model, as are the formal models used by most evaluated TCBs. As with other evaluated TCBs, Contel did not intend to modify the model, but intended to provide an interpretation of how the system mapped into the model. Since other COTS TCB's were also modeled using Bell and LaPadula, this appeared a reasonable approach. However, the Government maintained that the ACCIS was a system and not a product, and Contel was directed to develop a FMSP that was specific to the ACCIS. Accordingly, Contel extensively modified the Bell and LaPadula Model to make it specific to the ACCIS Security Policy. Notions such as ownership and group membership were incorporated. The model's rules were replaced by ACCIS specific rules. The formal proofs were revised but the core theorems, properties and corollaries of the model were preserved, though in a modified form. However, the Government determined that the resultant model, as modified to be ACCIS-specific, was overly complicated and did not effectively represent the ACCIS security policy. Consequently, a joint effort between the Government and CONTEL is in process to rewrite the model without any pre-ordained dependencies on the traditional Bell and LaPadula framework. That effort led to a cooperative FMSP production activity.

Because there were no formal definitions of the certifiability process and the nature and content of the required certification evidence, the resolution of issues like these threatened to stymie the project. As a response, the Government and Contel agreed upon the necessity for a certification plan. To this end, the Government has developed a certification plan specific to the ACCIS. It details the tasks that must be performed by the system evaluators and indirectly, what is expected to be produced by Contel as certification evidence. The certification plan establishes orderings and dependencies between the certification evidence documents. The certification plan is the framework for understanding how the ACCIS can achieve B3 certifiability.

The security evaluator for the ACCIS, entitled Security Certification Technical Agent, is the United States Naval Research Laboratory (NRL). The Certification Authority for the SACLANT CCIS is the United States Commander-in-Chief Atlantic (USCINCLANT) and SACLANT is the accreditation authority. The Certification Authority for CINCIBERLANT is the Portuguese Autoridade Nacional de Seguranca (ANS) and CINCIBERLANT is the Accreditation Authority.

A Security Certification Working Group (SCWG) has been formed to oversee the security aspects of the developing ACCIS system and to reduce the risk of not achieving certification. The SCWG is chartered to provide guidance and resolve security issues as they arise. The membership of the SCWG includes representatives from the ADP security organizations of SACLANT and CINCIBERLANT, NRL, Mitre, Military Committee Communications and Information Systems Security and Evaluation Agency (SECAN), and Contel.

An extensive amount of time and resources will be expended by the Government in order to evaluate, certify, and accredit these systems. The availability of technical support required to evaluate a system of this complexity is very limited and expensive. Careful planning must be exercised to ensure that the contractual milestones and associated deliverables align closely with the certification plan to ensure that these valuable resources are effectively employed and ultimately assist rather than hinder the project's progress.

SECURITY DOCUMENTATION

While there existed substantial guidance on types of security documents required for a B3 system, little information pertaining to document content was available. Consequently, the ACCIS project did not define security-specific Data Item Descriptions (DID) for the security CDRLs. In many cases the only clear documentation specification was that the security documents were to be developed in accordance with the Orange Book. The Orange Book was never intended to be used for defining the content of contract deliverables or for determining the form of certification evidence. Considerable time was expended in SCWG meetings trying to establish agreement on what was expected for each security CDRL. While the security CDRLs closely mirrored the list of certification evidence documentation required by the Orange Book it was not always certain which document would contain specific evidence and if all of the evidence would be accounted for in the complete set of security CDRLs.

To alleviate these documentation problems, the Government and Contel developed DIDs for the ACCIS security CDRLs. A mapping of certification evidence to CDRLs was performed to ensure that all of the evidence would be produced. The NCSC "Guide to Understanding" series, especially the one for Design Documentation [3], was used to develop the DIDs. The Guide for Security-Relevant Acquisitions CDRL and DID Handbook [4] developed by the Headquarters Electronic Security Command, Air Force Computer Center at Kelly Air Force Base, Texas and the Trusted Computer System Security Requirements Guide for DoD Applications [5] developed by MITRE were also used extensively in determining the standards for security CDRLs. Finally, the certification plan was brought into alignment with the evidence mapping and associated DIDs.

Security DIDs had to be written or at least modified to accommodate the Security Policy Input, Formal Model of the Security Policy, Descriptive Top Level

Specification (DTLS), DTLS Implementation Mapping, Security Test and Evaluation Plan, Security Test Procedures, Security Test Descriptions, Security Test Reports, Covert Channel Analysis Report, Trusted Facility Manual and Security Features Users Guide. System documentation follows DOD-STD-2167A DIDs as tailored by the contract.

TCB DEVELOPMENT AND THE SYSTEMS ENGINEERING PROCESS

Both the Government and Contel recognized the importance of not separating the development of the ACCIS TCB and the remainder of the system. Consequently, the TCB development process was integrated into the systems engineering process when these processes were being defined in the Systems Engineering Management Plan (SEMP), the Software Development Plan (SDP), the Software Quality Assurance Plan (SQAP) and the Configuration Management Plan (CMP). Similarly, the security requirements are contained in the Systems Requirements Document (SRD) along with all other requirements.

When unique aspects of the security engineering process required special procedures, those procedures were detailed in the appropriate system plan. For example, the requirements for configuration control of the TCB are more rigorous than for the system as a whole and the CMP describes the additional TCB unique procedures. In the final analysis, requirements for development of the TCB are represented by good engineering practices which were adopted by the systems engineering process.

Contel's ACCIS systems engineering methodology closely adheres to the methodology described in U.S. Army Field Manual 770-78 System Engineering and the U.S. DoD Systems Management College's Systems Engineering Management Guide. The methodology is being augmented by techniques defined in Yourdon's Modern Structured Analysis and in Ward-Mellor's Structured Development for Real Time Systems.

System development will be evaluated at key contractual milestones as defined in and using the criteria of MIL-STD-1521A [6]. Engineering management follows the general guidelines of MIL-STD-1521A, DoD-STD-2167A [7], and MIL-STD-499A [8]. Configuration Management practices and procedures are based upon MIL-STD-480B [9], MIL-STD-483A [10], and NCSC-TG-006-88 [11].

SECURITY TESTING

An early topic at the SCWG meetings was security testing. It was agreed that Contel would perform penetration testing and testing in support of covert channel analysis. The Government could also perform those forms of testing at their option. Less clear was what constituted the testing of the system's security functionality.

The consensus that eventually evolved was that there existed two sets of security functional tests. The first set of tests would be on the system boundary. These tests would be standard, "black box" validation tests, conducted against system security requirements and performed during the factory and site acceptance test phases. These tests would be performed by Contel's testing organization as part of the suite of tests that demonstrate that all system requirements have been satisfied.

The second set of tests would be conducted against the TCB boundary using the DTLS as the basis for the tests. These tests would include the software drivers that exercise the interface into the TCB from the untrusted application environment. Conduct of these tests would be the responsibility of Contel's security engineering organization.

SECURITY RISK REDUCTION

Historically, the development of high assurance computer systems has been troubled by a myriad of technical problems that either delay the project or cause it to be terminated. Both the complexity of the engineering and the complexity of the evaluation processes can contribute to the development problems. In an attempt to identify risk areas early in the development process, the Government devised a demonstration to test the systems engineering process defined by Contel and to assess the complexities of evaluating the product.

To demonstrate their engineering process, Contel was asked to develop a representative "slice" of the TCB, exercising all phases of the engineering process, commencing with requirements definition and carrying the development through to its detailed design. The requirements that were selected involved the processing of messages received from the MLS ACP-127 communications lines. Following the procedures defined in the SEMP, SDP, CMP, and SQAP, Contel produced the TCB slice with all of its required documentation. The results of this exercise significantly aided the refinement of the ACCIS engineering process and provided the Government and Contel with a clear understanding of the complexity associated with developing and evaluating a high assurance system.

TCB design issues are also being addressed early in the development process by prototyping. The purpose of this prototyping effort is to identify solutions to difficult TCB design issues so those solutions can direct the design of the actual ACCIS TCB. This will help reduce the risk that the system may fail to be certifiable at the B3 level. The prototyping effort will be formally documented in the Interface Requirements Specification, Software Requirements Specification and System Design Document CDRs. The Government and Contel will be able to explore design alternatives, seeking solutions that provide the required functionality without introducing unnecessary COTS TCB modifications or ACCIS TCB complexities. The prototype will also identify early in the design process any operational impacts that may occur by implementing some of the security features as they are currently defined.

The prototyping will include the ACCIS specific security requirements, including the Two-Designated-Man-Rule and Trusted Turnover. The requirement for a trusted data base management system will also be prototyped.

STATUS

Project management for both systems has been delegated to a joint project office located at SACLANT Headquarters, Norfolk, Virginia. In addition to the obvious benefits of centralized management, the logistics support problems inherent in the geographic separation between the European and U.S. sites were removed. The joint ACCIS Security Policy and associated security requirements have been aligned and the development of an ACCIS formal model is in process. A certification plan has been produced and agreed by all parties. The role of the

Security Certification Technical Agent and the Security Engineering Support Agent have been identified and are in place.

Host Nation responsibilities for their respective projects remain autonomous. Contract modifications are in process to unify contract milestones and deliverables. The System Requirements Review phase concluded 10 October 1990. System Design Review is scheduled for November 1991, Critical Design Review is scheduled for October 1992 and Initial Operating Capability is slated for October 1993.

CONCLUSION

The development and evaluation of complex, secure command and control systems is only now being better understood. All of the tools and experience necessary are not yet available. There is a dearth of evaluated high assurance TCBs available to use as a basis for secure systems and there are few in the evaluation pipeline that will ultimately achieve endorsement. Performance requirements can further reduce the number of suitable TCBs for a given system. What should be clear by now is that it simply is not possible at this time to acquire secure systems "off-the-shelf" that satisfy all the requirements of real systems.

The absence of standard security DIDs, certification plans, and secure systems development processes also hampers the ability to define, develop and evaluate secure systems. The Government and the developer must share the same understanding of how security is to be integrated into the system and how that security will be evaluated.

Because the development of secure systems currently involves some uncertainty, strong management support is required. Management must set a course through the sparsely defined territory of secure systems development, identifying deficiencies in the process and finding ways to correct them. A rigorous project management structure is required to ensure that the possibly conflicting interests of building a system that satisfies a critical military mission and maintains a demonstrably high level of security do not bring the development to a standstill.

The ACCIS project can be viewed as a useful case study of the development of a secure, complex military system. It suffered from the lack of tools, products and experience. Fortunately, the mutual desire of both the Government and Contel to complete this project has brought it through the most difficult period.

References

- [1] Department of Defense, Trusted Computer Systems Evaluation Criteria, DOD 5200.28 STD, December 1985
- [2] Guidance For Applying The Department of Defense Trusted Computer System Evaluation Criteria In Specific Environments, CSC-STD-003-85 and CSC-STD-004-85, 25 June 1985
- [3] A Guide To Understanding Design Documentation In Trusted Systems, NCSC-TG-007, Version-1, 2 October 1988

- [4] Guide For Security Relevant Acquisitions CDRL and DID Handbook, Volumes 1 and 2, 1 May 1989**
- [5] Mitre Trusted Computer System Security Requirements Guide for DOD Applications (Draft), 18 March 1988**
- [6] Technical Review and Audits for System Equipments and Computer Software, MIL-STD-1521A**
- [7] Department of Defense System Software Development, DOD-2167A**
- [8] Engineering Management For Total System Development, MIL-STD-499A**
- [9] Configuration Control, Engineering Changes, Deviations and Waivers, MIL-STD-480B**
- [10] Configuration Management Practices for System Equipment and Computer Software, MIL-STD-483A**
- [11] A Guide to Understanding Configuration Management In Trusted Systems, NCSC-TG-006, Version-1, 28 March 1988**

CONTRACTORS AND COMPUTER SECURITY - AWARENESS, EDUCATION, AND PERFORMANCE

Ronald G. Brunner
Ronald G. Brunner and Associates
2 Jasmine Court
Rockville, Maryland 20853
(301) 929-1518

Preface

This paper addresses the dual problems of monitoring a contractor's performance and providing adequate computer security within the Federal government environment. Contractors perform many of the government's computer functions, therefore security must be a part of their services and products.

How does the government know that the contractors' are performing the computer security function in an acceptable manner, and that they have the proper level of awareness, commitment, and skills to provide this security? Guidance for determining a contractor's experience, and assuring performance, as they relate to computer security, are contained in this paper.

The paper is intended for use by computer security officers, computer resources management and technical staffs, and contracting officers, as well as by educators who are responsible for training the government personnel. Although the paper is directed toward those individuals within the Federal government, most of what is stated would also be of value to individuals who are working for a commercial organization.

The contents of the paper is based on the author's thirty years of experience working as a computer manager, technician, and educator within both the Federal government and commercial environments.

I. BACKGROUND

A. Contractors in the Federal Computer Environment

During the past thirty years, computer technology, and the way in which in it is used, has changed dramatically. Automated Data Processing (ADP) at one time meant transferring data from written documents to punched cards, for processing by large computer systems in secure data centers. The results of the processing were volumes of printed reports, difficult to handle and use. Any change in the requirements meant days, if not weeks, of work by computer programmer/analysts. Manual activities supported the processing, and when the computer failed, manual work could replace its function. Computers were not easy to use, and they were only critical for a small number of a Federal agency's functions.

Today, terminology has changed. ADP has been replaced by such terms as Management Information Systems (MIS), Information Resources Management (IRM), Federal Information Processing (FIP), and many others which are too numerous to list here. Today data is entered into a computer by optical scanning, voice recognition, remote sensing, data communications, and a variety of other methods. Computers can be

large, still housed in a data center, or small, sitting in a person's lap. The data which they process may be shown in printed form, graphically, or be an electronic signal which is sent to a different location to perform another task. Computer programs can be modified easily, sometimes even by non-technical personnel.

The importance of using computer technology has changed also. Today computer technology relates to the sharing of information by the use of local area networks, the creation of documents via word processing, the electronic transfer of funds, the instantaneous location of a unique document, fax transmissions, and a wide variety of tasks and functions. These tasks are critical to the successful completion of an agency's mission. They can no longer be performed manually. The use of computers is no longer optional. This paper will use the term computer resources in the broadest sense, as it relates to all of the technologies and terminologies described above. It relates to any data, information, hardware, software, system, facility, or communications function, where technology is utilized in the performance of an agency's function.

The Federal government is one of the largest users, if not the largest user, of computer technology in the world. It operates thousands of data centers and communications networks, and hundreds of thousands of personal computers. Many of these computer functions are performed by civil service employees, but an increasing number are performed by contractors. Many Federal organizations use contractors to perform the majority of their computer related work, with the civil servants only monitoring the contractors' functions. There are thousands of contracting firms, large and small, whose only source of business is providing computer services to the Federal government. The Office of Management and Budget reports that for FY-1990, the Federal government spent over ten billion dollars for these services. The hardware, software, communications networks, and other computer products which the Federal government uses on a daily basis are also produced by the contractor community. The Federal government, in general, relies completely on the private sector for the computer related products which it requires, and could not function without the products which the private sector provides to it. For FY1990, the Office of Management and Budget reports that nearly five billion dollars were spent on these products.

B. Federal Computer Security Requirements

As computer resources become more of a critical component within the government's work processes, measures have to be taken to assure that these resources are always available for use, for without them, organizations, and even entire government agencies, could stop functioning. Protection has to be provided to guard the government's computer resources from (1) adverse actions such as acts of nature and accidents, (2) improper actions such as malicious and illegal acts, and (3) undesirable occurrences such as system failures due to design limitations and inadequate testing. The integrity, confidentiality, and access to all of the government's computer resources must be protected. In this paper, the term computer security is meant to include protection against all threats to all of these resources.

To assure that the computer resources are adequately protected, Congress created the Computer Security Act of 1987, as well as numerous other laws. Federal agencies with oversight responsibility, such as the Office of Management and Budget, and the General Services Administration, have published numerous regulations which all agencies must follow regarding computer security. Individual agencies have created

their own rules. Organizations are legally bound to provide security for their computer resources.

The laws and regulations require organizations to be concerned about computer security, and to implement computer security programs. Many government managers have also realized, from a practical viewpoint, that their computer resources are very critical to the performance of their functions, and as a result have taken prudent actions to protect those functions.

C. Contractors' Role in Computer Security

Two major trends within the Federal government have now converged, the use of contractors' products and services to perform the governments' expanding and critical computer related functions, and the expanding concern about the security of the government's computer resources. Contractors therefore must not only be concerned about computer security, they must take an active role in protecting the government's computer resources.

II. THE CURRENT STATE OF COMPUTER SECURITY

A. Contractors' Involvement With Security

Are all contractors fully aware of all of the laws and regulations which apply to computer security, of all of the threats which exist which can adversely affect the government's computer resources, and of the impact to the government if those resources are not available? Most of us will agree that "all" contractors do not have this awareness, and some government employees will state that "many" contractors do not have this awareness.

While this author has not conducted, nor knows of anyone conducting, a formal survey as to the degree of contractor awareness concerning computer security, the author has held dozens of discussions with computer security officers, contracting officers, technical monitors, trainers, and managers within the government, as well as many contractor personnel. These discussions have lead the author to believe that "many" contractors do not have the proper level of security awareness. The fact that you are reading this paper, could be interpreted as indicating that you too are concerned about potential threats to your computer resources, which your contractor is doing little to protect.

Before a contractor can be expected to performance a task, the contractor must be aware of the task. Unfortunately today, "many" contractors are not aware of the need for adequate computer security. They are not aware because their contract with the government does not address it, or because of their lack of understanding about computer security. Computer security is a sleeping giant which is very easy for the contractor, and government, to ignore until a disaster occurs.

Are all contractors fully committed to computer security? The same survey discussed above indicates that there is also a lack of commitment by "many" contractors in implementing a computer security program, or in implementing good security features in their products. This lack of commitment can be the result of a lack of awareness, or it can be that computer security conflicts with the contractors' prime objective, which is to make a profit, or with their client's objectives, which do not include security. If the contract does not spell out what the contractor's responsibilities are regarding computer security, or if the contractor's client is not concerned about security, is the contractor going to do anything about security? Probably not.

Many contractors have the problem of not possessing the necessary skills to implement an adequate computer security program, or to include adequate security features in their products. This is not a problem which is unique to contractors. In general, there appears to be a shortage of computer security individuals who are knowledgeable in both the theoretical and practical aspects of security. This condition is known to anyone who has attempted to recruit an experienced computer security technician.

There are many reasons for the shortage, but they include an ever changing technology, a lack of interest by the computer industry in computer security, and a lack of good, practical, computer security educational programs. Many government agencies and private organizations do offer educational courses on computer security, but the quality of these classes varies greatly, and only a small number of courses are offered. In addition, training funds are usually in short supply. Too often, training on how to develop new systems, install LAN's, or use a state-of-the-art technique takes precedence over security training.

B. Government's Involvement With Computer Security

Before all problems relating to the lack of computer security are blamed on the contractors, the government has to review its own environment. Some organizations within the Federal government have excellent computer security programs, some have adequate security programs, but some just provide "lip service" to computer security. This observation is based on the author's interactions with dozens of government organizations. The lack of awareness and commitment by Federal employees applies to both civil servant managers, computer technicians, and contracting officers.

Many agencies today have serious shortages of civil servant employees. Therefore government employees often state that it is difficult, if not impossible, to monitor on a regular basis what the contractor is doing, as it relates to computer security. Some government contracting officer's technical representatives (COTR) say that they have all to do to assure that the contractor is delivering its products on time, or that it is responding to all of the users' problems, without monitoring what the contractor is doing about computer security.

Even if the COTR has the time to monitor the contractor performance in the computer security area, how do they determine that the contractor's computer security awareness, commitment, and skills are adequate? In too many cases, the COTR does not know enough about computer security to question the contractor about it.

C. Contracts

What work a contractor does, or does not, do is defined in the contract which exists between the government and the contractor. The contractor is not going to do anything not contained in the contract because the contractor will not get paid for it. In some cases, if the contractor performs work not contained in the contract, it could even be considered as being an illegal act.

Computer security has not been adequately addressed in many of the government contracts the author has reviewed. Many times this is because the technical security requirements have not been determined, other times because the necessary security clauses have not been kept current, and still other times, because there are no funds available to include any security features.

III. IMPROVING COMPUTER SECURITY

A. Overview

You are very concerned about the security of your computer resources. You do not believe that your contractor is doing an adequate job in protecting those resources, resources which must be operational for your organization to fulfill its mission. How do you get the contractor to be more responsive and to protect the computer resources, and maybe even your job? Perhaps, you do not even know whether your contractor is performing in a satisfactory manner or not, or whether the contractor has the skills and motivation to do the job. Or maybe, both you and your contractor know exactly what should be done, but you can not convince your management to authorize the necessary security program, or there are no funds to perform the work. The remainder of this paper will provide guidance to anyone who is concerned about these conditions, or has to educate government personnel, such as contractor monitors, about computer security.

B. Government's Awareness

The first thing any government organization must do to develop and implement an adequate computer security program, is to make the government's managers and technical staff aware of the legal and management needs for computer security, and to educate them as to what computer security really means. You can not place requirements on your contractor, if you and your management do not understand the requirements, or if there are no funds to perform the work. You can not tell your contractor that computer security is important, if you and/or your management do not agree. You can not monitor a contractor's technical performance if you do not understand what the results of that performance should be.

Awareness and education of the government personnel concerning computer security are necessary before you can get the contractor involved. A contractor who is knowledgeable in the area of computer security, can assist you in "selling" security to your management, and in the education of your staff regarding the technical aspects of computer security. The contractor can not be the driving force behind computer security, it must be the government. Step one then in any computer security program is to assure that you, your management, and technical staff are knowledgeable about the legal, management, and technical requirements for computer security, and are committed to a reasonable and adequate security program.

How do you accomplish that? Lengthy papers have been devoted to informing and educating people about computer security, and to completely address the issue here would not be practical. Briefly though, the legal and management need for computer security must be made known to all. It is the process of awareness. You have to know that if you do not have an adequate computer security program, you are not complying with the law. You have to be aware that your organization may be placed at great risk if you do not have an adequate computer security program. Your computer resources could be vulnerable to a wide variety of threats, which could have a significant adverse effect on the mission of your organization.

Those threats not only include computer viruses and white collar criminals, but also the results of unusual weather conditions, the failure of a sprinkler system, a labor dispute, the detection of a hazardous material in your physical environment, a design error in your computer system, a feature in a product which does not work as promised, and thousands of other actions and events which occur in computer environments throughout the world on a regular basis.

After awareness has been established, a program of computer security education must take place within the government. The government does not have to understand the details of how a specific virus works, or what are the different types of data encryption, or how an uninterruptible power system functions. They do have to be able to recognize what are the potential threats to their resources, where and how they are vulnerable to those threats, what will be the effect on their operations if the threat turns into an adverse action, what technology and management practices exist to counter those vulnerabilities and adverse actions, and whether the cost for this protection can be justified. The question must be addressed as to how much computer security, and at what cost, is appropriate. This is risk management. A contractor can assist you in gathering all of the required information and performing the necessary analyses, but the final decision as to which risk is not acceptable and therefore must be protected against, and which risk is acceptable, for which no protection will be provided, must be made by the government.

C. Expanding the Contract

After it has been determined what level of computer security is required and must be provided by the contractor, it must be addressed in a contract. If your computer security requirements are not described in your solicitation, and not detailed in the resulting contract, those requirements are not going to be satisfied by your contractor. You can not assume that the security features or commitment you desire are going to be provided, nor that the contractor will want to, or be able to, provide features or work not contained in the contract. After a contract has been created, it can be modified to add security requirements to it, but that is usually costly, sometimes difficult, and always politically a problem. You must ask your boss for more money than you had planned for, and he/she asks why?

Your security requirements must be described in detail in your solicitation. Any specific security requirements which are unique to your environment must be inserted. General requirements can be obtained from your agency's procurement "boiler plate", or from prior solicitations, but usually that will not provide all of the protection you need.

Your solicitation must address your overall needs, and what security features and services those needs require. Areas to be included are: how is security controlled by the hardware and software; what security features do you require in the products and systems; how is access to the resources controlled; what user and technical documentation is needed; what are the legal considerations; how are communications to be protected; what is the importance, confidentiality, and criticality of your data; what clearances and skills must the personnel possess; how will security training and awareness be handled; what are the security needs for the facilities; and how will contract administration be conducted. Assistance in placing the proper requirements in your solicitation is provided by the National Institute of Standards and Technology, as well as the General Services Administration.

In addition to your security requirements, you must describe your environment to your contractor. If a risk analysis just determined that your installation has serious security problems, you must tell the contractor. If you do not, the contractor may not address those issues, and costs, in the proposal. If a contingency plan exists to be used in the event of a computer failure, the contractor must know about the plan. Anything which can affect the security and the functioning of your computer resources must be told during the solicitation phase to your potential contractors.

Many times, all of the security requirements which you desire the contractor to respond to, can not be identified at contract award, or if they have been identified, there are no funds to support them. It is therefore wise to include in any contract, options for the contractor to perform additional security activities, if the requirement and/or funds arise. Options do not commit the government to having the contractor perform the work, but they can save significant time and procurement effort if the government does require the work to be performed.

D. Responsibilities

The solicitation, and the resulting contract, must make it clear as to the responsibilities of the government and the contractor. Who trains the contractor, pays for the training, determines the level of training required, and determines when it has been successfully completed? Who creates the risk assessment, screens the contractor's personnel, and controls the passwords to access the computer system? The who, and how, and what, and when, for all activities must be included. The specific areas to be considered included: computer security planning, risk determination and analysis, identification of sensitive systems, contingency plans, training, procurement of additional security related products and services, personnel requirements, determination of costs and available funding, and controlling and monitoring the contractor's performance.

Unless these responsibilities are clearly defined, several adverse actions will occur. First, the government and the contractor will be arguing throughout the term of the contract as to who is doing what and who is going to pay for it, and second, the even more important, required functions may not be performed.

General guidance is that the government is responsible for overall planning and control, conducting awareness training, the identification of sensitive systems, and funding. The contractor would perform all of the required analyses, do the detailed training, and be responsible for the day by day security actions. The exact breakdown of responsibilities will vary from contract to contract. It is very important that all responsibilities be identified, and assigned to either the government, or the contractor, or even a different contractor.

E. Determining The Contractors Awareness, Commitment, and Skills

You have defined in your contract what support you require, but how do you know if the contractor has the capabilities to provide that support? The contractor's proposal states that they have the knowledge and experience, or their product performs that function, but how can you be sure? Techniques which are used include: (1) reviewing their past performance, (2) assessing their plans for future performance, (3) interviewing their proposed personnel, and (4) seeing a demonstration of their proposed products.

Past performance can be determined by checking with organizations who the contractor has supported in the past. The contractor says that they have performed these security activities for this client. Check with the client to determine if that is true, and whether the client was fully satisfied with the contractor's performance. Check with many prior clients, and with both the technical and procurement monitors. Determine if the contractor has the skills, commitment, and record for doing what they say they can do. If the contractor is proposing personnel, check the references on the resumes.

In addition to past performance, you want to know what the contractor plans to do for you. Require the contractor, in their proposal, or response to your task order, to provide to you a detailed plan as to how the security work will be accomplished. The plan should address exactly what is going to be accomplished and how. What is the level of staffing proposed, and what are the qualifications of the staff? What is the detailed schedule with interim milestones? What are the responsibilities of the contractor and the government? What are the deliverables, and how will the government know that the work has been performed in a satisfactory manner? This plan must be agreed to by the government before the contract is awarded, or the task signed off by the government. It should provide to the government a feeling that the contractor understands the security problem, has the talent to attack the problem, and possesses a plan to resolve the problem.

A part of a government contract, should be the contractor's security training plan. The difficult question is, what is the correct level of training? There is no correct answer, for the answer will depend on the required level of security, the funding which is available, the sensitivity of the application, and many other factors. You should assure though that the training proposed corresponds to all of those factors.

At the time a contractor submits a proposal to the government, it is proper and advantageous for the government to interview some, if not all, of the contractor's proposed personnel. This will assure that they are committed to working on your project, and that they possess the necessary skills to support your security program. If you do not have the necessary skills to adequately interview them, obtain those skills from elsewhere in your agency, or even hire another contractor to assist you. Do not assume that because the proposed resume says that the person has the required experience, that the person actually does have the experience.

Benchmarks, or product demonstrations, can be used to prove to you that the proposed product or technique, actually accomplishes what it is suppose to accomplish. This "hands on" viewing of what is being proposed can be very detailed, and require many hours of time by both the government and contractor personnel, or it can be just a brief demonstration at another client's site. Some contractor's products are formally approved, or accepted, by some government agencies, or the commercial marketplace. This approval or acceptance could be used instead of having your own demonstration, but be sure that the environment in which the product was approved or accepted is exactly the same as yours. Demonstrations do not only apply only to products, they also apply to techniques. If the contractor proposes to conduct their own in-house training program, it is appropriate for government personnel to sit in on those training sessions to assure that they will meet the government's specific requirements. If the contractor is to develop a contingency plan, the government may review prior contingency plans the contractor has created to assure that the approach is acceptable. Do not accept promises from the contractor. Rather, view with your own eyes what you are going to receive from the contractor, and determine whether it meets the contractual requirements.

F. Monitoring Performance

You have now been convinced, at least in theory, that your contractor has the skills, experience, commitment, and plan to protect your computer resources. How you do know that the contractor is doing what the contractor has promised to do. How do you monitor and control the contractor? Especially when your staff tells you that they do not have the time to perform this monitoring function. First, the question of priorities must be addressed. If computer security is really important to your organization, and if you really desire to control what your contractor is doing, you

will have to locate resources to monitor the contract. They do not have to be full time resources, but they do have to be at a level which relates directly with the importance of what the contractor is doing.

The best way to monitor your contractor, at any level, is by using multiple control points. What is meant by that, is that you obtain, on a regular basis, data concerning your contractors performance not from one source, but from multiple sources. You then compare all of the data to assure that it is consistent. If it is not, you have a problem. For example, you receive from your contractor status reports telling you of the good things the contractor is doing for you. At the same frequency, you should also receive similar data from your user community, your operations personnel, even from other contractors concerning the contractors performance. Is all of the information consistent? You receive written reports concerning your contractors performance. Does verbal inquiries agree with the written data? In walking around the entire environment being protected, does your visual inspection and discussions agree with the written data? If it does not compare, you better start asking many more questions.

For example, how do you know that the contractor's employees are receiving the security training which was proposed. First, obtain from the contractor detailed information concerning which contractor employees went to class. Since the government is paying, either directly or indirectly, for the training this is an appropriate request. Contact the trainer yourself to obtain feedback as to who was trained, and what was their performance in the class. Talk to government employees who may have been in the same class. Submit to the contractor a simple task concerning the material which was covered in the class, to be completed by the students who just completed the class. Have the results of the task reviewed by competent security sources. Does all of the information you have gathered agree, or not?

Remember though, that any information, especially written, which you require from the contractor should have been identified in the contract. This does not mean that every report has to be listed, but categories and frequencies of reports should be addressed. You can only perform the contact monitoring specified in your contract, and/or permitted by government laws and regulations.

G. Contract Administration

Most interactions between you and contractors are to be in writing, and flow through your contracting officer, or at least the contracting officer's technical representative. Too often that does not occur. Verbal direction, usually illegal, is provided to the contractor by a variety of government employees. The result is confusion on the part of the contractor because they do not know who to respond to, frustration by the government's technical staff because the required work is not being performed, and anger by the procurement officer because the laws and contract are not being followed. The government's procurement laws and regulations must be followed in administrating any contract, not just because they are the laws, but because good management practices require that they be followed. In addition, there must be a good contract "audit trail", that is a file of documents showing what the contractor was to do and what they actually accomplished. Letters of commendation as well as complaints must be included. Do not assume that the contractor will respond to your verbal requests or concerns. Put it in writing, send a copy to the contracting officer, and place it in the file.

Either because you are doing a good job in monitoring your contractor, or because a disaster just occurred, you determine that your contractor is not accomplishing what

you thought the contractor was accomplishing. What do you do now? First, you have to determine if the function is addressed in the contract or not. If it is not in the contract, you can not force the contractor to do something which the contractor is not legally bound to do. If your legal and procurement staff tell you the contract is not clear, "the monkey is on your back". Your only real course of action, is to have the contracting officer issue a modification to the contract, or to obtain other resources to get the job done. Hopefully you will have learned from your mistake, and will write a better solicitation, or task order, the next time.

Suppose though, the contract is complete and clear, and your contractor is just not performing. Initiating the disputes, default, and termination clauses in the contract is normally not the way to go. The contractor, your contracting officer, and your management, will all get mad at you, but the work still is not getting done. Instead attempt to determine what is the problem. Most contractor problems have been caused by incorrect, incomplete, or erroneous communications. Therefore, when a problem does arise, talk to your contractor project manager, vice president, or the owner of the company, and determine what the problem is and how it can be corrected. The contractor usually is just as interested as you are in solving the problem. Contractors can obtain additional contracts only if their prior performance has been acceptable. The last thing they desire is to have an unhappy client.

It is possible though that you can encounter the contractor that can not or will not perform. The task then is one of terminating the contract, and obtaining a new contractor. Remember, that if your contract is clear and complete, and you have good administration records, there should be no question concerning the contractor's lack of performance. These documents will permit you to terminate the contract in a shorter time, and with less frustration, than if things are not documented.

You have placed the security of your computer resources in the hands of your contractor, but computer security is still the government's responsibility. You must work together with your contractor to attack and resolve your security concerns. In this way, the resolution of most problems will occur in the shortest of time, the protection of the computer resources will be maximized, and everyone will benefit. Computer security is a sleeping giant. You are going to need all of the help you can get, to properly protect all of your computer resources, from those bad things, which are guaranteed to happen to you.

COVERT CHANNEL ANALYSIS PLANNING FOR LARGE SYSTEMS

Lee Badger
Trusted Information Systems, Inc.
3060 Washington Road (Rt. 97)
Glenwood, MD 21738

Abstract

Covert channel analysis is a challenging task, particularly when performed during the development of a large system. Some elements of covert channel analysis, such as timing channel identification and reduction, require techniques currently beyond the state of the art. Performing a useful covert channel analysis during development requires a careful balancing of costs and assurance, and a careful selection of currently available techniques. While it is possible for new research to assist in the covert channel analysis of large systems, developers cannot plan on breakthroughs. This paper discusses available techniques, their limitations and tradeoffs, and makes recommendations for performing covert channel analysis.¹

Keywords: Assurance and Analytic Techniques, Conducting Security Evaluations.

Introduction

Covert channel analysis (CCA) is a process of identifying and analysing information flows in a security policy model, system specification, or system implementation. CCA is required to satisfy the TCSEC [1] B2 and higher evaluation class requirements and also the ITSEC [9] E4 and higher assurance levels. CCA may be performed either informally or formally. In general, CCA has 3 distinct components: 1) identification of covert channels, 2) estimation of their capacities, and 3) reduction of capacities. An additional, implicit component of CCA, is to gain assurance that each of the three tasks are correctly performed.

Performing a credible CCA, successfully and at reasonable cost, during a large (i.e., complex) system's development is a challenging task. In the context of the Trusted Mach system currently under development at Trusted Information Systems, a CCA plan has been evolved that balances concerns over assurance, cost, and feasibility. This paper first provides definitions and summarizes available techniques. It then compares the techniques, and presents an approach for performing CCA during system development.

Definitions

Generally, covert channels make use of system characteristics, such as error return codes or global identifiers, that are not normally thought of as containers of information but that reveal the state of shared resources (hence the word "covert"). In contrast, information flows that occur between system "objects" as a result of using system primitives in the intended way can normally be thought of as "overt channels." A *covert channel* is usually defined to be a "communications channel that allows a process to transfer information in a manner that violates the system's security policy."¹ A system security policy (for B2 and greater systems) should be completely stated in its FSPM (Formal Security Policy Model^[1]).² If the FSPM includes an information flow policy, such as noninterference [6] or nondeducibility [18], this definition is accurate. For access control FSPMs (e.g., [3]), however, the system security policy makes no statement about information

¹This work was supported by DARPA/ISTO Contract MDA97-90-C-0027.

²The construction of an FSPM that accurately reflects external security requirements is beyond the scope of this paper; a valid FSPM is assumed.

flow, and the "intent" of the system security policy must be inferred from the properties of the defined secure state. For example, the *ss*-property and ***-property of the Bell and LaPadula FSPM imply an information flow rule of "no flow down." For these systems, the purpose of CCA is to gain assurance that information may not flow contrary to the *intent* of the system's security policy. It should be noted that this definition is very broad, including as covert channels all mechanisms that reveal failures by the TCB (Trusted Computing Base) to satisfy the FSPM.

Although not required by the definition of a covert channel, the general paradigm of covert channel exploitation involves one or more sending subjects that have access to sensitive information, and one or more receiving subjects that have lower access to sensitive information. Under the assumption that components of the TCB do not intentionally compromise information, there must at least exist a receiver that is untrusted. If there is no untrusted sender, the receiver's actions amount to spying on events going on in the TCB, which satisfies the definition above, but is a much smaller threat because there can be no cooperation between sender and receiver, and because presumably the TCB is using care in its handling of information. Most of the literature on covert channels focuses on the more dangerous case, where both sender and receiver are untrusted subjects and cooperate. In this case, the sending subject must be executing a trojan horse program that is using the access rights of a highly cleared user. Although there may be many senders and receivers to exploit a given channel, the number is an implementation detail of the exploitation; this paper refers to "the sender" and "the receiver."

Typically (and in the TCSEC), covert channels are divided into two classes: storage channels and timing channels. A *storage channel* is a covert channel in which the transmission of information involves the alteration and observation of storage locations in the TCB. A *timing channel* is a covert channel in which the transmission of information involves the manipulation, by the sender, of the length of time that the receiver requires to perform some operation. For a timing channel to exist, the receiver must have access to a timing reference in order to measure the time required. Some channels are difficult to categorise as either timing or storage[23]. For example, the following channel would appear to satisfy both definitions: a sender positions a disk's arm to the middle or outer track of a disk by performing I/O to files that are known to reside in those places; the receiver performs I/O to an inner track, measuring the delay in servicing the I/O request. The position of the arm is internal state (storage), and the receiver deduces that information using timing properties.

Because covert channel exploitations bring about the disclosure of information, a definition of information is necessary. Although the intuitive definition of information as "bits" is useful, a more formal foundation is required to calculate the capacity of a channel, that is, the rate at which information flows through it. Shannon's definition [17] is widely accepted as the proper foundation. Very informally stated, information is the amount of "surprise" that the receiver experiences when learning the value of a symbol received. As an example, a receiver that receives one of n symbols (all equally likely) learns more than a receiver that receives one of m symbols (all equally likely) when $n > m$. The amount of information received depends on the probabilities of the symbols. If one of the n symbols is very likely, so that the rest are very unlikely, then receiving one of the rest is relatively "surprising," and more information is received than would be the case if the probabilities were equal. Because the overhead of sending different symbols may vary dramatically, covert channel exploitations may substantially increase channel capacity through the use of coding. Using coding, a sender can change the probability distribution of symbols received by the receiver by encoding expensive symbols as sequences of less expensive symbols.

TCSEC B3 Requirements

The TCSEC requires a system developer to conduct a thorough search for covert channels (storage channels only at B2; timing channels also at B3 and A1), and to determine channel capacities for identified channels using either actual measurement or engineering estimation. In the recent Trusted Xenix B2 evaluation, the evaluation team rejected the use of actual measurement because there could be no guarantee that the strategies and code used to drive the channel were the most efficient possible. Analytic techniques must therefore be used for measurement (note: just as measurement is prone to underestimation, analytic techniques are

prone to overestimation). The TCSEC criteria refers the reader to the covert channel guideline section of the TCSEC for guidance on both acceptable capacity and auditing. The guideline asserts that all channels with capacities above 1 bit per second can be audited without adversely affecting system performance, and therefore that such channels should be audited.³ Additionally, it recommends auditing of channels with capacities above 1/10 bits per second where possible. Since the guideline is not part of the criteria, however, it is subject to modification by precedent. During the Trusted Xenix B2 evaluation, the team found the following channel capacity and auditing categorisation acceptable:

<i>Capacity</i>	<i>Action</i>
< 1	no concern
1 – 10	document, audit if possible
10 – 100	if not possible to reduce, audit and document
> 100	not generally acceptable

ITSEC Requirements

Development of ITSEC rated systems has 4 phases: requirements, architectural design, detailed design, and implementation. At the E4 assurance level, CCA is required in the detailed design phase. At E5 and E6, CCA is required both during the detailed design and implementation phases.

In the detailed design phase, a specification using “some form of rigorous approach and notation” is required. The specification is required to provide a DTLs and to identify all security mechanisms. A “design vulnerability analysis” must be conducted on the specification to determine how security may be subverted on a system configured in a specific way by a security administrator. This analysis must identify covert channels. It is required that the exploitation of covert channels be auditable. In the implementation phase, an “implementation vulnerability analysis,” for a given configuration, must identify covert channels.

The ITSEC targets covert channel analysis at specific configurations, which opens the possibility of supporting both covert channel analysed configurations and other, perhaps more useful, configurations. In general, the ITSEC does not appear mature in its treatment of covert channels. First, a definition is not given, although the reader might be justified in using the TCSEC definition. Second, the requirement that all channels be auditable is probably not technically feasible.

Channel Identification

There is currently no known technique for identifying all covert channels in an implementation. Relatively high confidence can be gained that storage channels have been eliminated from specifications [11] Unfortunately, implementation details not present in an interface specification may introduce new channels. For a complete, rigorous treatment of storage channels, it is probably necessary to combine analysis of interface specifications with code level (or very low level specification) checking to validate the interface analysis. Although at least one informal methodology exists for searching for timing channels (summarised below), identification of timing channels remains ad hoc. Most of these methods focus on finding “potential” channels; informal techniques must then be used to determine if the channels can actually be used. This section presents three different approaches to finding storage channels, and one approach for finding timing channels.

Shared Resource Matrix Methodology

The shared resource matrix (SRM) methodology of Kemmerer [10] focuses on identifying the shared resources whose “attributes” can be used for covert channel exploitation, and the system primitives that must be used to manipulate the attributes. As defined by Kemmerer, a shared resource is “any object or collection of objects that may be referenced or modified by more than one process”⁴. The definition of the storage

³The auditing of some timing channels, if attempted, would severely degrade system performance.

⁴Kemmerer assumes that subjects are processes.

objects in the system's security policy model determines a subset of objects about which attributes may exist for covert manipulation. For example, a file or terminal may be a shared resource that has a size or lock attribute that is subject to manipulation. Subjects may communicate by changing the size or setting the lock. Whenever multiple subjects share a cpu, an additional shared attribute, the response time, is also available.

Kemmerer gives the following minimum criteria for the presence of a storage channel:

1. Sending and receiving subjects have access to an attribute.
2. The sender can cause the attribute to change.
3. The receiver can detect the change.
4. The sender and receiver are able to synchronise.

Timing channels have a slightly different set of criteria:

1. Sending and receiving subjects have access to an attribute.
2. The receiver has access to a time reference.⁵
3. The sender can modulate the receiver's response time for detecting a change in the attribute.
4. The sender and receiver are able to synchronise.⁶

The methodology is applied by first identifying the shared resources and their attributes, and then the system primitives that can be used to manipulate them. This information is then organised into a matrix where the rows correspond to shared attributes and the columns correspond to available primitives. The elements of the matrix are labeled by a R, M or both to indicate whether the primitive observes the attribute, modifies it, or both. If the row for an attribute contains both R and M, it may be usable as a covert channel. A weakness in the methodology is that it does not state how to identify the attributes or primitives, or how to determine whether or not an exploitation is possible. Due to this informality, concluding the analysis is a subjective decision. At the lowest level, analysis is performed on every primitive: 1) that a subject may invoke, and 2) that causes or observes a system state change. These primitives include all system traps, kernel interface calls (which are interpretations of system traps), functions made available by trusted processes, and cpu instructions. Depending on its arguments, for example, the move instruction may affect system memory or cache contents; these effects may be visible to subjects at other security levels.

Once constructed, the matrix is transformed by calculating the transitive closure of the information flows. The transitive closure simply extends all direct information flows to include indirect flows as well. Both Tsai [20] and Levin [11] have argued that this step is not necessary because indirect flows must be based on direct flows. Levin notes that, because an exploitation may exist for indirect flows, such a conclusion is only justified if no direct flows are eliminated from consideration as having no exploitation. Some tools exist for assisting in constructing an SRM from specifications. Gemsos [11] used FDM (Formal Development Methodology) tools [5] to generate a SRM for storage channel analysis. The system interface was described using the Ina Jo specification language.

⁵ Actually, Kemmerer asserts that both sender and receiver need access to a time reference. It does not appear necessary, however, for the sender to have such access so long as the actions that the sender takes are known in advance to affect the receiver's response time.

⁶ This is not necessarily a significant requirement. In the absence of synchronisation, variations in the sender's and receiver's relative speeds show up as noise in the transmission (which can be eliminated using suitable encoding). Synchronisation is required for a noiseless channel, however.

Noninterference

A subject is “noninterfering” with another if the subject's actions do not affect the other subjects view of the system's behavior [6]. If we view a system as a sequence of inputs and outputs, noninterference can be stated: subject A does not interfere with subject B if, for every sequence w of inputs and outputs of A and B, the output seen by B is identical to that which would be seen by B in the sequence that is identical to w except that all of A's inputs have been deleted. A system has the MLS property if, for every subject A whose security level properly dominates that of another subject B, A does not interfere with B. These ideas can be more precisely stated; a state machine definition is given in [6].

Noninterference is characterised by system behavior at an interface; the interface may be at any level of abstraction. Noninterference belongs to the family of information flow models because the satisfaction of the policy can be shown by demonstrating that no information flows between noninterfering subjects (as defined by the label dominance relation). Because covert channels are means by which high subjects can interfere with low subjects, a system that has the MLS property has no covert storage channels.⁷ Covert channels may then be discovered by attempting to show the MLS property for a system, and examining the places where the proof fails. This approach was used, in comparison with the SRM methodology, to analyse specifications of the Secure Ada Target for covert storage channels [7].

The noninterference approach has the advantage that, unlike the Shared Resource Matrix approach, it is possible to “know when you are done.” The method of analysis, however, is extremely arduous. Constructing proofs that source code satisfies a particular specification is extremely difficult; producing arguments about where and why a proof fails (and that the proof could not in fact have succeeded) is even more difficult.

A Code Level Technique

Although the SRM methodology [10] provides an approach to identifying covert channels, it leaves out the specifics of how to find channels in source code. Tsai [19] provides a way to identify channels in C source code using semi-automatic analysis. It is claimed that the method is formal and that all storage channels are found. Although the method does use some formal techniques, the strength of the results is limited by the strength of the (to date, informal) correctness claims for system implementation in general. Additionally, the choice of C as the implementation language makes the analysis vulnerable to incorrectly implemented pointer manipulations that cannot be caught by the analysis. Tsai's method can be seen as an extension of the SRM methodology. It can be described in three broad phases:

Identify trusted interface primitives: This information is available from the system DTLS.

Determine the visibility/alterability of internal TCB variables This determination starts by first examining, using dataflow concepts, whether or not variable values are (potentially) returned to a caller of a TCB primitive, or are potentially altered by call. For example, the statement “ $x = y;$ ” causes information to flow from y to x . If the statement is guarded by an “if B”, then information flows from B to x as well. Dataflow rules for tracking information flow in code have been given by Denning [4]. This analysis is performed on a function by function basis. Potential function call paths are then examined by discovering which functions can be called from each TCB primitive. TCB variables that can be set or observed from the TCB interface are then flagged as covert channel attributes for a code level SRM. The TCB primitives from which the attributes can be modified or from which the attributes are visible are identified as the columns of the code level SRM.

Analyse shared attributes (and weed most out): The criteria for weeding out identified attributes are not formalised. Attributes may be weeded out either because the information flows supported are legal, or because they confer no useful information.

Tsai's method identifies numerous attributes. It does not, however, provide a formal way to determine which

⁷In [7] it is stated that a system having the MLS property might still have timing channels, because there is no explicit representation of time in the noninterference model.

are actually harmful. Ideally, an FSPM would be mapped down onto the code, and the legal channels could at least then be formally eliminated. The "no useful information" channels are more difficult yet.

A weakness of the method is its focus on global variables. First, such variables should be considered in *assembler* as well, and also i/o should be analysed. Additionally, it should be possible to include the CPU instructions as part of the TCB interface. I/o can present obvious channels, such as the print job identifier channel for Unix. In that channel, print job numbers are written into a file in a DAC-protected directory by the trusted printer daemon. New jobs are numbered after the last job written into the file. Because users cannot access the directory directly, they cannot read the file, but they can notice which job numbers are assigned to their print jobs. Using the SRM methodology, such channels can be detected by identifying the resource consisting of print job numbers.

Tsai's method, used in Trusted Xenix [2], identifies essentially three kinds of storage channels:

resource exhaustion A resource pool (e.g., a memory allocator) returns an error message when there is no more resource to allocate.

policy conflict An operation that may compromise information, but which must be maintained for compatibility or usability. For example, some systems refuse to remove directories when their (high) contents are not empty.

event count Channels in which the sender can manipulate a (usually integer valued) index or size attribute of a resource. For example, a report of the total number of free disk blocks is an event count channel.

Timing Channels

Timing channel identification has historically been ad hoc. In order to measure the time that an operation requires, the receiver of a timing channel must have a point of comparison. The most obvious such point of comparison is the system clock. Points of comparison need not be so obvious, however. As stated in [22], a timing channel may exist whenever there are two or more clocks where a clock is defined to be "any series of events, visible to a process, which may be used by the process to measure the passage of time."

In [22], Wray proposes a methodology that focuses on the identification of clocks. Using the methodology, all clocks are identified and an N by N matrix for the N clocks is constructed. The vertical axis would list the clocks to be modulated by the sender, and the horizontal axis would list the clocks to be used by the receiver to measure the modulations. Except for the diagonal of the matrix, each cell can be filled in with the modulation scenario. It is not possible to modulate some clocks. This technique is not extremely different from the SRM approach. Clocks are discovered by first listing "clock classes" (an informal activity), and then subdividing the clock classes by their internal events. For example, some clock classes proposed in [22] were: instruction timings, operating system calls, the system clock, and disk I/O transfer time.

Once clock classes have been identified, individual clocks (usually subparts of a class, for example the different interrupts for a disk transfer) are identified, and example exploitation scenarios are hypothesised. For a particular pair of clocks there may be a large number of possible exploitation scenarios. Choosing the fastest and most difficult-to-audit scenarios is an ad hoc process. In [22], Wray provides a number of example exploitation scenarios:

disk-arm The sender positions the disk head by performing i/o on known tracks. The sender issues two read requests (to different sectors) and examines the completion time of two read requests.

disk-arm write Similar to the above, the sender first positions the disk head. The receiver issues two write requests such that they partially overlap on the disk and such that one will happen first depending on the position of the disk head. The location of the disk head is revealed by which value remains. This is an example of a "direct" channel, in which the information is deposited on a medium without the receiver learning it first.

printer write with timing loop The receiver issues print requests and waits in a timing loop, after which it cancels the request. The sender modulates the length of the receiver's timing loop by contending for memory access.

bus contention A high processor modulates memory access contention with low processors. This channel is potentially large.

cache reload The receiver fills the processor cache with low information. The sender causes some cache entries to be invalidated, and the receiver then notices the time delay in accessing memory.

Other timing channels have been presented in the literature; an exhaustive list of reported channels is beyond the scope of this document.

Channel Capacity Estimation

Channel capacity estimation should be done after the identification phase is complete. The capacity estimates serve as input to the channel reduction process. Capacity estimation has three components:

- measuring the time each TCB primitive requires to execute,
- finding scenarios for the manipulation of each channel, abstracting the scenarios to gain a guarantee that no other scenario exists that can drive the channel at higher speeds, and
- estimating the rate at which information can be transmitted using the abstracted scenario.

In principle this approach works for both storage and timing channels, but techniques for finding the information rates of abstract scenarios may differ. This section discusses each of these components.

Measurement

Measurement requires that, for each evaluated hardware base, all TCB primitives, that have been related to covert channels in the identification phase, be timed. Both kernel and server interactions will require timings since CCA will be performed for both the kernel and servers. It can be difficult to obtain believable timings for TCB primitives. Primitives may execute much faster than the clock ticks of the system clock used to measure the time.⁶ In this case, it is necessary to time n calls of a primitive. For primitives that allocate (or deallocate) resources, however, it may not be possible to execute the primitive n consecutive times. Primitives may have to be paired (allocating and deallocating) to measure their composite timings. Many primitives may require different amounts of time to execute depending on the system state. Characterising the state is sometimes possible (e.g., file creation in a large directory is slow), but often the state is such a complex result of previous system history that analysis is not feasible. To blend the differences, the timings should be measured multiple times, and confidence intervals should be used to gain assurance that actual times are close to measured times with high probability.

Many primitives transmit information through failure conditions; it is therefore necessary to measure both calls that succeed and calls that fail. An additional, unquantifiable, concern is that the time that a primitive requires to execute often depends on what arguments are provided it. Arguments can sometimes be selected that "do no work" (resulting in fast executions), but covert channels cannot in general be driven by "null" operations. "Reasonable" arguments must be chosen.

Idealised Scenarios

Channel capacity depends heavily on the scenario, or algorithm, used to manipulate it. In the abstract, it is very difficult (virtually impossible) to show that a particular scenario for manipulating a covert channel

⁶If special diagnostic hardware is available, this may not be an issue.

is optimal. It is much easier to show that every scenario for the exploitation of a certain channel must pay a specific overhead (e.g., deallocating resources that must be allocated to exploit the channel). Some well known overheads, like synchronisation, however, cannot in general be included because it is not known how clever an attacker might be in synchronising the sender and receiver. In general the attacker is assumed to have the use of the entire system (no interference from others). The exploitation scenario can therefore be started by selecting the most efficient TCB primitives for manipulating (sender modifies, receiver observes) the channel, regardless of whether there is an apparent way to use them for that purpose. If the capacity is sufficiently low, the analysis can end there. If the capacity is high, a search can then be performed for reasons why those primitives can't be used, or why other primitives must also be used, lowering channel capacity.

Estimation of Information Rate

Although there is general agreement that information theory (Shannon's definition) is the proper basis for capacity calculations, methods of calculating covert channel capacity is an ongoing research area [21, 13]. Except for some simple classes of channels, precise calculation of covert channel capacity exceeds current mathematical techniques. In order to make calculations feasible, however, simplifying assumptions can be made. By avoiding capacity underestimation, simplifying assumptions sometimes dramatically exaggerate channel capacities.

In the TCSEC, acceptable capacities are expressed for individual channels. In previous evaluations, channel "aggregation" has been an issue. The motivation for aggregating several channels into a single one is the recognition that it may be possible to exploit several channels in parallel, thus increasing the rate at which information is compromised. In the Trusted Xenix evaluation, aggregation was a consideration for channels based on attributes which could be created in large numbers (e.g., directories) by an attacker. For single-processor systems, the effects are essentially to drop context switch time from the capacity calculation. For multiprocessors, aggregation may introduce a factor of n into the capacity estimation where there are n processors (because the channels can be exploited in parallel). Some agreement with the evaluation teams will be required to determine which channels will be subject to aggregation and which will not.

Resource exhaustion (and some policy conflict) channels may be modeled as a one bit noiseless channels. Analytic techniques (and even tools⁹) exist that are adequate to calculate capacities for one bit noiseless channels. An upper bound on the capacities of event count channels can be obtained through simplifying assumptions of the technique used for one bit noiseless channels. Timing channels are more difficult to estimate. Some timing channels operate at memory speeds, limited only by the time required to resolve hardware contention [14]. In this case, the channel is not sustainable using encoding because the sender must "take time out" to encode the information¹⁰, and the analysis can be simplified. Also, contention resolution that is fair in the sense that it does not penalise one symbol or another with a delay reduces the benefits to be gained through coding. Reduced channels will require more careful analysis, however. The following section presents a measurement technique that is useful for many storage channels.

One Bit Noiseless Channels

This section summarises the technique given by Millen [13] for finding the capacity of one bit noiseless channels. A channel may not be noiseless in a real system, but this results only in possible overestimation of channel capacity. Using information theory, the capacity of a noiseless discrete channel is known to be defined by the limit

$$\lim_{t \rightarrow \infty} \log_2(N(t))/t$$

where $N(t)$ denotes the number of messages that can be sent in time t . When the effort required to send a 1 is much different from the effort required to send a 0, the capacity significantly exceeds the information rate

⁹Which were used in the Trusted Xenix evaluation.

¹⁰A consequence of allowing coding in channels that operate at memory speeds is to have channel capacities that exceed memory speeds.

obtained when an equal distribution of ones and zeros is assumed. When the transmission of information is effected by a state machine with more than one state, the effort required to send a 1 and to send a 0 may depend on the current state of the state machine. Figure 1 shows a two state machine which corresponds to a one-bit noiseless channel where the edges are labeled by pairs: the symbol before each "/" designates the symbol being transmitted when that edge is traversed, and the letter after the "/" identifies the edge and is a parameter for how much time is required to traverse that edge. The parameters can be understood as follows:

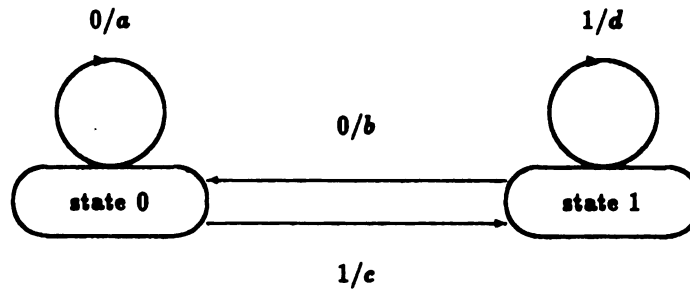


Figure 1: State Diagram For A One-Bit Channel

- a send 0 if the last bit sent was 0,
- b send 0 if the last bit sent was 1
- c send 1 if the last bit sent was 0
- d send 1 if the last bit sent was 1

These parameters are related to the definition of channel capacity in [13], where it is shown that the capacity is given by $\log_2(r)$ where $r > 1$ is the (unique) solution of the equation

$$\begin{vmatrix} 1 - r^{-a} & -r^{-b} \\ -r^{-c} & 1 - r^{-d} \end{vmatrix} = 0$$

This equation can be solved numerically given the four state transition times. A more general form, presented in [13], may be applied to state machines which have more than two states and two symbols and can therefore transmit more than one bit at one time. The solution to the resulting equations, however, becomes unworkable when the number of states is much larger than two. In order to measure channel capacity for event count channels, which are modeled as state machines with N states (N possibly large), we can use a simplification which is guaranteed to not underestimate the channel capacity. The simplification finds an upper bound for n -bit channels by always using the smallest state transition time. For N states and N^2 state transitions $s_1, s_2, s_3, \dots, s_{N^2}$, $\log_2(N)$ bits may be transmitted at one time. An upper bound on the channel capacity is therefore given by:

$$\frac{\log_2(N)}{\min(s_1, s_2, s_3, \dots, s_{N^2})}$$

This upper bound is not tight, but may allow the elimination of some event count channels from further analysis.

Channel Capacity Reduction

If a channel's capacity exceeds acceptable limits, channel capacity must be reduced or audited. Accurate estimation of channel capacity is important because it determines the selection of and severity (performance impact) of reduction techniques. For example, delays that are unnecessarily large degrade system performance unnecessarily whereas delays that are inadequately small affect system security adversely. Some channels may be eliminated by design changes (that usually reduce functionality), or by using certain configuration options. For example, Gemsos allows most storage resources to be statically preallocated by security level, therefore eliminating most resource exhaustion channels. Such preallocation is expensive, however, and primarily addresses a class of storage channels that can be effectively reduced using delays.

Storage Channel Reduction

The two major techniques for reducing storage channels are delay and randomisation. Resource exhaustion channels can be reduced by temporarily suspending (delaying) any process that exhausts a resource. Such delays usually have acceptable performance impact because resource exhaustion is a (relative) rare event for most resources. Delay can be used in a similar way for policy conflict channels. Delay is both less effective and more costly for event count channels that report global status (e.g., total free blocks), however, because: 1) the attribute being observed may take on many values and the receiver therefore may receive more than one bit per delay, and 2) the delay must be imposed on every use of the reporting function.

Event count channels that show how resources are allocated (e.g., new Unix pid's) respond well to randomisation, assuming a sufficiently strong random number generator. For Trusted Xenix, a congruential random number generator seeded by the time of day and number of system calls provided sufficient strength. In practice, an exploiter could not discover the seed because of the frequency and variable number of system calls.

Randomisation is less effective against status reporting event count channels because the accuracy of the functions is inversely related to the degree of "fussing" provided by randomisation.

Timing Channel Reduction

For some timing channels, a system has no way to tell the difference between exploitation and normal activity. This characteristic makes timing channels intrinsically more difficult to reduce than many storage channels. This is particularly true when the channel is based on high speed hardware based contention. The (now classic) example is the shared bus multiprocessor where there are three or more processors [14]. In that channel, low processor A increments a global memory location as rapidly as possible, high processor B sometimes accesses global memory, contending for the bus, and low processor C continually checks the progress of processor A. Bus cycles stolen by B show up (to C) as failures to increment the memory location. This channel operates at memory speeds, and cannot be meaningfully audited by software because the operations used to transmit information are "normal" processing, and because their volume would quickly overwhelm any audit system.

It is beyond the scope of this plan to describe how to delay all timing channels. Several possibilities are:

- Where the system primitives that return the value of a clock can be identified, use delay to reduce the capacity. It is worth noting that the alphabet of such channels may be large, and that the information rate may not be reduced as effectively as it is for resource exhaustion channels (which have an alphabet of {error, not error}).
- Randomly introduce perturbations into readily available clocks to reduce the speed or ease of signaling. Noise may reduce, but cannot close such channels. Analytic techniques for evaluating the effect of the noise may be difficult.
- Fuss some clocks to reduce the accuracy with which covert senders and receivers can measure clock differences. A variant of this approach, used in the VAX security kernel [8], randomised system timers

and added random delays to the initiations and notifications (of completion) of IO. This technique, called "fussy time," attempts to isolate each process from the precise timing information provided by hardware supplied clocks such as interrupts and cpu bus contention. Although the measurement technique was not specified, [8] reports evaluation team agreement that all timing channels in the VAX security kernel were reduced to less than 10 bits per second.

- For contention channels like the bus channel, schedule the resource (in that case, the bus) by security level, so that most contention is limited to being within a security level (and therefore legal). The performance impact of this approach is not known, but may be severe (all processors contending for the bus would have to change security level at the same time).

Assurance of Channel Reduction

Although channel identification may be conducted using specifications, channel reduction techniques must be implemented, and assurance of their effectiveness must be gained at the code level. At the least, some form of covert channel testing must be performed to evaluate the effectiveness of reduction techniques. Code analysis tools may assist for storage channels. Trusted Xenix used "covert channel flow tracing," a method in which function call trees and variable references are analysed to ensure that a delay or randomisation algorithm is always used before selected variables can be reported to a receiving process. IBM did not have a production quality tool and, in practice, performed much of the analysis manually. If the analysis is correctly performed, assurance can be gained in general that storage channels are reduced. It is not clear that such tools can be effective for timing channels, however. Assurance for timing channels may depend on comprehensive testing and code inspection.

Planning the Analysis

The covert channel analysis for a large system should satisfy three goals: 1) proceed concurrently with system development, 2) provide credible results, and, and 3) remain within available resources.

There are basically two approaches to concurrently performing CCA and system development: 1) substantially automate the analysis so that it can be completely redone after each significant system change, or 2) decompose the system into parts each of which can be independently analysed, and then combine the analyses as the system is constructed. In either approach, analysis should be performed continually during development so that feedback from the analysis can impact the system design and implementation.

Although attractive in the abstract, substantial automation of CCA is an area of active research. A number of tools exist that may assist in CCA by automating part of the process or by enforcing rigor in specification: Malpas[12], Ina Flo [5], and an IBM proprietary tool [19] (this list is not exhaustive). In addition, a covert channel analysis tool is under development inside TIS [16]. As is the case with programming projects, the use of such tools may require dramatically more time than is anticipated.

Unfortunately, decomposing the system into components upon which independent CCA can be performed is also a research area. In principle, modular covert channel analysis could be based on Kemmerer's SRM, but there are no worked examples (known to the author). Changes to each component would at the least force reanalysis of the affected component. If the reanalysis changes the results obtained by the previous analysis, other components that depend on the changed component must be reanalysed as well.

Because CCA is still an art, the credibility of the results is somewhat subjective. Clearly an analysis that fails to find many channels that are subsequently discovered in penetration testing or evaluation will not be credible, however. Both specification and code level analyses may miss channels. In general, the rigor imposed by using tools or formal techniques may increase the confidence that specification based analysis is sufficient. It has been claimed that code level analysis finds all storage channels [19]¹¹. The handling of timing channels will of necessity be informal; here, confidence can be gained only through sustained effort to find as many channels as possible.

¹¹ However, see section A Code Level Technique

Channel Identification

The most fundamental decision for covert channel identification is whether to use noninterference or some form of the SRM methodology (or both). In [7], noninterference was compared with the SRM methodology. Although the authors refrained from selecting one strategy as the best, they noted that noninterference proof failures might become unworkably difficult as the size of a specification increases. Although, in a high level (and simple) specification, the ideal of noninterference might be reasonable because channels present at that level would of necessity be present in any faithful implementation, a low level specification would (practically speaking) always cause proof failures. The authors further noted in [7] that noninterference, by itself, probably could not be a comprehensive tool, although the SRM might be. Noting that their study was limited, the authors in [7] refrained from selecting one strategy as the “best” and indicated their intention to use both in the future. Unfortunately, a developer must choose a strategy even though there may not be adequate information to show that it is always superior.

Selection of SRM methodology versus noninterference is difficult; in many large scale development activities, however, the following disadvantages of noninterference seem to argue against its use:

- Proofs are difficult; interpreting proof failures is even more difficult.
- Proof failures that are not understood provide no information.
- Noninterference may require a level of formality that cannot be sustained on a large project with many changes to the system.

The following assumes the use of some form of the SRM methodology.

Storage Channels

The primary decision to be made is whether to pursue a code level analysis, an analysis based on specifications, or both. CCA has been more frequently performed on specifications than on implementation code. The considerations can be broken down:

- specification analysis
 - pro
 - * easier to do informally
 - * potentially less expensive
 - * some tools exist (e.g., Ina Jo, Ina Flo[5] ¹²)
 - * the analysis is less sensitive to minor system changes
 - con
 - * depends on specification accuracy
 - * omits necessary detail—channels not present in the specification will not be discovered
 - * there is no way to know when the job is finished (i.e., what specification is low level enough?)
- code level analysis
 - pro
 - * includes implementation detail
 - con
 - * more expensive
 - * few tools, e.g., Malpas [12], are available

¹² Experience on two projects indicates that Ina Flo is not yet mature enough to use.

- * tools are required
- * because of the complexity of the real implementation, coverage is not likely to be complete—the detail can overwhelm the analysis

The true difference between analysis of specifications and code depends on the amount of detail present in the specifications. Some analyses have used very detailed specifications [11] containing more than 700 state variables. Although there are more “pro” items for the specification approach, the omission of necessary detail and the dependency on specification accuracy are severe handicaps. Equally severe is the great complexity of a code level analysis, in which detail can overwhelm the intuition of the person performing the analysis. Given the limitations and costs of each approach, it is difficult to choose one exclusively. A dual track approach therefore seems most prudent.

A specification analysis should be conducted on the interface specifications, and on each refinement of those specifications. Parallel with that, a code level analysis should focus on validating (not verifying) the correspondence between the specification and the implementation. Although a breakthrough in formal code analysis is possible ([16] may eventually be such a silver bullet), the code level analysis should focus on “informally” validating the specification analysis. Much of the code analysis will probably be manual, but tools to assist the analyst should be obtained or written as necessary. If possible, tools such as Ina Jo and the SRM matrix generator should be used to enforce specification consistency through the provision of type checking, etc., and to construct the SRM.

At the interface level, the first step in the construction of the SRM is to identify the TCB primitives that may be used to manipulate system attributes. Normally this is the TCB interface. It is necessary in the SRM approach for the R and M entries in the cells of the matrix to represent all direct flows between primitives.¹³ In this context, two primitives A and B are *atomic* if every interaction between them affects the system state as if they executed sequentially in some order. If two primitives of the SRM were not atomic, then a worst case analysis (including all possible interleavings) would have to be applied to determine what data flows between the two primitives were possible. The kernel calls of some operating systems (e.g., most versions of Unix) provide a simple version of atomicity by suspending most process scheduling during kernel processing. Even with these kernels, some operations will not be atomic because multiple processes may have to be suspended in the kernel waiting for I/O. The analysis should identify what operations are atomic, and how information flows between any non-atomic operations are included in the SRM.

Timing Channels

Identification of timing channels must depend on an informal but extensive search by knowledgeable developers. Wray's methodology can assist in guiding the search for clocks, and the matrix proposed in the methodology can assist the developers in keeping track of the relationships between different identified channels. An approach similar to that used for penetration testing (the flaw hypothesis methodology) may provide the best results. Because timing channels often depend on hardware contention, it will be necessary to conduct the testing on all significantly different hardware platforms (particularly multiprocessors).

Capacity Estimation

CCA for a family of hardware bases should be parameterised by hardware timing characteristics for each supported hardware base. The determination of which channels can be aggregated affects capacity estimation. This determination should be made as channels are identified. Channel capacities for multiprocessor hardwares will require special consideration since the multiprocessor version will probably have more identified timing channels. The timing information can be derived from engineering data or from test programs written to derive the characteristics of each hardware base. The multiprocessor hardware bases will require additional tests to measure characteristics not present in uniprocessors.

As given above, analytic techniques exist for some channel types. For others, upper bounds are required. The use of coding theory is indicated wherever the cost of sending one symbol is much larger than the cost

¹³If a transitive closure is performed, indirect flows would be present as well.

of sending another (perhaps because of a delay). For channels in which all symbols are equally easy to send, the use of coding theory provides little capacity increase, and capacity can be approximated by assuming an equal distribution of output symbols. For such channels, the capacity is $\log_2(n) \times \text{cycles per second}$ where n is the number of possible output symbols in a cycle.

Storage Channel Reduction

The kind of channel (resource exhaustion, policy conflict, or event count) affects the available alphabet. Exhaustion and policy conflict tend to be binary valued. Event count channels usually have numerous symbols. Some channels can be eliminated through design changes, for example, by removing the status reporting functions, or by changing them to tell white lies. Other channels can be reduced primarily through delay and randomisation. Global identifiers, for example, the process id in Unix, present special problems. They can be reduced using randomisation so long as the space of identifiers is much larger than the number of identifiers that can be in use at any one time, and so long as allocation of the next identifier always chooses randomly from the entire pool of unused identifiers. When caching is used to optimise the use of resources associated with an identifier, the cache reduces the options for selection of the next identifier, and can be exploited to signal. For such global identifiers, the maintenance of separate security level partitions (that move slowly in response to demand) for the identifiers and the cache can be used to reduce capacity.

Two attacks on per-process delays must be prevented for delays to be effective: 1) interruption of the delay, and 2) overlapping of multiple delays. If a delay can be interrupted in any way, it is not effective because a process can notice when another process is in a delay, interrupt it, and resume covert communication. It should not be possible to destroy processes that are suspended in delays.

If multiple per-process delays can be overlapped, an attacker may use multiple processes to effectively poll a resource more rapidly than permitted during the delay. This scenario can be prevented by serialising delays. A general serialisation scheme is as follows. Let the delay period be D seconds. The first process to be delayed for use of the channel is delayed D seconds. The second process to use the channel is delayed for the greater of: D seconds or D seconds from the time the first process finishes its delay. Multiple delays for a resource may therefore not overlap. Using this technique, delays can be overlapped when they are imposed on different resources.

Timing Channel Reduction

The suggestions in the above section on timing channel reduction apply as stated. In addition to the use of delays to reduce capacity, however, delays might be used to *hide* activity. For example, to prevent a channel in which one process infers information from another through the time to access a shared page (i.e., whether a page fetch was necessary or not), sporadic delays that would correspond to page fetching could be introduced. The delay must conceal from the receiver the fact that a page fault was not necessary because the sender had already paged in the data. Specifically, all low processing that could not occur during a real page fault must be prevented during a delay that mimics a page fault. If other low tasks could run, the receiver could schedule another task to run and then measure its progress. Processing by higher level tasks could continue, however. Additionally, the delay must be realistic. For example, actually performing a page fault can be expected to take varying amounts of time to account for disk latency, rotational delay, etc. If a delay always takes exactly the same amount of time, but the real operation times would vary, the channel is not effectively reduced.

Recommendations

Covert channel analysis can be approached in the following sequential phases. In each phase, all activities may proceed in parallel:

1. (a) Obtain timing parameters for all hardware bases. Programs that obtain the parameters may be developed on prototype or untrusted versions of the final system.
- (b) Begin the search for timing channels.

- (c) Survey available tools for system specification and SRM construction, and evaluate. Select one, or reject all and develop and use a proprietary notation.
- 2. (a) Continue search for timing channels.
- (b) Complete design documentation to incorporate the use of the selected tool or notation in the specification layers.
- (c) Decide which system components can be independently analysed.
- (d) Begin development of source level tools to support the specification analysis, and also to provide evidence of coverage for channel reduction.
- 3. (a) Continue search for timing channels.
- (b) Enhance source level tools as necessary.
- (c) Construct the SRM for each system component with a stable interface.
- 4. (a) Continue search for timing channels.
- (b) Combine analyses of separate components and categorise channels discovered by the SRM.
- (c) Use the source tool to validate the specification analysis.
- (d) Calculate channel capacities (for all platforms, as possible), eliminating from further consideration channels that are too slow.
- 5. (a) Continue search for timing channels.
- (b) Reduce or audit identified channels through system source or configuration changes.
- (c) Use the source tools to check coverage of reduction techniques.
- 6. (a) Continue search for timing channels.
- (b) For all changed components, until the system is frozen:
 - i. Recalculate the SRM (or determine informally that it need not be recalculated).
 - ii. Revalidate the SRM using source tools (incrementally, if possible).
 - iii. If any new channels are discovered, calculate their capacity, and reduce or audit as necessary and possible.

The search for timing channels is present in each phase, but the effort required in each phase may not be equal. The search for timing channels should be performed until the number (and severity) of additional channels discovered using a given amount of energy falls below some threshold. Because system changes can introduce new channels, the search must be revisited until the system is frozen (but perhaps at much reduced levels of effort).

The CCA will require a diverse set of skills: 1) skills in the use and evaluation of tools (including an understanding of formalism), 2) coding skills, 3) knowledge of the role that covert channels played in past evaluations, and 4) design knowledge of the system being analysed. The writing of test programs and the search for timing channels can contribute to design knowledge. The covert channel "team" should include trust engineers and developers.

It is important to allocate sufficient energy for these tasks. The energy devoted to CCA will be used to evaluate tools, create (modest) tools, write test programs, perform analysis on a complicated body of changing software, produce designs to reduce and audit identified channels, and achieve assurance that identified channels are reduced. *This is an enormous amount of work and should not be underestimated.*

Acknowledgments

The author would like to thank Mike Masurek, Tim Redmond, and the reviewers for helpful comments on the technical content and presentation.

*

References

- [1] National Computer Security Center, *Department of Defense Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD, December 1985.
- [2] L. Badger, F. L. Mayer, T. Redmond, "Trusted Xenix Covert Channel Capacity Estimation And Reduction," TIS Technical Report #364.
- [3] D.E. Bell and L. Lapadula, *Secure Computer System: Unified Exposition and Multics Interpretation*. (Technical Report No. ESD-TR-75-306, Electronics Systems Division, AFSC, Hanscom AF Base, Bedford MA, 1976).
- [4] D. E. Denning and P. J. Denning, "Certification of programs for secure information flow," *Communications of the ACM*, (July, 1977), pp. 504-513.
- [5] S. T. Eckmann, "Ina Flo: The FDM Flow Tool," *Proceedings of the 10th National Computer Security Conference*, p. 175-182.
- [6] J.A. Goguen and J. Meseguer, "Unwinding and Inference Control." *Proceedings of the 1984 IEEE Symposium on Security and Privacy*, 1984.
- [7] J. T. Halgh, R. A. Kemmerer, J. McHugh, and W. D. Young, "An Experience Using Two Covert Channel Analysis Techniques on a Real System Design," *IEEE TSE* Vol. SE-13, No. 2, Feb. 1987.
- [8] W. Hu, "Reducing Timing Channels with Fussy Time," *Proceedings of the 1991 Symposium on Research in Security and Privacy*, May, Oakland, Cal.
- [9] *Information Technology Security Evaluation Criteria, Harmonised Criteria of France, Germany, the Netherlands, and the United Kingdom* May 1990.
- [10] R.A. Kemmerer, "Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channels," *ACM Tran. on Computer Systems*, 1:3, August 1983, p 256-277.
- [11] T. E. Levin, A. Tao, S. J. Padilla, "Covert Storage Channel Analysis: A Worked Example," *Proceedings 13th National Computer Security Conference*, Oct. 1990.
- [12] Malpas: a commercial tool used on the VAX security kernel that can assist in code level analysis for multiple languages: Pascal, PL/1, Fortran, and C.
- [13] J.K. Millen, "Finite-State Noiseless Covert Channels," *Proceedings of the Computer Security Foundations Workshop II, Franconia, New Hampshire, P.81, 1989*.
- [14] "Minutes of the First Workshop on Covert Channel Analysis," September 19-21, 1989.
- [15] T. Redmond, B. A. Mayer, F. L. Mayer, "The Abstract TMach Kernel Model," TIS Technical Report #293.
- [16] T. Redmond, "Formal Approach to Identifying Covert Channels in Source Code", TIS Technical Report (in progress).

- [17] C. E. Shannon and W. Weaver, "The Mathematical Theory of Communication," the University of Illinois Press, Urbana, Illinois, 1964.
- [18] D. Sutherland, "A Model of Information," Proceedings of the 9th National Computer Security Conference, Sept. 1986, p. 175.
- [19] C. Tsai, "A Formal Method for the Identification of Covert Storage Channels in Source Code," Proceedings of the 1987 symposium on Research in Security and Privacy, May, Oakland, Cal.
- [20] C.R. Tsai, "Covert-Channel Analysis in Secure Computer Systems," Phd. Dissertation, University of Maryland, College Park, Maryland, Aug. 1987.
- [21] J. T. Wittbold, "Information Flow in Nondeterministic Systems," Proceedings of the 1990 Symposium on Research in Security and Privacy, May, Oakland, Cal.
- [22] J. C. Wray, "A Methodology for the Detection of Timing Channels," Proceedings of the Computer Security Foundations Workshop II, Franconia, New Hampshire.
- [23] J. C. Wray "An Analysis of Covert Timing Channels, " Proceedings of the 1991 symposium on Research in Security and Privacy, May, Oakland, Cal.

**DEALING WITH MALICIOUS LOGIC THREAT
A PROPOSED AIR FORCE APPROACH**

Howard L. Johnson
Information Intelligence Sciences, Inc.
1903 So. Franklin St.
Denver, CO 80210
(303) 777-4266

Chuck Arvin and Earl Jenkinson
CTA Incorporated
7150 Campus Drive, Suite 100
Colorado Springs, CO 80918
(719) 590-8890

Captain Bob Pierce
AF Cryptologic Support Center
Hq. Electronic Security Command, AFCSC/SR
Kelly AFB, TX 78243-5000
(512) 925-2511

ABSTRACT

Trojan horses, viruses, worms, and other malicious logic that seek to interrupt service or modify or destroy data are not necessarily defeated by confidentiality mechanisms. The Air Force Trusted Critical System Evaluation Criteria (AFTCCSEC) [1] supplements confidentiality requirements found in the Trusted Computer System Evaluation Criteria (TCSEC) [2] by addressing integrity and service assurance. This paper introduces and describes criticality division/class G2 found in the AFTCCSEC. The approach imposes mandatory controls including access constraints, type enforcement, detection techniques, and use of a resource scheduling architecture. It applies to all life-cycle phases: development, distribution, operations, and maintenance. Features include program/data isolation (e.g., physical, logical or use of cryptography), protection against covert criticality channels (that allow malicious code insertion), and strict configuration control of software and hardware. Any TCSEC division/class and G2 criticality can coexist, though retrofit of G2 will require an existing TCSEC TCB of B1 or higher. This paper provides a basic understanding of the concepts and policy, and also addresses questions most often asked by reviewers of the AFTCCSEC document.

THE PROBLEM

Compromise of classified information has been the primary concern of DoD computer security for three decades. The viruses and Internet worms have shown the reality of malicious code attacks.

Work was accomplished under CTA Contract Number F41621-88-D5001 issued by Hq. Electronic Security Command, AFCSC/SR, Kelly AFB, TX 78243-5000.

There have been no previous DoD requirements, evaluation criteria, and models that specifically address the malicious logic problem.

Thus, systems that previously were deemed secure according to TCSEC requirements may in fact be vulnerable. Because upgrade message flow is usually allowed, this could theoretically provide the ability to insert and execute malicious programs. A likely attack involves inserting a small virus to monitor a compromised channel. It would recognize other malicious code hidden between communications protocol "end" codes in incoming messages and cause its execution. It might then erase any trace of the larger code. The enemy would have subversion capability over the life of the system (or until thwarted) to perform subversive missions (fact finding, sabotage, or attempting to leak classified data). The attackers would make these activities difficult to detect or to distinguish from other failures types.

There is a sense of urgency to provide defenses against potential debilitating malicious logic attacks on major command and control systems. We believe the best immediate defense is to provide quick, substantial reaction to the threat.

THE NATURE OF ERRORS AND FAILURES

When a computer system error failure is discovered, it is often not immediately known whether the cause is hardware, a design/development error, an accident, or a malicious attack. An error or an accident may result in a normal or simple failure, a failure that propagates, or one that exhibits nonpredictable (chaotic) behavior [3]. The most common state is "normal" (see for example Beizer 1983 [4] for references) although people like to talk about the exceptions (the 1989 AT&T failure Neumann [3]). An accident has no goal so one would expect the impact to a system to be naturally (e.g., normally) distributed. A malicious intruder (not a harmless hacker) will often seek debilitating system impact.

Malicious logic is generally more complex than an accident or an error. Accidents and errors are seldom caused by more than a single action and or flaw. An accidental action or flaw can normally be emulated by a few computer instructions. The length of known virus and worm attacks, however, is generally on the order of 1000 or more bytes. Sixty percent of reported viruses are derived from the Jerusalem B virus which is approximately 1800 bytes. Others range from 405 bytes (405 Virus) to 60,000 bytes (the Internet Worm). The reason code for a malicious attack is large is because the perpetrator usually has multiple objectives that include detection avoidance, formatting to conform to applications, causing a state of quiescence, planning, file searching, communications monitoring, trigger monitoring, self erasure, and/or propagation.

Design/development errors exist prior to and after validation (if not discovered) and generally repeat (e.g., after rollback). Hardware failures occur after validation and may be transient. If

they repeat, they can be caught by diagnostics. Accidents seldom repeat and are usually recognized by individuals involved after they occur. Malicious attacks can occur either prior to or after validation, though avoiding a thorough validation is difficult. Malicious attacks can repeat, may not repeat, are probably not revealed by diagnostics (but could be if the attacker desired), often have multiple stages, and sometimes give multiple independent results. Joseph and Avizienis [5] propose a logic tree approach that assists in determining the cause of an error or a failure.

SCOPE

In recent papers (e.g., [6]) we defined a need and an approach to deal with loss of integrity and denial of resource use. This evolved into the Air Force Trusted Critical Computer System Evaluation Criteria (AFTCCSEC) patterned after the Orange Book (TCSEC). The AFTCCSEC has been published as Air Force Special Security Instruction (AFSSI) 5029. Figure 1 shows the

division/classes of the AFTCCSEC and the relationship to the TCSEC D and C1 levels. This paper focuses on criticality class G2 that incorporates protection against malicious logic. Class G3 which addresses critical Air Force systems and classes F3, F2, F1 and E1 that address multilevel systems and higher assurance, are discussed in other papers [7 and 8]. The AFTCCSEC basically uses the TCSEC control objectives. A reinterpretation is required since AFTCCSEC addresses integrity and service assurance which complements the TCSEC application to confidentiality.

Criticality Division/Class	Protection
H	Same as sensitivity D
G	Single Level
G1	Almost the same as sensitivity C1
G2	Protects against malicious logic
G3	Supports Critical operations
F	Multilevel (Labels)
F1	Critical and Highly Critical
F2	Critical and Non Critical
F3	No clearance and Critical
E(E1)	Formal methods (no clearance and Highly Critical)

Figure 1 AFTCCSEC Division/Class

APPROACH

Current DoD budgets cannot afford to duplicate present Orange Book security costs. Therefore, in the AFTCCSEC we have taken an approach that has three implementation cost reducers. Each also reduces time until implementation and implementation risk.

a) Division/class required depends on mission criticality. Malicious logic protection is introduced at the G2 level where systems are neither critical nor highly critical and can be realized with a minimum fund expenditure.

b) Since the approach follows directly from the TCSEC, most

mechanisms and procedures required by the TCSEC can be used directly or modified to accommodate AFTCCSEC requirements.

c) Cryptography and cryptographic checksums used as isolation mechanisms will reduce vulnerability and cost of protection.

POLICY

This section reviews new policy proposed for the Air Force:

There shall be protection against malicious logic throughout the system life-cycle beginning with development and continuing through assurance, distribution, operations, and maintenance.

COMPUSEC techniques used by the TCSEC for discretionary access control, object reuse, accountability, assurance, and documentation shall be used where possible for program and data integrity and assurance of service protection.

COMPUSEC techniques shall be employed using Air Force accepted trusted approaches to control access by individuals and processes to programs (stored processes), data, and system resources.

Intrusion detection shall be used to discover unauthorized users, system misuse, or malicious logic. Response should include fault isolation, analysis, and malicious logic elimination. These capabilities shall be protected from malicious logic attacks.

Public and private key encryption, and cryptographic checksums shall be used for the protection of data and programs where technically feasible and when cost, performance, and risk requirements can be met. (Standards shall be developed that relate the strength of algorithms and key management approaches to the protection required, supplementing current use of encryption to protect classified and sensitive information.)

Information gained from traffic analysis shall not reveal knowledge of system or security protection details that could be used in a malicious attack.

Software shall be developed, stored, and delivered under strict configuration control and screening to make the probability of malicious hardware or software reasonably small.

PREVENTION APPROACH

As stated in policy, the AFTCCSEC uses techniques identical to the TCSEC including the trusted computing base (TCB), discretionary access control, object reuse, accountability, assurance, and documentation. Additional or changed techniques introduced at the G2 level are discussed further.

Constrained Access

Current security mechanisms control access of subjects to objects. Constrained access (Figure 2) adds one dimension to the access control process by constraining process access to objects, independent of user. Specific access type is also controlled (called type enforcement). A process must be on a valid process list to be executed and can be removed from that list to quickly contain malicious code. Constraints are identified by way of process and object profiles. Processes are restricted to interact as they were intended when programmed. Additional constraints restrict operations on objects to the minimum required subset. Constraints are identified by the developer and established by the security officer. Attempts to violate the restrictions are reported to the auditing function. There is strict configuration control of programs, constant data, valid process lists, process profiles, and object profiles throughout the system life-cycle to detect unauthorized modification or other potential malicious characteristics. The idea of a security policy between users, processes, and objects, (also called triplets) is discussed by Clark and Wilson [9] and the control by access type (also called type enforcement) is discussed by Boebert and Kain [10].

Covert Channels

A covert channel is a communication channel that allows unauthorized transfer of data in a manner that violates security policy. A sensitivity attack using a covert channel has three steps and a criticality attack has two steps as shown in Figure 3. In the TCSEC, the protection objective is to control data leakage. The TCSEC does

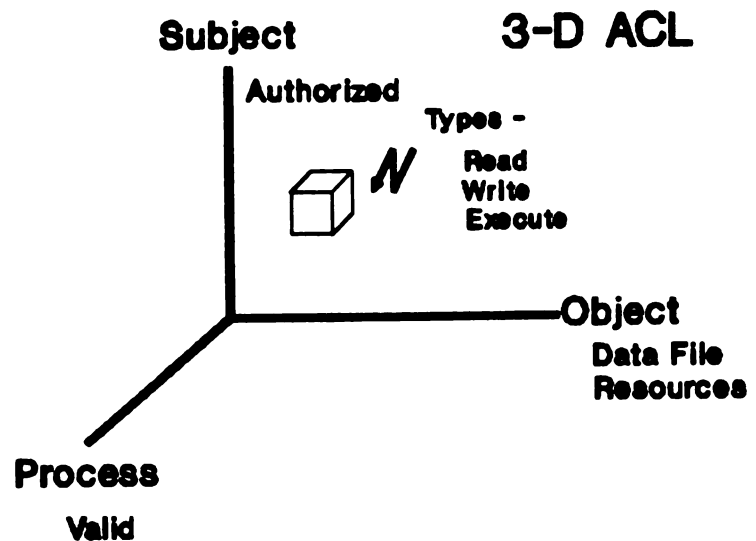
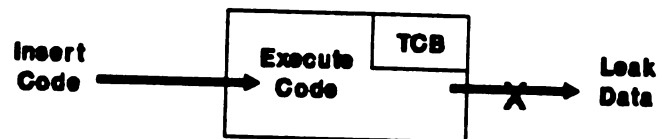
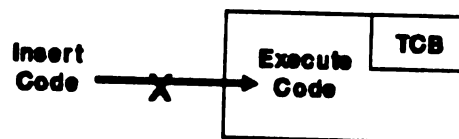


Figure 2 Access Control Triplets

Sensitivity



Criticality



X Covert Channels

Figure 3 Covert Channels

little, except through discretionary security, to control insertion and running of malicious code.

In criticality, covert channels are considered at the G2 division/class. Input data must be assured to be malicious logic free. Unauthorized channels "in" are potential covert channels that must be plugged or monitored and are of concern during development, delivery, operations and maintenance. AFTCCSEC covert channel methods are much the same as the TCSEC.

Cryptography

Cryptographic processes protect data from vulnerabilities in trusted domains or when data is traveling through untrusted domains. Private keys and private encryption algorithms are controlled by the TCB. Private key encryption prevents unauthorized reads and executes and some algorithms detect data modification. Decryption can invalidate unencrypted malicious logic (see the Pozzo-Gray Virus Containment Model [11]). Cryptographic checksums detect unauthorized modification. Public key encryption identifies the originator and, when used with a checksum, allows users even in an untrusted domain to detect modification. One-way encryption can be used for identification/authentication. Useful cryptographic processes are itemized in Figure 4.

Non Disclosure Identification Key Management Labeling Mechanism Protection Modification Detection	Bandwidth Filling (Covert Channel) Execution Prevention No Intelligent Change Enemy Spoofing Signature
--	---

Figure 4 Encryption Uses

AFTCCSEC requirements can be implemented with or without cryptographic processes. The intent is to open the door to cryptography use for other than confidentiality. Cryptography is efficient and inexpensive, and will become even more so as popularity is gained. The issue of required strength can be raised during design and dealt with by the appropriate DoD organizations.

Criticality TCB

The TCB for integrity and denial of service protection is larger and more complex than required by the TCSEC. Some of the functions (e.g., encryption) will normally be implemented in hardware. The primary increases in complexity are for detection and resource scheduling. Protocols, constant data, programs, and other control data are protected by the TCB by ensuring against unauthorized modification using cryptographic checksums. Cryptographic processes are essentially an extension of the TCB.

Trusted Distribution

The TCSEC is concerned about someone tampering with the TCB. The AFTCCSEC additionally worries about injection of malicious logic to system hardware, firmware, or software. Downloading of software within a complex system is also considered a distribution problem.

DETECTION APPROACH

Different than confidentiality, in preserving integrity and assuring service, an effective approach is to detect a problem and respond in adequate time (called critical time) to ensure the mission is still accomplished. At the G2 level, missions are not critical, however, detection is still based on a response time model. Assuming the malicious logic has avoided or defeated prevention mechanisms, the strategy is to identify the occurrence of a malicious attack, minimize its impact, and make the required correction (e.g., remove malicious logic).

Real-Time Audit

Malicious attack detection uses both an inductive and a deductive approach. The inductive approach determines intrusion behavioral characteristics and seeks them out. The deductive approach determines the normal behavior of many aspects of the system through statistics and use of profiles to help determine what is abnormal behavior. In each case a discrimination technique is used to reduce false alarms. This approach makes use of current intrusion detection research (presented by Lunt [12]) in application of statistical, rule based, expert, and other heuristic approaches. Nothing previously unproven is required by the AFTCCSEC, and the door has been left open to technological advances.

To avoid overhead, auditing can be accomplished in parallel by low-cost, high-performance hardware. Auditing may be thought of as a time prioritized data driven process. An audit function is triggered by the availability of its applicable detection/audit data. The maximum time until execution is determined based on the time variables specified by the policy. The function and time are placed in a time prioritized queue. The time is counted down and the function with minimum time is executed. The detection process checks itself for a possible denial of service attack and responds with a corresponding predefined response plan. Data compressing and discarding can be used.

Resource Scheduling

A precise resource scheduling policy must be defined, both to define what constitutes denial of service and to know what action must be taken in response to a denial of service attack.

Malicious Code Search

A tool that searches for malicious logic can be used during development as part of validation and during operations as part of configuration control, real-time audit, and communications monitoring. Search profiles help to recognize known or modified malicious logic, illegal system functions, or system-only functions. Non random data in encrypted (random) data streams can also be identified. Keeping the search profiles secret and the search process protected increases the mechanism effectiveness.

Hardware pattern matching logic can perform a fuzzy search. The

term "fuzzy" means that the profiles need not match exactly. Application specific frequency weighting can be used to further discriminate. Hardware implementation can reduce search of very large databases to a few hours and keep up with very high communication bandwidths.

SUMMARY

Current approaches in PCs to virus prevention, detection, and isolation/removal are an ever growing compilation of checks that a clever infiltrator eventually can work his way around. The philosophy of playing catch-up will always leave the penetrator with the advantage. That approach presumes repetition or variations on past attacks. The professional infiltrator will probably not use known malicious code.

The approach in the AFTCCSEC contains as a minimum all of the known protections used by antivirus software. The approach further depends on the existence of a TCB and utilizes strong encryption. The approach allows the protection to be site, application, and security officer specific, avoiding the predictability of canned solutions.

This paper has presented the policy and discussed new approaches introduced at the G2 division/class of the AFTCCSEC to deal with the malicious logic problem for DoD systems. The approach has used the concepts, mechanisms and language of the Orange Book (TCSEC) to simplify understanding and reduce implementation cost. The approach can be implemented in an Orange Book protected system or one where confidentiality is not an issue.

GLOSSARY

Constrained Access Control- A security policy that identifies which processes may be executed and what objects (i.e., other processes, storage objects, and I/O devices) they may access. Process and object profile data are used to ensure that each process access of an object is allowed and is of the allowed type.

Denial of Service - Action or actions that result in the inability of the system or any essential part to perform its designated mission either by loss or degradation of operational capability.

Integrity - Ensuring that data changes in only highly structured and controlled ways. Air Force regulations define integrity as a computer security characteristic that ensures computer resources operate correctly and that data in the data bases are correct. The integrity protection goal is to protect against deliberate or inadvertent unauthorized modification or execution.

Malicious Attack - Insertion of malicious logic, exploitation of system flaw (e.g., trapdoor), or protection mechanism bypass. The attack is considered a fault which may or may not result in an error.

Malicious Logic - Computer hardware, firmware, or software intended to do harm to the system, its data, or the mission being supported.

Object Profile Data - An access control list of processes (programs), the objects for which they are authorized access, and their access type.

Process - A program that has been requested to be executed. It is completely characterized by a single current execution point (represented by the machine state) and address space. The process becomes an entity once it is recognized by the Trusted Computing Base (TCB) that it is potentially to be run (e.g., executed). A process that is not part of the TCB is an internal subject.

Process Profile Data - Identifies legitimate objects (files, resources, and programs) to be accessed by processes and access type.

Program - An object containing potentially executable computer instructions.

Service Assurance - Ensuring availability of a system disrupted by malicious or nonmalicious errors or failures where availability is defined as the computer security characteristic that makes certain computer resources are available to authorized users when needed.

REFERENCES

- [1] AFSSI 5029, Air Force Trusted Critical Computer System Evaluation Criteria, Air Force Special Security Instruction 5029, Air Force Cryptologic Support Center, June 1, 1991
- [2] DoD 5200.28-STD, Trusted Computer System Evaluation Criteria, December, 1985
- [3] Neumann, P.G., "Toward Standards and Criteria for Critical Computer Systems," Compass '90, Proceedings of the Fifth Annual Conference on Computer Assurance, 25-28 June 1990, NIST and IEEE, pp. 186-188
- [4] Beizer, Software Testing Techniques, Van Nostrand, 1983, p. 35
- [5] Joseph, M.K., and A. Avizienis, "A Fault Tolerant Approach to Computer Viruses," Proceedings 1988 IEEE Symposium on Security and Privacy, 18-21 April 1988, pp. 52-58
- [6] Johnson, H.L., "Security Protection Based on Mission Criticality," Proceedings Fourth Aerospace Computer Security Applications Conference, IEEE, December 12-16, 1988, pp.228-232
- [7] Johnson, H.L., C. Arvin, E. Jenkinson, B. Pierce, "Proposed Security for Critical Air Force Missions," Information Intelligence Sciences, Inc, February 15, 1991

- [8] Johnson, H.L., C. Arvin, E. Jenkinson, B. Pierce, "Proposed USAF Approach to Multilevel Criticality," Information Intelligence Sciences, Inc, February 15, 1991
- [9] Clark, D.D, and D.R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," Proceedings of the 1987 Symposium on Security and Privacy, Oakland, CA., April 1987, pp. 184-194
- [10] Boebert, E. and R. Kain, "A Practical Alternative to Hierarchical Integrity Policies," Proceedings 8th National Computer Security Conference, 30 September 1985
- [11] Pozzo, M.M., and T.E. Gray, "Computer Virus Containment in Untrusted Computing Environments," Information Security: The Challenge, preprints of papers from the Fourth IFIP Security of Information Systems Conference, Monte Carlo, December, 1986
- [12] Lunt, Theresa "Survey of Intrusion Detection Approaches," Proceedings 11th National Computer Security Conference, NBS and NCSC, 17-20 October, 1988
- [13] AFR 205-16, Security: Automatic Data Processing (ADP) Security Policy, Procedures and Responsibilities, Department of the Air Force, April 28, 1989

Developing Applications on LOCK*

Richard O'Brien and Clyde Rogers
SCTC
1210 W. County Road E., Suite 100
Arden Hills, MN 55112

Abstract

The Logical Coprocessing Kernel (LOCK) system is a highly assured INFOSEC system that can be used as a platform to develop countermeasures to current and future security threats. In this paper we discuss the manner in which applications are developed on LOCK and the features of the LOCK system that allow these applications to be developed quickly and securely. The paper focuses on the design of such applications using LOCK's type enforcement and the implementation of these applications using the current LOCK software development environment.

1 INTRODUCTION

The Logical Coprocessing Kernel (LOCK) system is a highly assured INFOSEC system that can be used as a platform to develop countermeasures to current and future security threats. The system is based on a trusted computing base (TCB) that satisfies the security requirements defined for the A1 level in the *Trusted Computer System Evaluation Criteria* [1] and includes embedded encryption for media storage. The LOCK design uses a security coprocessor, called the SIDEARM, that makes access decisions based on conventional multilevel and discretionary security mechanisms as well as LOCK's unique type enforcement mechanism. The SIDEARM attaches to a host processor and, together, the two processors define and enforce the system's security decisions [2], [3], [4].

The approach taken in the design of the LOCK system is based on the belief that the threats that a computer system faces are constantly growing. As more secure computer systems are developed, techniques for attacking these systems are also being developed and becoming more sophisticated. In order to counter these new threats, LOCK is based on an open security architecture that allows for the development of additional security countermeasures as the need arises. In this paper we discuss the manner in which applications are developed on LOCK and the features of the LOCK system that allow these applications to be developed quickly and securely. The paper focuses on the design of such applications using LOCK's type enforcement and the implementation of these applications using the current LOCK software development environment. We also describe future enhancements to the software development environment.

In section 2, a brief description of type enforcement is presented, and section 3 then describes some ways in which applications can be designed to take advantage of the enhanced security and integrity provided by type enforcement. A description of LOCKix, LOCK's version of Unix¹, and the manner in which applications can currently be implemented on LOCK using either LOCKix or the LOCK TCB interface is presented in section 4. Future enhancements that will provide additional support for implementing privileged applications are described in section 5, and section 6 gives examples to illustrate these ideas.

*©1991 SCTC. All Rights Reserved.

¹Unix is a registered trademark of AT&T

2 Type Enforcement

LOCK provides a *type enforcement* mechanism, used to restrict the access of subjects (processes) to objects (data) and other subjects. In contrast to discretionary access mechanisms, which can be circumvented, type enforcement supports mandatory controls which provide assurance equivalent to that provided by the multilevel controls. Type enforcement controls are orthogonal to multilevel controls, and provide separation and security both within and across levels. In this section, we present a brief review of the type enforcement concept. More details can be found in [5]. A comparison of the type enforcement mechanism with the ring mechanism of Multics can be found in [6].

The LOCK type enforcement mechanism associates a *type* with each object and a *domain* with each subject on the system. The access a subject is permitted to an object depends on the access capability that the subject's domain is permitted to the object's type. Further, the access a subject is permitted to another subject depends on the access capability that the first subject's domain is permitted to the second subject's domain.

Conceptually, the access a subject has to an object via type enforcement can be thought of as an entry in a data structure called the Domain Definition Table (DDT). The DDT is a matrix with columns indexed by type and rows indexed by domain. Figure 1 shows a portion of a sample DDT and lists the possible capabilities a subject can be granted to an object. The matrix entry in the (d, t) position contains the access capability a subject in domain d is permitted to an object of type t . Similarly, the access capability that one subject has to another subject via type enforcement can be thought of as an entry in a data structure called the Domain Interaction Table (DIT). The DIT is a matrix with columns and rows both indexed by domain. The matrix entry in the $(d1, d2)$ position contains the access that a subject in domain $d1$ is permitted to a subject in domain $d2$. The subject to subject capabilities are: observe, signal, create, and destroy. Trusted capabilities are defined for each access capability that involves modification: trusted write, trusted create, trusted destroy and trusted signal.

The LOCK type enforcement mechanism can be used to solve security problems not addressed by the multilevel and discretionary security policies. It can also be used to develop high integrity subsystems. The manner in which this is done is described more completely in section 3.

3 Designing Applications that Use Type Enforcement

Designing a LOCK application adds a major step to a developer's software design process. Rather than just decomposing the application along functional lines, it must also be partitioned along security and integrity lines. The application designer must identify the components of the application that require added security or integrity, and modularize the application to isolate those components in separate subjects. The collection of subjects that make up an application are called a software *subsystem*.

The design goal is to put each different security or integrity relevant task into its own subject that runs in a distinct domain, and to isolate the data that these subjects must handle into special types. Only the appropriate access capabilities that a subject in each domain requires to perform its task and to communicate with other subjects are assigned to the domain via the DDT and DIT. Then, rather than calling a function to perform a security relevant task, a subject sends a message to an isolated subject designed to perform that task and waits for a return message.

A number of design concerns may require parts of a system to be modularized and isolated. In this section, we discuss some of these concerns and describe how type enforcement can be used to address them.

Subsystem Separation

As part of a subsystem design, special types for subsystem objects and special domains for subsystem subjects are generally defined. The degree and manner of interaction between the subsystem and other subsystems can be rigidly controlled by the DDT and DIT configuration. If

Type Domain	...	UnFl Data	F1 Data	F1 Code	TrP1 Data	TrP1 Code	...	DB Data	DB Code
Pre F1		r, w c, d	-	-	-	-		-	-
F1	-	r	r, w c, d	e	-	-	-	-	-
TrP1	-	-	r	-	r w, tw c, tc	e	-	-	-
.			-	-	r, d	-		-	-
DB	-	-	-	-	-	-	-	r, w c, d	e
.			-	-	r, d	-		-	-

Figure 1: A Sample DDT. Domains are listed down the left-hand side, and types are listed across the top. The capabilities are: r - read, w - write, a - append, e - execute, c - create, d - destroy. Trusted capabilities grant the domain the privilege of violating the *-property in a well-defined fashion for objects of the given type. The trusted capabilities are: tc - trusted create, tw - trusted write, td - trusted destroy. A dash, -, indicates that the domain is not allowed any access to the type.

total isolation of the subsystem files from other system subjects is desired, then the DDT can be configured so that subjects that are not in one of the subsystem domains are not allowed access to objects of the subsystem types. Hence, no subject outside of the subsystem can access the subsystem's data. Similarly, subjects within the subsystem can be prevented from accessing data outside of the subsystem. The DB domain and its corresponding types, DB data and DB code, in Figure 1 is an example of a subsystem that has been completely isolated from the rest of the system by the proper configuration of the DDT.

The DDT and DIT can also be configured so that communication between different subsystems can only occur through a well-defined interface. For example, a subsystem can have a message queue of a special type that provides the only means for subjects outside the system to contact it. Access to this message queue can then be limited to subjects in special domains.

Managing Trust

Trust on the LOCK system has a very specific meaning. It can be used to override the *-property and permit a subject to modify (write, append, create, destroy) a lower level object, or modify (signal, create, destroy) a lower level subject. It is implemented and enforced using the type enforcement mechanism by defining special domains that have *trusted* access capabilities to objects of special types. A subject in one of these domains has the privilege to perform trusted accesses.

Note that only accesses that involve modification have trusted modes. Accesses that involve observing (such as read and execute) have no trusted mode on LOCK. There is no privilege that allows a lower level subject to read higher level data.

LOCK's approach to trust provides a number of design and security advantages. Trust can be granted at a very fine granularity in conformance with least privilege. Since there are separate trusted accesses for each mode of modification, only the access that is required needs to be granted. Furthermore, the DDT can be configured so that these accesses are only granted to special domains and types. That is, for objects trust is granted on a domain-to-type basis, and granting a trusted access to objects of a given type does not mean that such access is also granted to objects of other types. Similarly, for subjects, accesses are granted on a domain-to-domain basis. Hence, even if a subject has a trusted access, it can only use this access on objects of the indicated type, or subjects of the indicated domain. Since those subjects that use trust are specifically identified and isolated, a least privilege policy with respect to the use of trust can be implemented.

(In this paper we use the term *privileged* subject to indicate a subject that is intended to perform some security or integrity critical function. This is what often is called a *trusted* subject. We use the term *privileged*, rather than *trusted*, to avoid confusion with the more restricted notion of trust, described above, that involves the ability to override the *-property of the Bell and LaPadula model. We will restrict our use of the phrase *trusted subject* to indicate a subject whose domain has a trusted access capability.)

Separation of Duties

Within a subsystem, the LOCK type enforcement mechanism allows a strict *least privilege* policy to be implemented and enforced. In order to take advantage of this capability, the subsystem must be designed in a modular fashion that isolates privileged functionality in separate modules. These modules can then be implemented as separate subjects, each in its own special domain, and the data that they access can be assigned special types. The assured pipelines, described in the next section, are examples of such design. The DDT and DIT can be configured to allow only the least amount of access necessary for the desired functionality. In particular, individual subsystem modules can be prevented from accessing data or communicating with other subsystem subjects in ways that are unnecessary for the proper function of the module.

Such a design allows for simple modifications and additions. Adding a new subject to perform a new task is a localized operation, so its effects on system security and integrity can be easily identified. Also, such a design simplifies assurance work by identifying and isolating security and integrity critical subsystem portions. The primary assurance effort can then be directed toward only those subjects that perform privileged tasks.

Unbypassable Filters

The type enforcement mechanism also provides a means for implementing high integrity operations. By using special domains and types, filter processes can be created to strictly control the manner and order in which certain operations are performed. As figure 2 indicates, these filters have the three critical properties of a reference monitor. They are unbypassable, tamperproof, and can be verified correct. These properties are implemented by the definition of the necessary types and domains and by the correct configuration of the DDT. In Figure 1, the F1 and the TrP1 processes are examples of unbypassable filters.

By composing one or more such filters, *assured pipelines* can be constructed that ensure the security and enhance the integrity of data that flows through the pipeline. This is illustrated in figure 3. Assured pipelines and the LOCK concept of a role, described below, can be used to implement a variety of integrity policies, including those proposed by Clark and Wilson [7], [8]. One application of an assured pipeline might be to guarantee that any modifications to user records must pass through a previewer pipeline before the modifications are committed. This previewer pipeline allows the user to review and commit the changes using filter processes that have been assured to maintain certain integrity properties of the records.

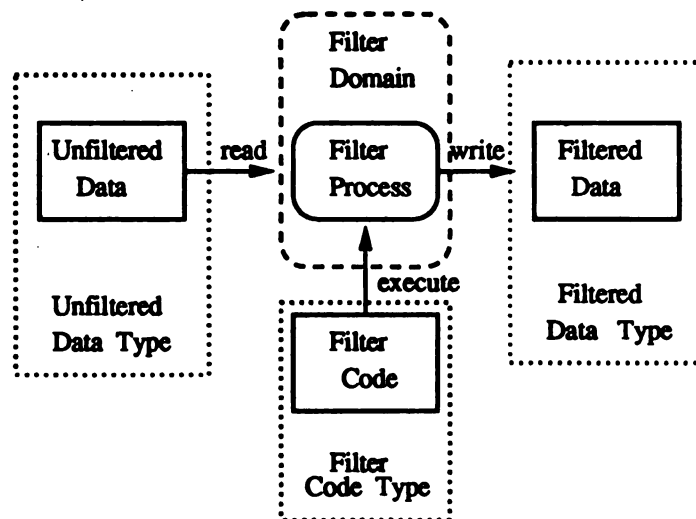


Figure 2: A filter process. The filter reads the unfiltered data and performs its filtering operation before writing the data to a new object of a different type. A special domain (the Filter Domain) is created for the filter process and special types are created for the unfiltered data (Unfiltered Data Type), the filtered data (Filtered Data Type), and the filter code (Filtered Code Type). By using the DDT to restrict create and write access to Filtered Data Type objects to the Filter Domain, the filter process is made unbypassable—it is only through the Filter Domain that filtered data can be produced. By allowing the Filter Domain execute access to only objects of Filter Code Type and by not allowing any other domain create or write access to objects of Filter Code Type, the filter process is made tamperproof. (There is no way to modify the code that it executes.) By having only one object of type Filter Code Type and performing the desired assurance on that code object, the filter process can be verified to perform its filtering process correctly.

Roles

In the LOCK system, user roles are implemented in a manner that relies on the use of types and domains. Every subject is associated with a user. A Role Authorization Table is used to determine in which domains each user is allowed to have subjects operating. Roles are represented as sets of domains, and a user is allowed to operate in a particular role (or subrole) only if the Role Authorization Table permits the user to have subjects in the domains associated with that role (or some subset of these domains).

To extend the example from the previous subsection, the Role Authorization Table can be configured so that only users identified as System Security Officers (SSOs) are allowed the ability to have subjects in the previewer filter domain. In this way, only an SSO is allowed to modify LOCK user records.

4 Implementing LOCK Applications

After developing a design that takes advantage of LOCK's type enforcement mechanism, the next step is to implement the design on LOCK. LOCK currently provides two interfaces for software development. For applications requiring no assurance, a fully functional Unix interface, LOCKix, is provided. Privileged applications, on the other hand, must be implemented on the TCB interface directly. A third interface that allows privileged applications to be developed on a LOCKix style interface is under development. It is described in Section 5.

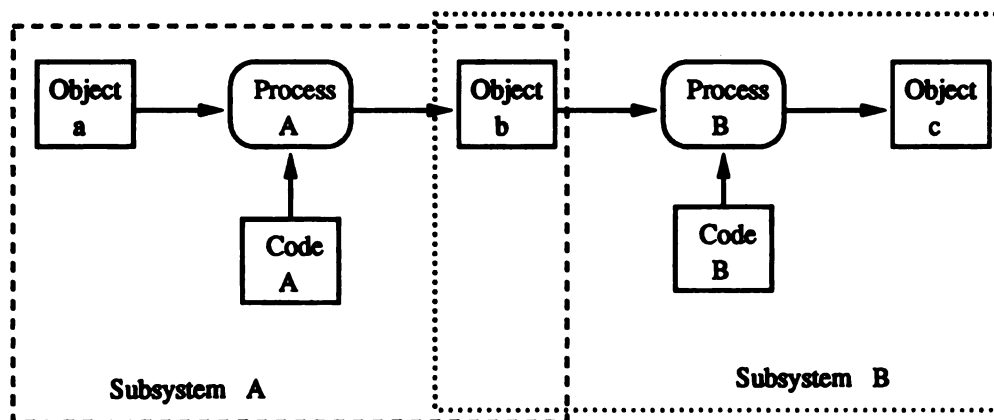


Figure 3: An Assured Pipeline. This example of an assured pipeline is composed of two filters, each designed and assured to perform its particular function. Process A filters the data in object a and places it into object b. Object b is a shared object used to construct the pipeline. Process B filters data from object b and places it into object c. An example of an assured pipeline might be a labeler process (process A) followed by a printer process (process B). Object c in this case would be the output from the printer. The labeler process would be assured to correctly label the data and put the labeled data in object b. The printer process would be assured to print the data it receives correctly.

4.1 Implementing Unprivileged LOCK Applications

For unprivileged software that does not need to communicate with other LOCK subjects, a developer can use the LOCKix programming environment. LOCKix is an unprivileged application providing a Unix interface on top of the LOCK TCB. It provides a fully functional single level Unix kernel with read only access to files at dominated levels. LOCKix is based on Unix System V, Release 1 and is over ninety percent system call compatible with Unix System V Release 2 as measured by the System V Verification Suite. The next release of LOCKix will be based on System V, Release 4.

LOCKix supports a C compiler, 68000 assembler, loader and C library. It also has program development utilities such as an archiver and "make", and runs many existing Unix programs with little or no modification. Most of the modifications required, in fact, are corrections of hardware dependent programming errors in older programs. Most modern Unix code ports reasonably easily. LOCKix provides a familiar programming interface and Unix library support.

The LOCK type enforcement mechanism allows a great deal of flexibility in controlling use of the LOCKix compiler. LOCK can be configured so that only users privileged to run LOCKix in a special domain can create objects that the LOCK host can execute. This prevents unauthorized users from creating LOCK executable code.

LOCKix currently does not support a debugger, but will at some time in the future. The present lack of a debugger makes LOCKix a less than ideal environment for program development. Further work is also needed to develop high level support for inter-subject communication. Multiple processes running inside the same LOCKix session communicate like any Unix process. However, no Unix library support currently exists to enable LOCKix processes to communicate with other LOCK subjects. LOCKix currently does not have a library interface to the LOCK TCB (although creation of one is planned), so the direct TCB calls currently required for inter-subject communication must be made in assembly language.

Because LOCKix presents a compatible Unix interface, the current development approach is for application developers to write, debug, test and run applications on their favorite Unix system,

and then, once the application is ready for use, simply recompile and run it on LOCKix. This approach has been used with great success in porting Unix software to LOCKix. Portable software (Kermit, some GNU software, etc.) has been compiled and run on LOCKix without modification.

While the best method of implementing most unprivileged applications on LOCK is to develop them to run on LOCKix, there may be some applications that would require little or no support from LOCKix. Such applications could be implemented directly on the LOCK TCB interface used for privileged software. The method by which this is done is discussed in the following section.

4.2 Implementing Privileged LOCK Applications

Privileged software cannot depend on LOCKix as the underlying system because it is large and unassured, and if subverted, could cause the privileged software to be subverted also. Privileged software must be developed to run directly on the native LOCK TCB. The TCB provides a small set of well understood, well behaved primitives providing simple memory and communication facilities. The simplicity and power of the LOCK TCB interface makes development of sophisticated, multi-level assured applications possible.

Privileged (and some unprivileged) applications have been developed using the library interface to the TCB. A full set of routines for inter-subject communication, memory management, device handling, signaling and more are available in this library. A set of Unix stubs that simulates most of these library routines has been developed so that the first phase of debugging can take place on a Unix system, using its program development utilities. The LOCKix C compiler cannot be used to compile the code because it cannot generate the fully relocatable code required to run on the native TCB. LOCK TCB code is generated using a cross compiler. Once code is moved to the TCB, it must be integrated using a hardware level debugger. A software based debugging capability should be available some time in the future.

5 The Future of LOCK Software Development

In future LOCK systems the goal is to provide a complete software development environment in which both privileged and unprivileged software can be developed in the same manner and with the same ease. This section describes some of the ideas and enhancements that will make such an approach possible.

5.1 Features of the Software Development Environment

An Isolated Development Environment. The LOCK type enforcement mechanism can be used to create insulated test environments for development of privileged applications. By insulating the development and test environment, it becomes reasonable to develop and test privileged applications using LOCKix. For example, when testing a text downgrader a special LOCKix domain can be created that has read and trusted write access capabilities to a special type of test object. That domain would not be able to read or write any other type of object, and other domains would only be able to destroy that type of object. This allows controlled creation of a high level object that contains no high level information, which can be safely downgraded during testing. This way when testing the downgrade function, the domain restriction keeps any information from being accidentally or deliberately downgraded during testing. Less critical software can be developed in less insulated domains with fewer controls.

An Assurable Unix Interface Library. The LOCK assurable Unix interface library is a small subset of the Unix system call interface. This library will provide developers with access to a simple Unix file system, allowing them to specify object identifiers using a pathname. System calls providing file manipulation, interprocess and inter-subject communication, and signal management will also be provided.

This library will provide an interface for privileged software to be compiled using a subset of Unix system calls. The need for TCB interface stubs will be eliminated, and programming privileged software will be simplified due to the more familiar Unix interface.

Run-Time Environment Enhancements. The format and execution of TCB subjects will be changed to add support for non-relocatable program code. Also, the TCB subject calling conventions will be enhanced to provide arguments and environment information in a manner similar to a Unix system. With these enhancements, development of privileged software can take place in LOCKix without special support tools.

Complete Software Development Toolset. Another major improvement will be the addition of a standard Unix source level debugger to LOCKix. This will make it possible for developers to debug LOCKix code while running in LOCKix, and will complete the LOCKix programmer's toolset.

5.2 Using the Software Development Environment

With a complete toolset in place, a more Unix-like TCB run time environment, the assured Unix interface library, and the proper use of type enforcement, LOCKix will become an effective platform for developing both unprivileged and privileged LOCK applications. Developers will be able to write and test assured software in a special domain insulated from regular system users. In such an environment they can compile and run privileged (and unprivileged) code until they are satisfied with its correctness.

Applications would then be moved from a development environment to a production environment via an assured pipeline. The source code written in the assured software development environment would be of type *assurable code*. To compile this code in a format executable by general LOCK users, it would have to be reviewed using a privileged source code reviewer that runs in a domain that can read objects of type *assurable code* and can write objects of type *reviewed code*. The LOCKix compiler would then be run in a special domain that can read objects of type *reviewed code*, and write objects of a type that can be executed outside of the insulated development domain. Different review steps could be added or deleted as required by individual sites.

This model allows for controlled transition of software from development to operational status. It supports role separation, allowing sites to separate the roles of software developer, reviewer and installer. It uses many of the features of type enforcement to provide a secure, controlled environment for the complete LOCK application development cycle.

6 Examples

In order to illustrate the manner in which critical applications can be designed and implemented on LOCK, we present some examples.

Example 1. A Privileged Subsystem.

For our first example consider a subsystem that is designed to run as a single privileged subject on top of a TCB. Such a subsystem might be a multilevel DBMS that performs all of its processing at system high and then downgrades the results.

If the design is such that the entire DBMS cannot be easily decomposed into modules, some of which need to be privileged and others that do not, then the full advantage of type enforcement cannot be gained. However, it is still desirable to create a special DBMS domain, in which the DBMS privileged subject would run, and special types for the DBMS files. It would then be possible to run the DBMS as an isolated subject on the system as described in figure 1.

Although implementing the subsystem directly on a version of the standard LOCKix system might be very easy, this approach has the disadvantage that since LOCKix is unprivileged, if it is corrupted, the privileged subsystem might be subverted. This danger can be mitigated by using a small, well-understood LOCKix subset, that only supports the functionality required by the DBMS, on which to perform the port and then configuring the DDT so that this code object can not be modified and so that it is the only code object that can be executed from the special DBMS

domain. Note that if the application is properly isolated, even if it is subverted, it can only affect information available within its subsystem.

Of course, the danger that the LOCKix subsystem can be subverted is always present. To insure that the subsystem cannot be compromised, it would be necessary to implement it using either the LOCK TCB interface or, in the future, the assurable Unix interface. If it provides the required support that the privileged subsystem needs, then the assurable Unix interface is probably the best choice, since the implementation would be easier. In fact, if minimal additional support is required, it might be desirable to add this support in an assured manner. In this way additional functionality can be added to the assurable Unix interface in an incremental manner.

Example 2. A Modularized Subsystem

The real advantage of type enforcement is only obtained when a subsystem is designed so that its security and integrity components are separated into modules which are small enough so that the corresponding code can be properly assured for correctness. This highly reliable code, and the data it deals with, can then be isolated using special domains and types and the remainder of the subsystem can be implemented as an unprivileged subject.

As an example consider a multilevel DBMS design, such as LDV [9], [10] in which all privileged processing can be isolated in a few modules, and most of the DBMS functionality is unprivileged. The unprivileged portion of the DBMS could be implemented on standard LOCKix using standard database code. This might involve nothing more than compiling the code on LOCKix. The privileged portions would be implemented as discussed in the previous example, using either the LOCK TCB or the assurable Unix kernel. The LDV design is an example of an assured pipeline, since any query first passes through the unprivileged DBMS, then the privileged filter that determines what information can be released, and then the response passes back out through the unprivileged DBMS.

Example 3. A Role Based Subsystem

Examples 1 and 2 illustrate the manner in which unprivileged and privileged software can be ported to LOCK to take advantage of LOCKix and type enforcement. To illustrate how roles can be implemented, consider a simple example in which a DBMS is used to create reports which must be reviewed by a human before they are released. The DBMS may itself be unprivileged, but the previewer subject that handles the review processing is privileged to correctly display the report, so that a user can review it, and only release it if it passes review. In effect, the previewer acts as a filter. Furthermore, the role of the reviewer is only allowed to certain privileged individuals.

This subsystem can be implemented on LOCK in much the same way as described in Example 2 with the previewer being put in a special domain that acts as a filter between objects of type *report* and objects of type *reviewed report*. The role of the reviewer is then implemented by using the Role Authorization Table. Only users who are allowed to be reviewers are permitted to have subjects that execute in the previewer domain. Hence, only these users are allowed to perform the role of a reviewer.

7 Conclusion

This paper has discussed some issues involved in designing and implementing an application on the LOCK system, and some of the features LOCK provides to aid application development.

LOCK's type enforcement mechanism allows an application designer to decompose an application into modules which can then be separated into separate domains and types allowing the interaction between the modules to be strictly controlled. Type enforcement also allows separation of duties, simplified trust management, creation of assured pipelines and role enforcement.

Once an application is designed, LOCKix and the TCB interface provide tools a developer can use to build privileged and unprivileged applications. In the future, LOCKix will add functionality

to its software development environment, and the assurable Unix interface will allow privileged modules to be implemented in an even more efficient manner.

References

- [1] "Trusted Computer System Evaluation Criteria", Department of Defense Standard 5200.28-STD, December 26, 1985.
- [2] O. Sami Saydjari, J. Beckman and J. Leaman, "LOCKing Computers Securely", *Proceedings of the 10th DoD/NBS Computer Security Conference*, NBS, 1987, pp. 129-140.
- [3] W.E. Boebert, "Constructing an Infosec System Using the LOCK Technology", *Proceedings of the 11th National Computer Security Conference*, NBS, 1988, Postscript Volume.
- [4] O. Sami Saydjari, J. Beckman and J. Leaman, "LOCK Trek: Navigating Uncharted Space", *Proceedings of the IEEE Symposium on Security and Privacy*, 1989, pp.167-175.
- [5] W.E. Boebert and R.Y. Kain, "A Practical Alternative to Heirarchical Integrity Policies", *Proceedings of the 8th National Computer Security Conference*, NBS, 1985, pp. 18-27.
- [6] W.E. Boebert and Steven M. Miller, "Comparison of Mechanisms for Implementing Protected Software Subsystems", unpublished draft.
- [7] D.D. Clark and D.R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," *Proceedings of the 1987 Symposium on Security and Privacy*, 1987, pp. 184-194.
- [8] D. J. Thomsen and J. T. Haigh, "A Comparison of Type Enforcement and Unix Setuid Implementation of Well Formed Transactions," *Proceedings of the 1990 Computer Security Applications Conference*, 1990, pp. 304-312.
- [9] J.T. Haigh, R.C. O'Brien, and D. Thomsen, "The LDV Secure Relational DBMS Model", *Proceedings of the 1990 IFIP Database Workshop*, to appear.
- [10] P.D. Stachour and B. Thuraisingham, "Design of LDV: A Multilevel Secure Relational Database Management System", *IEEE Transactions on Knowledge and Data Engineering*, 1990, Vol.2, pp. 190-209.

THE DEVELOPMENT OF A LOW-TO-HIGH GUARD

**Michelle J. Gosselin
The MITRE Corporation
Burlington Rd., Bedford, MA 01730**

ABSTRACT

This report details the development of a guard to monitor electronic traffic between two computer systems. The guard is intended to operate between two computer systems that are accredited to operate at different security levels. One system (the high system) must be accredited to process all information on the other system (the low system). The purpose of the guard is to automate the delivery of information from the low system to the high system while preventing any flow of information from the high system to the low system.

ACKNOWLEDGEMENT

This paper is based on work done for the Air Force, ESD-ICD. Their support in this effort is greatly appreciated. Many co-workers at the MITRE Corporation are also to be thanked for their contributions to this effort. Specifically, Karen Johnson is thanked for implementing major pieces of this design.

INTRODUCTION

The low-to-high security guard described in this report was developed for two specific systems accredited at two specific security levels. However, the design of the guard allows for the guard to be easily applied to other environments with systems operating at different security levels involving a low-to-high information flow.

The purpose of the guard is to automate the delivery of data from the low system to the high system while preventing any flow of information from the high system to the low system.

The development of the guard occurred in three phases. The first phase involved the selection of the guard platform. Once the platform was chosen, the operational concept of the guard was defined. The third and final phase was the actual implementation of the operational concept. Each of these phases is discussed in detail in this report.

Also discussed in this report are the issues that must be addressed before the guard can be accredited for operation.

GUARD PLATFORM

SELECTION

The first task in developing the guard was to select a suitable hardware and software platform. There were three requirements that were considered when selecting the guard platform. These requirements are stated in the following list.

- 1. The guard should be a low cost system with a maximum cost near \$10,000.**
- 2. Because the guard processes sensitive data, a guard must satisfy certain security requirements mandated by the Department of Defense (DOD) in DOD Directive**

5200.28. According to Directive 5200.28, a system that processes classified information and that requires controlled access protection must meet, at a minimum, the C2 class of security requirements specified in the DOD Trusted Computer System Evaluation Criteria (TCSEC). The operating mode and the level of trust that is required for the guard is determined by risk assessment which is determined by the minimum clearance of the users and the maximum data sensitivity. For systems of different security levels, B class services that provide mandatory access control and can separate different security levels are required. The directives that govern the interfacing of intelligence systems with nonintelligence systems generally require that the guards be multilevel and provide a B2 level of trust as a minimum.

3. Since the purpose of the guard is to prevent the transmission of data from high to low, the guard must be capable of keeping any network traffic intended for the low system separate from any network traffic intended for the high system. One method of keeping the traffic separate is for the guard to use separate network protocol stacks and Ethernet cards to interface with each system. The configuration of such a system is displayed in figure 1.

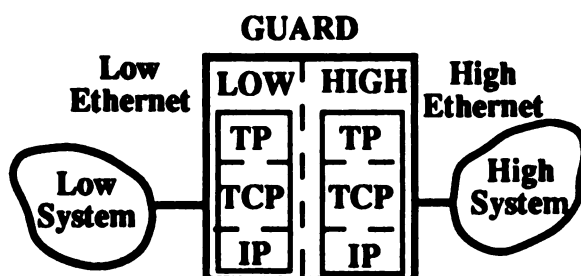


Figure 1. Configuration With Separate Ethernet

There are several systems that meet or surpass the C2 class of security requirements. However, some of these systems far exceed the cost limit of \$10,000, and many of the systems do not have separate protocol stacks. The only system that meets the above criteria is the Trusted Xenix operating system produced by Trusted Information Systems (TIS). Trusted Xenix has received a B2 rating, which surpasses the required C2 rating. The operating system runs on a variety of hardware platforms, including many INTEL 80286 and 80386 based workstations. Trusted Xenix and an appropriate hardware platform can be purchased for approximately \$10,000.

TIS has developed a version of Trusted Xenix that implements two separate instantiations of the networking protocols. Two separate Ethernet cards (and ports) are also supported. Traffic can be kept separate by regulating the access to the protocols and to the ports. The method in which access is regulated is explained in detail in the following section.¹

SECURITY POLICY

Trusted Xenix is a Unix-based operating system. The operating system has several types of objects to which information can be written. These objects include files, directories, and

¹ Unfortunately, the network interfaces are not included in the evaluated configuration. This matter will have to be addressed prior to or during the accreditation process.

ports. A directory can contain files and other directories. A port can be used to transmit information to devices external to the workstation.

Actions on the operating system objects are performed by processes. For instance, a process can read information from and write information to a file, a directory, or a port. A process can also initiate other processes. An example of a process is the networking process, referred to as the *inet* daemon, that handles information coming into the system from the network. On the guard, there are two *inet* daemons; one for the low side and one for the high side.

In the Trusted Xenix environment, each object and process has a sensitivity label associated with it. The sensitivity label represents the sensitivity of the data contained in the object or process. A sensitivity label is composed of two pieces; a hierarchical classification and a set of nonhierarchical categories. A sensitivity label S1 dominates a sensitivity label S2 if the classification of S1 is higher than or equal to the classification of S2 and the category set of S2 is a subset of the category set of S1.

The security policy of Trusted Xenix is composed of two pieces; a read policy and a write policy. The read policy states that a process can read an object if the sensitivity label of the process dominates the sensitivity label of the object. The write policy states that a process can write to an object (which includes creation and deletion) if the sensitivity label of the process equals the sensitivity label of the object. Trusted Xenix enforces strict adherence to these policies. For a process to override these policies, the process must possess special privileges granted by the System Security Officer using mechanisms provided by Trusted Xenix.

The security policy of Trusted Xenix also applies to the unprivileged processes associated with the guard software. The unprivileged processes of the guard software do not have a separate security policy, and are forced to follow the policy established by Trusted Xenix. However, there is one privileged guard process that does violate the Trusted Xenix security policy. This process is privileged to append one byte of information to a file that exists in a directory that is at a lower security level than the level of the process². The process does not otherwise violate the Trusted Xenix security policy.

OPERATING CONCEPT

Before discussing the details of the guard software, it is necessary to understand the operating concept of the guard. The operating concept is based on the following scenario: the low system, which is accredited to process SECRET A data, sends a database update to the high system, which is accredited to process SECRET A/B data. The high system then provides either an acknowledgement or a negative acknowledgement of the receipt of the database update. The operating concept can be separated into two phases. Phase I involves the transfer of the database update from low to high. Phase II involves the transfer from high to low of an indication of either having received (an acknowledgement) or not received (a negative acknowledgement) the database update. Phase I of the operating concept is depicted in figure 2, and phase II is shown in figure 3. In both figures, processes are represented by circles, and files (the database update and the acknowledgment or negative acknowledgement) are represented by squares.

² The privileged process is discussed in detail in the paper.

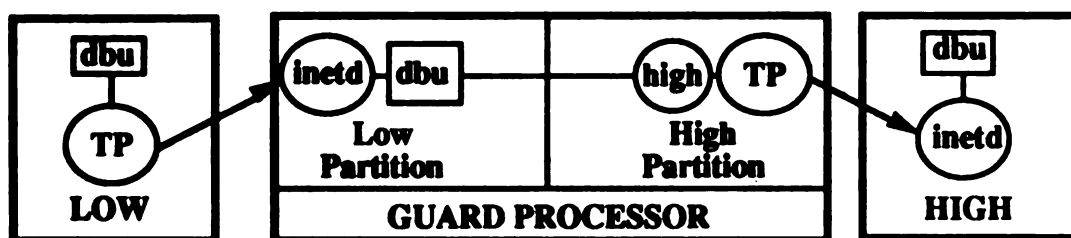


Figure 2. Operating Concept for File Transfer

The operational sequence of transferring the file is as follows. An authorized user of the low system will initiate a file transfer process using an application layer transfer protocol (TP), such as Simple Mail Transfer Protocol (SMTP) or File Transfer Protocol (FTP). The *inet* daemon (*inetd*) on the low side of the guard will download the database update (*dbu*), in the form of a file, to the guard platform in the low partition (or directory).

Once the *dbu* file is stored on the low partition, the guard software performs various actions (such as reading and copying the file) in order to transfer the file to the high system. The guard software is separated into three different processes: a low process, a guard process, and a high process. The first process that takes any action on the file is the high process. When the high process detects the presence of the file in the low partition by reading the contents of the low partition, the high process initiates a TP connection with the high system. Once the connection has been established, the file is transferred across the TP connection to the high system.

After the file has been successfully transferred, an acknowledgment of the successful transfer is sent back to the low system from the high system via the guard software. If the file was not successfully transferred, a negative acknowledgment is sent back to the low system from the high system via the guard software. The operating concept of the transfer of the acknowledgment (ack) or negative acknowledgment (nack) is depicted in figure 3.

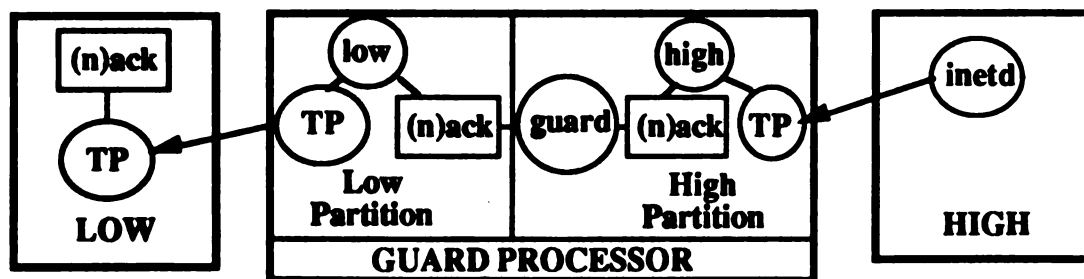


Figure 3. Operating Concept for Transfer of (N)ack

The operational sequence of transferring the (n)ack is as follows. The high process determines from the information received from the high *inet* daemon whether or not the *dbu* file transfer was successful. If successful, the high process creates an acknowledgment, in the form of a file, of the successful transfer on the high partition. If not successful, the high process creates a negative acknowledgement, in the form of a file, to indicate the failure on the high partition.

The guard process, after detecting the presence of the (n)ack file on the high partition creates another (n)ack on the low partition by appending information to the original *dbu* file. Once

the acknowledgment has been created, the guard process deletes the original (n)ack file on the high partition.

Finally, the low process detects that the size of the (n)ack file (previous dbu file) has increased and transmits the (n)ack file to the low system. The low process then deletes the (n)ack file on the low partition.

IMPLEMENTATION

The guard software is composed of three continuously running processes; a low process, a guard process, and a high process. Each of these processes is automatically started when the system is started. Each process is cyclical in that it executes a sequence of steps and then returns to the first step. The first step for each process is to detect the presence of a particular file. If the file is not detected, the process temporarily suspends execution for a configurable period of time (currently ten seconds). After this delay, the process again starts at the first step. The low process runs at the SECRET A level, and both the guard process and the high process run at the SECRET A/B level. The guard software also works in conjunction with an inet daemon dedicated to the low side and an inet daemon dedicated to the high side. Each of these processes is described in detail in this section.

TRANSFER OF THE DBU

Figure 4 depicts the actions that are taken on the incoming dbu file. Also provided in the figure is the resulting sensitivity labels on both the dbu file (F) and the process (P) involved in the action.

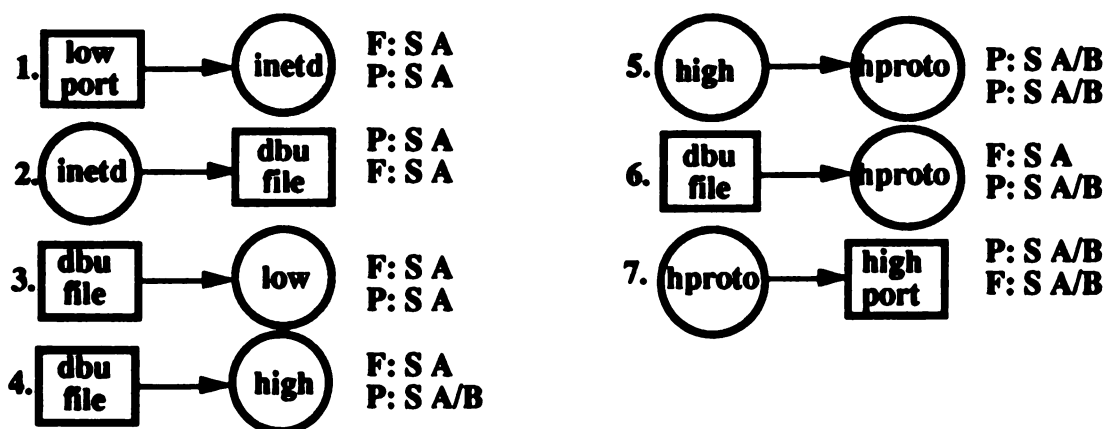


Figure 4. Transfer of the DBU

The dbu file arrives in the following manner on the system. The low user sends a file to the guard via a transport protocol. The only naming convention that the low user must follow is to not begin the name of the file with the character ".". When a low user sends data to the guard, the data comes across the low Ethernet and through the port dedicated to that Ethernet. This port is labeled SECRET A. When data comes across this port, the low inet daemon, which is labeled SECRET A, reads the data.³ The *inet* daemon then writes the data to a file

³ As stated previously, a port is an object that contains data. The data that the object contains is the data that is coming across the port. Therefore, the data coming across the port has the same sensitivity label that the port does.

in the low directory which is labeled SECRET A. This file is created at the SECRET A level.

The low process checks the low directory for a file containing a database update. The search of the low directory is allowed since both the low process and the low directory are SECRET A. If a database update (dbu) file is detected, the low process records the size of the dbu file.

The high process also checks the low directory for a file containing a database update. The search of the low directory is allowed since the high process is SECRET A/B and the low directory is SECRET A. If there is a dbu file, the high process initiates a new process that executes the commands found in a script file called *hproto* (for high protocol).

The *hproto* script file contains a list of executable commands. This script file is entirely tailorable to the specific environment that is hosting the guard. The script can be changed without having to recompile any of the guard software. This allows for complete freedom in choosing any of the TCP/IP based protocols to be used for file transfers. This includes FTP, SMTP, and rcp. Currently, the guard software calls a version of *hproto* that uses FTP as the transfer protocol. Hproto issues the appropriate TP command in order to transmit the dbu to high using the protocol stack dedicated to the high side.

After the transfer of the dbu file, the creation and transfer of the (n)ack file takes place as described in the following section.

TRANSFER OF THE (N)ACK

Figure 5 depicts the actions that are taken on the (n)ack file. Also provided in the figure is the resulting sensitivity labels on both the file (F) and the process (P) involved in the action.

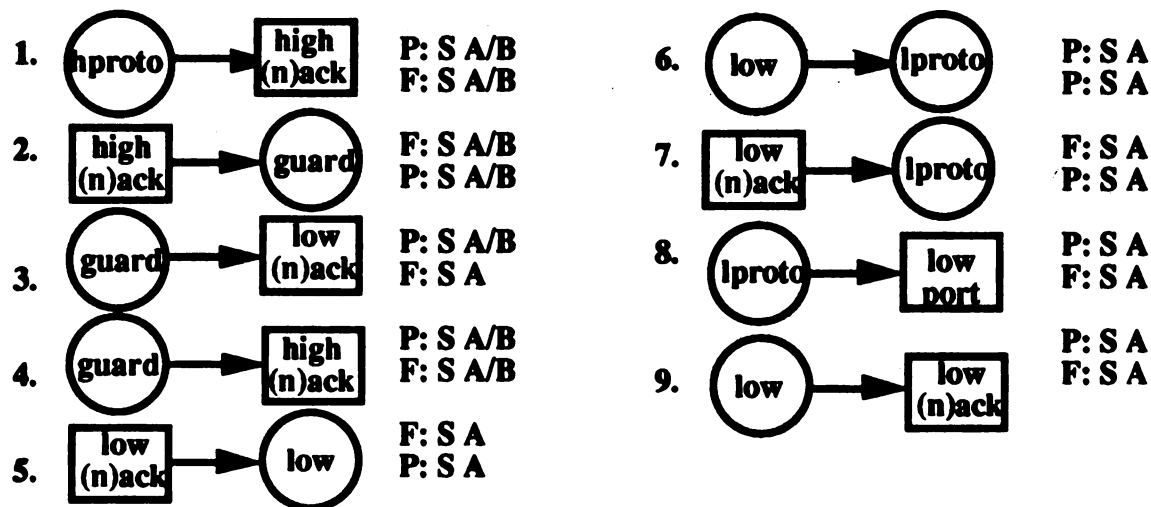


Figure 5. Transfer of the (N)ack

As the dbu file is being transferred, the data sent back from the inet daemon on the high host is then analyzed by *hproto* to determine if the file was successfully sent. If the transfer was successful, an ack file is created in the high directory with the name of the dbu file in the low directory. This ack file contains one byte that has a value of 0.

If the transfer was not successful, attempts are made to retransmit the file for a configurable number of times. If the maximum number of retransmission attempts is reached, the high process creates a nack file in the high directory with the name of the dbu file in the low directory. The nack file contains one byte that has a value of 1.

Once an (n)ack has been received, the next stage of the (n)ack transfer cycle involves the guard process.

The guard process periodically checks for (n)acks in the high directory. The guard process determines that a (n)ack exists when there is a file with a size of one byte. If a file is greater than one byte in length, a violation has occurred, and the violation is audited⁴.

Upon detection of a (n)ack, the guard process attempts to open a file in the low directory that has the same name as the (n)ack. If no such file exists, a (n)ack file exists that does not have any corresponding dbu file. Therefore, a violation is indicated, and the violation is audited.

If the guard process finds a corresponding dbu file, the guard process reads the byte of information in the (n)ack file in the high directory. If the byte is a 0 (ack) or a 1 (nack), the high process appends the byte to the dbu file in the low directory. This appended dbu file is now considered the low ack file. The guard process then deletes the (n)ack in the high directory. The writing of information to the SECRET A dbu file by the SECRET A/B guard process requires a special security privilege granted by the operating system to override the write policy.

If the byte in the high (n)ack file holds a value of other than a 0 or a 1, a violation is indicated, and the violation is audited. The (n)ack in the high directory is deleted.

The next stage of the database and (n)ack transfer cycle involves the low process.

The low process periodically checks the low directory for a file containing a (n)ack. The low process determines that a (n)ack exists when the size of a file has increased by one byte. If a (n)ack is found, the low process reads the file, which is labeled SECRET A, for the additional byte of information. The low process then deletes all previous information from the file. Once the low process has modified the (n)ack file to contain only the byte of information from the guard process, the low process initiates a new process that executes the commands found in a script file called *lproto* (for low protocol).

The *lproto* script file contains a list of executable commands. This script file is entirely tailorable to the specific environment that is hosting the guard. The script can be changed without having to recompile any of the guard software. This allows for complete freedom in choosing any of the TCP/IP based protocols to be used for file transfers. This includes FTP, SMTP, and rcp. Currently, the low process calls a version of *lproto* that uses FTP as the transfer protocol. *Lproto* issues the appropriate TP command in order to transmit the (n)ack to low using the protocol stack dedicated to the low side. low must determine if the file is an ack or nack based on the contents of the file.

Once the commands in the *lproto* script file have been executed, the low process deletes the the acknowledgment file.

⁴ All audit records includes the date and time when the violation was detected, the name of the (n)ack file, and the contents of the (n)ack file.

ACCREDITATION

For the low/high guard to be used operationally, it must first be certified and accredited. This section discusses the issues that need to be addressed in certifying and accrediting the guard.

ACCREDITATION AND CERTIFICATION PLAN

To properly accredit the low/high guard for operation, an accreditation and certification plan should be written for the guard. The primary purpose of an accreditation and certification plan is to serve as a handbook for the Information System Security Officer (ISSO) and the Designated Approving Authority (DAA) in carrying out their roles in the accreditation and certification process. An accreditation and certification plan usually provides the following information:

1. A system overview describing the operational environment,
2. Security requirements the system must meet,
3. Documentation that must be supplied to the ISSO and/or DAA,
4. Organizational responsibilities, and
5. A detailed description of the accreditation and certification process.

EVALUATION

As stated previously, the guard base (Trusted Xenix on a 286 or 386 machine) has received a B2 rating from the National Computer Security Center (NCSC). However, changes that have been made to this base to implement the guard effect this rating. Both the networking software and the guard software have to be analyzed for their effect on the overall rating.

Networking Software

NCSC currently does not evaluate any networking software and accompanying hardware. Each system under evaluation is treated as a standalone system. Therefore, the networking software used by the guard may not meet the NCSC B2 security requirements. During the accreditation and certification process, the networking software resident in the kernel of the operating system would have to be inspected by the ISSO and the DAA in order to determine whether or not the software was trustworthy enough for the environment. The networking software would have to be analyzed for covert channels and for its effect on the trusted operating system. According to TIS, there are two separate protocol stacks that can be labeled separately. These separate protocol stacks and labels allow the separation between low and high to be maintained. The ISSO and DAA would have to verify that there are indeed two separate stacks.

Guard Software

Since the guard software is given the privilege to violate the security policy of the operating system when creating the acknowledgment in the low directory, this software must also be inspected. However, inspecting the guard software is a much easier task than inspecting the networking software since the privileged portion of the guard software consists of six lines of code. Inside the guard software, the following steps are taken:

1. A privilege to override the mandatory access control policy is granted.
2. A file in the low directory is opened for writing.

3. A byte of information, containing either a 0 or a 1, is appended to the file.
4. The file is closed.
5. The privilege to override the mandatory access control policy is revoked.

These five lines of code could quickly be analyzed with respect to the B2 requirements, if the desire is to maintain a B2 system. One of the B2 requirements that may pose a problem is the covert channel analysis requirement. However, if the action of creating an acknowledgment file does create a covert channel, the channel could easily be reduced to an acceptable size by limiting the rate at which acknowledgments are created by the guard process.

OPERATIONAL ENVIRONMENT

The operational environment of the guard must also be considered when accrediting the guard. Aspects of the environment that must be considered are the location, additional uses of the guard, and the users.

Location

The guard should be located in a facility where access is restricted to individuals who are allowed to process SECRET A/B data.

Guard Uses

The guard is intended strictly for use as a guard. No application packages, such as word processors and spread sheets, are resident on the guard. Without any application packages, the desire to use the guard for other purposes will be minimal. By limiting the use of the guard, the number of accidental security infractions will be limited.

Users

There are two groups of users that have accounts on the guard. One group consists of the security personnel who are responsible for maintaining the system. The security personnel include a security administrator, an auditor, and an operator. The security administrator maintains user accounts and is responsible for the security of the system. The auditor analyzes the audit trail, and the operator performs day-to-day operations, such as systems backups.

The other group of users are general users who do not have system responsibilities. There is one general user with an account on the guard, which is the guard user. The guard user is the owner of the low process, the guard process, and the high process respectively. Since these processes are automatically started when the system is turned on, there is no need for this user to log onto the system. Therefore, for security purposes, the guard user should be administratively prevented from logging in.

By limiting the number of users of the guard, the number of intentional security infractions will be limited.

CONCLUSION

The guard documented in this paper provides for the automated transfer of a database update from a low system to a high system. The guard also automatically relays an acknowledgment of a successful transfer or a negative acknowledgement if the transfer was not successful back to the low system.

The guard also prevents the flow of information from the high community to the low community. There are two types of the high-to-low information flow; the high side writing to the low side, and the low side reading from the high side. The way in which these flows are prevented is as follows. For a user of the high side to gain access to the low side, the user must first gain access to the guard. Access can only be gained through the Ethernet card on the high side. The port associated with this Ethernet card is labeled SECRET A/B. Therefore, the operating system will automatically label any data coming through that port with the SECRET A/B label. Similarly, a user on the low side must first access the guard via the low side Ethernet card before accessing the high side. Since the port associated with the low side Ethernet card is labeled SECRET A, all data coming through that port will be labeled SECRET A by the operating system. If a SECRET A/B process running on behalf of a user from the high side attempted to write to the low side, the operating system would disallow the write because the low side port is only SECRET A. If a SECRET A process running on behalf of a user from the low side attempted to read from the high side, the operating system would disallow this since the high side port is SECRET A/B.

There is one instance where an information flow from high to low is allowed by the guard. This capability is granted to the guard process through a privilege mechanism. The guard process is trusted to append one byte of information to an existing file in the low directory. The byte contains a value of either 0 or 1. The value of the byte is determined from information supplied from high, and this value is passed to the low system. The guard process is trusted to append strictly one byte containing no other value except 0 or 1. The guard process cannot create a new file in the low directory.

As stated previously, the guard must be formally accredited before it is used operationally. It might also be useful to make further enhancements to the guard before employing it. For instance, the auditing and report generation features of the guard could be specialized for each specific environment. Trusted Xenix is responsible for auditing system events and creating an audit trail of these events. However, the Trusted Xenix auditing capabilities are general to the overall system. Auditing capabilities could be developed that are more specific to the guard operations. This could reduce the size of the audit trail and make the audit trail easier to understand.

Other features that could be added are host authentication and error reporting. Host authentication would be used to verify both the low host and the high host as valid members of the networks. Error reporting would give a better indication to the low host as to the condition of the message when it arrived at the high host.

DIDS (Distributed Intrusion Detection System) – Motivation, Architecture, and An Early Prototype

Steven R. Snapp¹, James Brentano², Gihan V. Dias, Terrance L. Goan,
L. Todd Heberlein, Che-Lin Ho, Karl N. Leviitt, Biswanath Mukherjee, Stephen E. Smaha¹,
Tim Grance³, Daniel M. Teaf³, and Doug Mansur⁴

Computer Security Laboratory
Division of Computer Science
University of California, Davis
Davis, California 95616

ABSTRACT

Intrusion detection is the problem of identifying unauthorized use, misuse, and abuse of computer systems by both system insiders and external penetrators. The proliferation of heterogeneous computer networks provides additional implications for the intrusion detection problem. Namely, the increased connectivity of computer systems gives greater access to outsiders, and makes it easier for intruders to avoid detection. IDS's are based on the belief that an intruder's behavior will be noticeably different from that of a legitimate user. We are designing and implementing a prototype Distributed Intrusion Detection System (DIDS) that combines distributed monitoring and data reduction (through individual host and LAN monitors) with centralized data analysis (through the DIDS director) to monitor a heterogeneous network of computers. This approach is unique among current IDS's. A main problem considered in this paper is the Network-user Identification problem, which is concerned with tracking a user moving across the network, possibly with a new user-id on each computer. Initial system prototypes have provided quite favorable results on this problem and the detection of attacks on a network. This paper provides an overview of the motivation behind DIDS, the system architecture and capabilities, and a discussion of the early prototype.

1. Introduction

Intrusion detection is defined to be the problem of identifying individuals who are using a computer system without authorization (i.e., *crackers*) and those who have legitimate access to the system but are exceeding their privileges (i.e., the *insider threat*). Work is being done elsewhere on Intrusion Detection Systems (IDS's) for a single host [8, 10, 11] and for several hosts connected by a network [6, 7, 12]. Our own earlier work on the Network Security Monitor (NSM) concentrated on monitoring a broadcast Local Area Network (LAN) [3].

The proliferation of heterogeneous computer networks has serious implications for the intrusion detection problem. Foremost among these implications is the increased opportunity for unauthorized access that is provided by the network's connectivity. This problem is exacerbated when dial-up or internetwork access is allowed, as well as when unmonitored hosts (viz. hosts without audit trails) are present. The use of distributed rather than centralized computing resources also implies reduced control over those resources. Moreover, multiple independent computers are likely to generate more audit data than a single computer, and this audit data is dispersed among the various systems. Clearly, not all of the audit data can be forwarded to a single IDS for analysis; some analysis must be accomplished locally.

¹ Haystack Laboratories, Inc., 8920 Business Park Dr, Suite 270, Austin, TX 78759

² Pacific Gas and Electric Company, 77 Beale St, Room 1871B, San Francisco, CA 94106

³ United States Air Force Cryptologic Support Center, San Antonio, TX 78243

⁴ Lawrence Livermore National Labs, Livermore, CA 94550

This paper describes a prototype Distributed Intrusion Detection System (DIDS) which generalizes the target environment in order to monitor multiple hosts connected via a network as well as the network itself. The DIDS components include the DIDS director, a single host monitor per host, and a single LAN monitor for each LAN segment of the monitored network. The information gathered by these distributed components is transported to, and analyzed at, a central location (viz. an expert system, which is a sub-component of the director), thus providing the capability to aggregate information from different sources. We can cope with any audit trail format as long as the events of interest are provided.

DIDS is designed to operate in a heterogeneous environment composed of C2 [1] or higher rated computers. The current target environment consists of several hosts connected by a broadcast LAN segment (presently an Ethernet, see Fig. 1). The use of C2-rated systems implies a consistency in the content of the system audit trails. This allows us to develop standard representations into which we can map audit data from UNIX, VMS, or any other system with C2 auditing capabilities. The C2 rating also guarantees, as part of the Trusted Computing Base (TCB), the security and integrity of the host's audit records. Although the hosts must comply with the C2 specifications in order to be monitored directly, the network related activity of non-compliant hosts can be monitored via the LAN monitor. Since all attacks that utilize the network for system access will pass through the LAN segment, the LAN monitor will be able to monitor all of this traffic.

Section 2 motivates our work by describing the type of behavior which DIDS is intended to detect. In Section 3 we present an overview of the DIDS architecture. In Section 4 we formulate the concept of the network-user identification (NID), an identifier for a network-wide user, and describe its use in distributed intrusion detection. Sections 5 and 6 deal with the host and LAN monitors, respectively, while Section 7 discusses the expert system and its processing mechanisms based on the NID. Section 8 provides some concluding remarks.

2. Scenarios

The detection of certain attacks against a networked system of computers requires information from multiple sources. A simple example of such an attack is the so-called *doorknob* attack. In a doorknob attack the intruder's goal is to discover, and gain access to, insufficiently-protected hosts on a system. The intruder generally tries a few common account and password combinations on each of a number of computers. These simple attacks can be remarkably successful [4]. As a case in point, UC Davis' NSM recently observed an attacker of this type gaining super-user access to an external computer which did not require a password for the super-user account. In this case, the intruder used *telnet* to make the connection from a university computer system, and then repeatedly tried to gain access to several different computers at the external site. In cases like these, the intruder tries only a few logins on each machine (usually with different account names), which means that an IDS on each host may not flag the attack. Even if the behavior is recognized as an attack on the individual host, current IDS's are generally unable to correlate reports from multiple hosts; thus they cannot recognize the *doorknob* attack as such. Because DIDS aggregates and correlates data from multiple hosts and the network, it is in a position to recognize the doorknob attack by detecting the pattern of repeated failed logins even though there may be too few on a single host to alert that host's monitor.

In another incident, our NSM recently observed an intruder gaining access to a computer using a guest account which did not require a password. Once the attacker had access to the system, he exhibited behavior which would have alerted most existing IDS's (e.g., changing passwords and failed events). In an incident such as this, DIDS would not only report the attack, but may also be able to identify the source of the attack. That is, while most IDS's would report the occurrence of an incident involving user "guest" on the target machine, DIDS would also report that user "guest" was really, for example, user "smith" on the source machine, assuming that the source machine was in the monitored domain. It may also be possible to go even further back and identify all of the different user accounts in the "chain" to find the initial launching point of the attack.

Another possible scenario is what we call *network browsing*. This occurs when a (network) user is looking through a number of files on several different computers within a short period of time. The browsing activity level on any single host may not be sufficiently high enough to raise any alarm by itself. However, the network-wide, aggregated browsing activity level may be high enough to raise suspicion on this user. Network browsing can be detected as follows. Each host monitor will report that a particular user is browsing on that system, even if the corresponding degree of browsing is small. The expert system can then aggregate such information from multiple hosts to determine that all of the browsing activity corresponds to the same network

user. This scenario presents a key challenge for DIDS: the tradeoff between sending all audit records to the director versus missing attacks because thresholds on each host are not exceeded.

In addition to the specific scenarios outlined above, there are a number of general ways that an intruder can use the connectivity of the network to hide his trail and to enhance his effectiveness. Some of the attack configurations which have been hypothesized include *chain* and *parallel* attacks [2]. DIDS combats these inherent vulnerabilities of the network by using the very same connectivity to help track and detect the intruder. Note that DIDS should be at least as effective as host-based IDS's (if we implement all of their functionality in the DIDS host monitor), and at least as effective as the stand-alone NSM.

3. DIDS Architecture

The DIDS architecture combines distributed monitoring and data reduction with centralized data analysis. This approach is unique among current IDS's. The components of DIDS are the *DIDS director*, a single *host monitor* per host, and a single *LAN monitor* for each broadcast LAN segment in the monitored network. DIDS can potentially handle hosts without monitors since the LAN monitor can report on the network activities of such hosts. The host and LAN monitors are primarily responsible for the collection of evidence of unauthorized or suspicious activity, while the DIDS director is primarily responsible for its evaluation. Reports are sent independently and asynchronously from the host and LAN monitors to the DIDS director through a communications infrastructure (Fig. 2). High level communication protocols between the components are based on the ISO Common Management Information Protocol (CMIP) recommendations, allowing for future inclusion of CMIP management tools as they become useful. The architecture also provides for bidirectional communication between the DIDS director and any monitor in the configuration. This communication consists primarily of notable events and anomaly reports from the monitors. The director can also make requests for more detailed information from the distributed monitors via a "GET" directive, and issue commands to have the distributed monitors modify their monitoring capabilities via a "SET" directive. A large amount of low level filtering and some analysis is performed by the host monitor to minimize the use of network bandwidth in passing evidence to the director.

The host monitor consists of a *host event generator* (HEG) and a *host agent*. The HEG collects and analyzes audit records from the host's operating system. The audit records are scanned for *notable events*, which are transactions that are of interest independent of any other records. These include, among others, failed events, user authentications, changes to the security state of the system, and any network access such as *rlogin* and *rsh*. These notable events are then sent to the director for further analysis. In enhancements under development, the HEG will also track user sessions and report anomalous behavior aggregated over time through user/group profiles and the integration of Haystack [10] into DIDS. The host agent handles all communications between the host monitor and the DIDS director.

Like the host monitor, the LAN monitor consists of a *LAN event generator* (LEG) and a *LAN agent*. The LEG is currently a subset of UC Davis' NSM [3]. Its main responsibility is to observe all of the traffic on its segment of the LAN to monitor host-to-host connections, services used, and volume of traffic. The LAN monitor reports on such network activity as *rlogin* and *telnet* connections, the use of security-related services, and changes in network traffic patterns.

The DIDS director consists of three major components that are all located on the same dedicated workstation. Because the components are logically independent processes, they could be distributed as well. The *communications manager* is responsible for the transfer of data between the director and each of the host and the LAN monitors. It accepts the notable event records from each of the host and LAN monitors and sends them to the *expert system*. On behalf of the expert system or user interface, it is also able to send requests to the host and LAN monitors for more information regarding a particular subject. The expert system is responsible for evaluating and reporting on the security state of the monitored system. It receives the reports from the host and the LAN monitors, and, based on these reports, it makes inferences about the security of each individual host, as well as the system as a whole. The expert system is a rule-based system with simple learning capabilities. The director's *user interface* allows the System Security Officer (SSO) interactive access to the entire system. The SSO is able to watch activities on each host, watch network traffic (by setting "wire-taps"), and request more specific types of information from the monitors.

We anticipate that a growing set of tools, including incident-handling tools and network-management tools, will be used in conjunction with the intrusion-detection functions of DIDS. This will give the SSO the

ability to actively respond to attacks against the system in real-time. Incident-handling tools may consist of possible courses of action to take against an attacker, such as cutting off network access, a directed investigation of a particular user, removal of system access, etc. Network-management tools that are able to perform network mapping would also be useful.

4. The Network-user Identification (NID)

One of the more interesting challenges for intrusion detection in a networked environment is to track users and objects (e.g., files) as they move across the network. For example, an intruder may use several different accounts on different machines during the course of an attack. Correlating data from several independent sources, including the network itself, can aid in recognizing this type of behavior and tracking an intruder to their source. In a networked environment, an intruder may often choose to employ the interconnectivity of the computers to hide his true identity and location. It may be that a single intruder uses multiple accounts to launch an attack, and that the behavior can be recognized as suspicious only if one knows that all of the activity emanates from a single source. For example, it is not particularly noteworthy if a user inquires about who is using a particular computer (e.g., using the UNIX *who* or *finger* command). However, it may be indicative of an attack if a user inquires about who is using each of the computers on a LAN and then subsequently logs into one of the hosts. Detecting this type of behavior requires attributing multiple sessions, perhaps with different account names, to a single source.

This problem is unique to the network environment and has not been dealt with before in this context. Our solution to the multiple user identity problem is to create a *network-user identification* (NID) the first time a user enters the monitored environment, and then to apply that NID to any further instances of the user. All evidence about the behavior of any instance of the user is then accountable to the single NID. In particular, we must be able to determine that "smith@host1" is the same user as "jones@host2", if in fact they are. Since the network-user identification problem involves the collection and evaluation of data from both the host and LAN monitors, examining it is a useful method to understand the operation of DIDS. In the following subsections we examine each of the components of DIDS in the context of the creation and use of the NID.

5. The Host Monitor

The host monitor is currently installed on Sun SPARCstations running SunOS 4.0.x with the Sun C2 security package [9]. Through the C2 security package, the operating system produces audit records for virtually every transaction on the system. These transactions include file accesses, system calls, process executions, and logins. The contents of the Sun C2 audit record are: record type, record event, time, real user ID, audit user ID, effective user ID, real group ID, process ID, error code, return value, and label.

The host monitor (Fig. 3) examines each audit record to determine if it should be forwarded to the expert system for further evaluation. Certain critical audit records are always passed directly to the expert system (i.e., *notable events*); others are processed locally by the host monitor (i.e., *profiles* and *attack signatures*, which are sequences of noteworthy events which indicate the symptoms of attacks) and only summary reports are sent to the expert system. Thus, one of the design objectives is to push as much of the processing operations down to the low-level monitors as possible. In order to do this, the HEG creates a more abstract object called an *event*. The event includes any significant data provided by the original audit record plus two new fields: the *action* and the *domain*. The action and domain are abstractions which are used to minimize operating system dependencies at higher levels. Actions characterize the dynamic aspect of the audit records. Domains characterize the objects of the audit records. In most cases, the objects are files or devices and their domain is determined by the characteristics of the object or its location in the file system. Since processes can also be objects of an audit record, they are also assigned to domains, in this case by their function.

The actions are: *session_start*, *session_end*, *read* (a file or device), *write* (a file or device), *execute* (a process), *terminate* (a process), *create* (a file or (virtual) device), *delete* (a file or (virtual) device), *move* (rename a file or device), *change_rights*, and *change_user_id*. The domains are: *tagged*, *authentication*, *audit*, *network*, *system*, *sys_info*, *user_info*, *utility*, *owned*, and *not_owned*.

The domains are prioritized so that an object is assigned to the first applicable domain. *Tagged* objects are ones which are thought a priori to be particularly interesting in terms of detecting intrusions. Any file, device, or process can be tagged (e.g., */etc/passwd*). *Authentication* objects are the processes and files which are used to provide access control on the system (e.g., the password file). Similarly, *audit* objects relate to the

accounting and security auditing processes and files. *Network* objects are the processes and files not covered in the previous domains which relate to the use of the network. *System* objects are primarily those which are concerned with the execution of the operating system itself, again exclusive of those objects already assigned to previously considered domains. *Sys_info* and *user_info* objects provide information about the system and about the users of the system, respectively. The *utility* objects are the bulk of the programs run by the users (e.g., compilers and editors). In general, the execution of an object in the utility domain is not interesting (except when the use is excessive), but the creation or modification of one is. *Owned* objects are relative to the user. *Not_owned* objects are, by exclusion, every object not assigned to a previous domain. They are also relative to a user; thus, files in the owned domain relative to "smith" are in the not_owned domain relative to "jones".

All possible transactions fall into one of a finite number of events formed by the cross product of the actions and the domains, and each event may also succeed or fail. Note that no distinction is made between files, directories or devices, and that all of these are treated simply as objects. Not every action is applicable to every object; for example, the *terminate* action is applicable only to processes. The choice of these domains and actions is somewhat arbitrary in that one could easily suggest both finer and coarser grained partitions. However, they capture most of the interesting behavior for intrusion detection and correspond reasonably well with what other researchers in this field have found to be of interest [5,10]. By mapping an infinite number of transactions to a finite number of events, we not only remove operating system dependencies, but also restrict the number of permutations that the expert system will have to deal with. The concept of the domain is one of the keys to detecting abuses. Using the domain allows us to make assertions about the nature of a user's behavior in a straightforward and systematic way. Although we lose some details provided by the raw audit information, that is more than made up for by the increase in portability, speed, simplicity, and generality.

An event reported by a host monitor is called a host audit record (har). The record syntax is: har(Monitor-ID, Host-ID, Audit-UID, Real-UID, Effective-UID, Time, Domain, Action, Transaction, Object, Parent Process, PID, Return Value, Error Code).

Of all the possible events, only a subset are forwarded to the expert system. For the creation and application of the NID, it is the events which relate to the creation of user sessions or to a change in an account that are important. These include all the events with *session_start* actions, as well as ones with an *execute* action applied to the *network* domain. These latter events capture such transactions as executing the *rlogin*, *telnet*, *rsh*, and *rexec* UNIX programs. The HEG consults external tables, which are built by hand, to determine which events should be forwarded to the expert system. Because they relate to events rather than to the audit records themselves, the tables and the modules of the HEG which use them are portable across operating systems. The only portion of the HEG which is operating system dependent is the module which creates the events.

6. The LAN Monitor

The LAN monitor is currently a subset of UC Davis' Network Security Monitor [3]. The LAN monitor builds its own "LAN audit trail". The LAN monitor observes each and every packet on its segment of the LAN and, from these packets, it is able to construct higher-level objects such as connections (logical circuits), and service requests using the TCP/IP or UDP/IP protocols. In particular, it audits host-to-host connections, services used, and volume of traffic per connection.

Similar to the host monitor, the LAN monitor uses several simple analysis techniques to identify significant events. The events include the use of certain services (e.g., *rlogin* and *telnet*) as well as activity by certain classes of hosts (e.g., a PC without a host monitor). The LAN monitor also uses and maintains profiles of expected network behavior. The profiles consist of expected data paths (e.g., which systems are expected to establish communication paths to which other systems, and by which service) and service profiles (e.g., what a typical *telnet*, *mail*, or *finger* is expected to look like).

The LAN monitor also uses heuristics in an attempt to identify the likelihood that a particular connection represents intrusive behavior. These heuristics consider the capabilities of each of the network services, the level of authentication required for each of the services, the security level for each machine on the network, and signatures of past attacks. The abnormality of a connection is based on the probability of that particular connection occurring and the behavior of the connection itself. Upon request, the LAN monitor is also able to provide a more detailed examination of any connection, including capturing every character crossing the network (i.e., a wire-tap). This capability can be used to support a directed investigation of a particular subject or object. Like the host monitor, the LAN monitor forwards relevant security information to the director through its LAN agent.

An event reported by a LAN monitor is called a network audit record (nar). The record syntax is: nar(Monitor-ID, Source_Host, Dest_Host, Time, Service, Domain, Status).

The LAN monitor has several responsibilities with respect to the creation and use of the NID. The LAN monitor is responsible for detecting any connections related to *rlogin* and *telnet* sessions. Once these connections are detected, the LAN monitor can be used to verify the owner of a connection. The LAN monitor can also be used to help track tagged objects moving across the network. The SSO can also ask for a wire-tap on a certain network connection to monitor a particular user's behavior.

7. The Expert System

DIDS utilizes a rule-based (or production) expert system. The expert system is currently written in Prolog, and much of the form of the rule base comes from Prolog and the logic notation that Prolog implies. The expert system uses rules derived from the hierarchical Intrusion Detection Model (IDM). The IDM describes the data abstractions used in inferring an attack on a network of computers. That is, it describes the transformation from the distributed raw audit data to high level hypotheses about intrusions and about the overall security of the monitored environment. In abstracting and correlating data from the distributed sources, the model builds a virtual machine which consists of all the connected hosts as well as the network itself. This unified view of the distributed system simplifies the recognition of intrusive behavior which spans individual hosts. The model is also applicable to the trivial network of a single computer.

The model is the basis of the rule base. It serves both as a description of the function of the rule base, and as a touchstone for the actual development of the rules. The IDM consists of 6 layers, each layer representing the result of a transformation performed on the data (see Table 1).

The objects at the first level of the model are the audit records provided by the host operating system, by the LAN monitor, or by a third party auditing package. The objects at this level are both syntactically and semantically dependent on the source. At this level, all of the activity on the host or LAN is represented.

At the second level, the *event* (which has already been discussed in the context of the host and LAN monitor) is both syntactically and semantically independent of the source standard format for events.

The third layer of the IDM creates a *subject*. This introduces a single identification for a user across many hosts on the network. It is the subject who is identified by the NID (see section 7.1). Upper layers of the model treat the network-user as a single entity, essentially ignoring the local identification on each host. Similarly, above this level, the collection of hosts on the LAN are generally treated as a single distributed system with little attention being paid to the individual hosts.

The fourth layer of the model introduces the event in *context*. There are two kinds of context: temporal and spatial. As an example of temporal context, behavior which is unremarkable during standard working hours may be highly suspicious during off hours [5]. The IDM, therefore, allows for the application of information about wall-clock time to the events it is considering. Wall-clock time refers to information about the time of day, weekdays versus weekends and holidays, as well as periods when an increase in activity is expected. In addition to the consideration of external temporal context, the expert system uses time windows to correlate events occurring in temporal proximity. This notion of temporal proximity implements the heuristic that a call to the UNIX *who* command followed closely by a *login* or *logout* is more likely to be related to an intrusion than either of those events occurring alone. Spatial context implies the relative importance of the source of events. That is, events related to a particular user, or events from a particular host, may be more likely to represent an intrusion than similar events from a different source. For instance, a user moving from a low-security machine to a high-security machine may be of greater concern than a user moving in the opposite direction. The model also allows for the correlation of multiple events from the same user or source. In both of these cases, multiple events are more noteworthy when they have a common element than when they do not.

The fifth layer of the model considers the *threats* to the network and the hosts connected to it. Events in context are combined to create threats. The threats are partitioned by the nature of the abuse and the nature of the target. In other words, what is the intruder doing, and what is he doing it to? Abuses are divided into *attacks*, *misuses*, and *suspicious acts*. Attacks represent abuses in which the state of the machine is changed. That is, the file system or process state is different after the attack than it was prior to the attack. Misuses represent out-of-policy behavior in which the state of the machine is not affected. Suspicious acts are events which, while not a violation of policy, are of interest to an IDS. For example, commands which provide

information about the state of the system may be suspicious. The targets of abuse are characterized as being either *system* objects or *user* objects and as being either *passive* or *active*. User objects are owned by non-privileged users and/or reside within a non-privileged user's directory hierarchy. System objects are the complement of user objects. Passive objects are files, including executable binaries, while active objects are essentially running processes.

At the highest level, the model produces a numeric value between one and 100 which represents the overall *security state* of the network. The higher the number the less secure the network. This value is a function of all the threats for all the subjects on the system. Here again we treat the collection of hosts as a single distributed system. Although representing the security level of the system as a single value seems to imply some loss of information, it provides a quick reference point for the SSO. In fact, in the current implementation, no information is lost since the expert system maintains all the evidence used in calculating the security state in its internal database, and the SSO has access to that database.

In the context of the network-user identification problem we are concerned primarily with the lowest three levels of the model: the audit data, the event, and the subject. The generation of the first two of these have already been discussed; thus, the creation of the subject is the focus of the following subsection.

The expert system is responsible for applying the rules to the evidence provided by the monitors. In general, the rules do not change during the execution of the expert system. What does change is a numerical value associated with each rule. This *Rule Value* (RV) represents our confidence that the rule is useful in detecting intrusions. These rule values are manipulated using a negative reinforcement training method which allows the expert system to continually lower the number of false attack reports. When a potential attack is reported by the expert system, the SSO determines the validity of the report and gives feedback to the expert system. If the report was deemed faulty, then the expert system lowers the RV's associated with the rules that were used to draw that conclusion. In addition to this directed training, which may lower some rule values, the system also automatically increases the RV's of all the rules on a regular basis. This recovery algorithm allows the system to adapt to changes in the environment as well as recover from faulty training.

Logically the rules have the form:

antecedent => consequence

where the antecedent is either a fact reported by one of the distributed monitors, or a consequence of some previously satisfied rule. The antecedent may also be a conjunction of these. The overall structure of the rule base is a tree rooted at the top. Thus, many facts at the bottom of the tree will lead to a few conclusions at the top of the tree.

The expert system shell consists of approximately a hundred lines of Prolog source code. The shell is responsible for reading new facts reported by the distributed monitors, attempting to apply the rules to the facts and hypotheses in the Prolog database, reporting suspected intrusions, and maintaining the various dynamic values associated with the rules and hypotheses. The syntax for rules is:

rule(*n*, *r*, (single, [*A*]), (*C*)).

where *n* is the rule number, *r* is the initial RV, *A* is the single antecedent, and *C* is the consequence. Conjunctive rules have the form:

rule(*n*, *r*, (and, [*A*₁ *A*₂ *A*₃]), (*C*)).

where *A*₁, *A*₂, *A*₃ are the antecedents and *C* is the consequence. Disjunctive rules are not allowed; that situation is dealt with by having multiple rules with the same consequence.

7.1. Building the NID

With respect to Unix, the only legitimate ways to create an instance of a user are for the user to login from a terminal, console, or off-LAN source, to change the user-id in an existing instance, or to create additional instances (local or remote) from an existing instance. In each case, there is only one initial login (system wide) from an external device. When this original login is detected, a new unique NID is created. This NID is applied to every subsequent action generated by that user. When a user with a NID creates a new login session, that new session is associated with his original NID. Thus the system maintains a single identification for each physical user.

We consider an instance of a user to be the 4-tuple $\langle \text{session_start}, \text{user-id}, \text{host-id}, \text{time} \rangle$. Thus each login creates a new instance of a user. In associating a NID with an instance of a user, the expert system first tries to use an existing NID. If no NID can be found which applies to the instance, a new one is created. Trying to find an applicable existing NID consists of several steps. If a user changes identity (e.g., using UNIX's *su* command) on a host, the new instance is assigned the same NID as the previous identity. If a user performs a remote login from one host to another host, the new instance gets the same NID as the source instance. When no applicable NID is found, a new unique NID is created by the following rule:

```
rule(111,1000,[
  hhar(_Host1,AUID,_,Time1,_session_start,_,_'local',_,_), /* login */
  \+ (ih(net_user(NID,AUID,Host,_),_,_,_)), /* no NID yet */
  newNID(X) /* create new NID */
],
  (net_user(X,AUID,Host1,Time1))). /* new net user */
```

The actual association of a NID with a user instance is through the hypothesis *net_user*. A new hypothesis is created for every event reported by the distributed monitors. This new hypothesis, called a *subject*, is formed by the rule:

```
rule(110,100,(and,[
  har(Mon,Host,AUID,UID,EUID,Time,Dom,Act,Trans,Obj,Parent,PID,Ret,Err).
  net_user(NID,AUID,Host,_)
]),
  subj(NID,Mon,Host,AUID,UID,EUID,Time,Dom,Act,Trans,Obj,Parent,PID,Ret,Err))).
```

The rule creates a subject, getting the NID from the *net_user* and the remaining fields from the host audit record, if and only if both the user-id and the host-id match. It is through the use of the subject that the expert system correlates a user's actions regardless of the login name or host-id.

There is still some uncertainty involved with the network-user identification problem. If a user leaves the monitored domain and then comes back in with a different user-id, it is not possible to connect the two instances. Similarly, if a user passes through an unmonitored host, there is still uncertainty that any connection leaving the host is attributable to any connection entering the host. Multiple connections originating from the same host at approximately the same time also allow uncertainty if the user names do not provide any helpful information. The expert system can make a final decision with additional information from the host and LAN monitors that can (with high probability) disambiguate the connections.

8. Conclusion

Our Distributed Intrusion Detection System (DIDS) is being developed to address the shortcomings of current single host IDS's by generalizing the target environment to multiple hosts connected via a network (LAN). Most current IDS's do not consider the impact of the LAN structure when attempting to monitor user behavior for attacks against the system. Intrusion detection systems designed for a network environment will become increasingly important as the number and size of LAN's increase. Our prototype has demonstrated the viability of our distributed architecture in solving the network-user identification problem. We have tested the system on a sub-network of Sun SPARCstations and it has correctly identified network users in a variety of scenarios. Work continues on the design, development, and refinement of rules, particularly those which can take advantage of knowledge about particular kinds of attacks. The initial prototype expert system has been written in Prolog, but it is currently being ported to CLIPS due to the latter's superior performance characteristics and easy integration with the C programming language. We are designing a signature analysis component for the host monitor to detect events and sequences of events that are known to be indicative of an attack, based on a specific context. In addition to the current host monitor, which is designed to detect attacks on general purpose multi-user computers, we intend to develop monitors for application specific hosts such as file servers and gateways. In support of the ongoing development of DIDS we are planning to extend our model to a hierarchical Wide Area Network environment.

Acknowledgments

The DIDS project is sponsored by the United States Air Force Cryptologic Support Center through a contract with the Lawrence Livermore National Labs.

References

1. Department of Defense, *Trusted Computer System Evaluation Criteria*, National Computer Security Center, DOD 5200.28-STD, Dec. 1985.
2. G.V. Dias, K.N. Levitt, and B. Mukherjee, "Modeling Attacks on Computer Systems: Evaluating Vulnerabilities and Forming a Basis for Attack Detection," Technical Report CSE-90-41, University of California, Davis, Jul. 1990.
3. L.T. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A Network Security Monitor," *Proc. 1990 Symposium on Research in Security and Privacy*, pp. 296-304, Oakland, CA, May 1990.
4. B. Landreth, *Out of the Inner Circle, A Hacker's Guide to Computer Security*, Microsoft Press, Bellevue, WA, 1985.
5. T. Lunt, "Automated Audit Trail Analysis and Intrusion Detection: A Survey," *Proc. 11th National Computer Security Conference*, pp. 65-73, Baltimore, MD, Oct. 1988.
6. T.F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P.G. Neumann, and C. Jalali, "IDES: A Progress Report," *Proc. Sixth Annual Computer Security Applications Conference*, Tucson, AZ, Dec. 1990.
7. T.F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. Jalali, H.S. Javitz, A. Valdes, and P.G. Neumann, "A Real-Time Intrusion-Detection Expert System (IDES)," Interim Progress Report, Project 6784, SRI International, May 1990.
8. M.M. Sebring, E. Shellhouse, M.E. Hanna, and R.A. Whitehurst, "Expert Systems in Intrusion Detection: A Case Study," *Proc. 11th National Computer Security Conference*, pp. 74-81, Oct. 1988.
9. W.O. Sibert, "Auditing in a Distributed System: SunOS MLS Audit Trails," *Proc. 11th National Computer Security Conference*, Baltimore, MD, Oct. 1988.
10. S.E. Smaha, "Haystack: An Intrusion Detection System," *Proc. IEEE Fourth Aerospace Computer Security Applications Conference*, Orlando, FL, Dec. 1988.
11. H.S. Vaccaro and G.E. Liepins, "Detection of Anomalous Computer Session Activity," *Proc. 1989 Symposium on Research in Security and Privacy*, pp. 280-289, Oakland, CA, May 1989.
12. J.R. Winkler, "A Unix Prototype for Intrusion and Anomaly Detection in Secure Networks," *Proc. 13th National Computer Security Conference*, pp. 115-124, Washington, D.C., Oct. 1990.

Level	Name	Explanation
6	Security State	overall network security level
5	Threat	definition of categories of abuse
4	Context	event placed in context
3	Subject	definition and disambiguation of network user
2	Event	OS independent representation of user action (finite number of these)
1	Data	audit or OS provided data

Table 1. Intrusion Detection Model

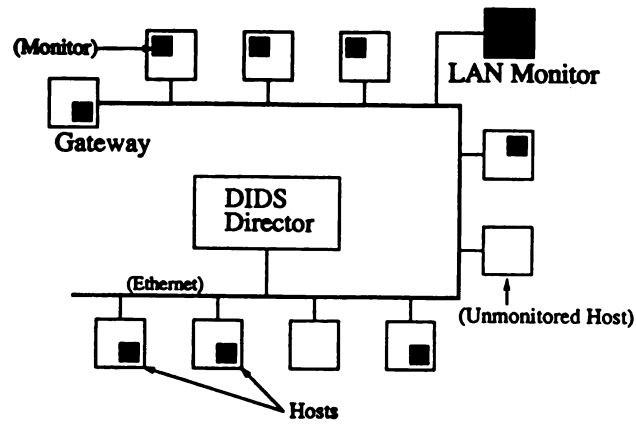


Fig. 1. DIDS Target Environment

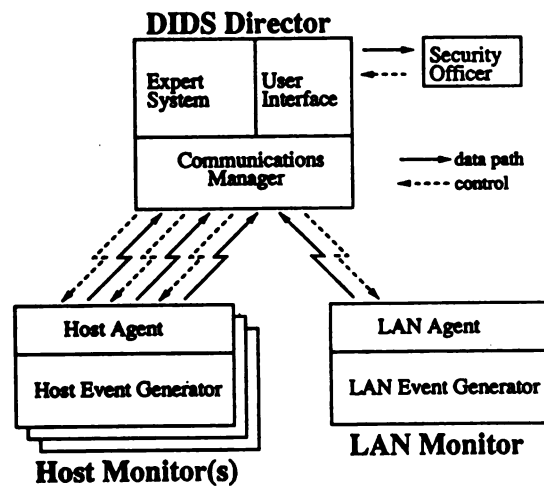


Fig. 2. Communications Architecture

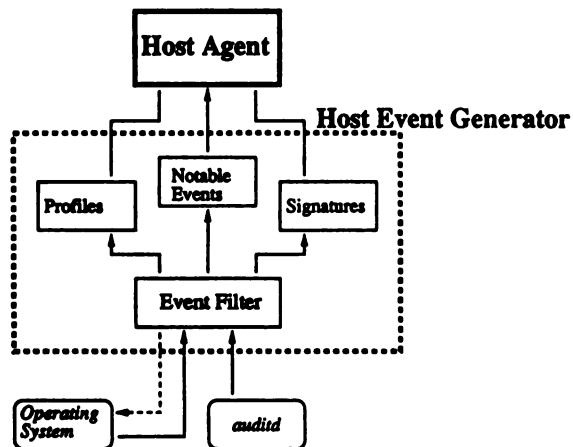


Fig. 3. Host Monitor Structure

A DISTRIBUTED IMPLEMENTATION OF THE TRANSFORM MODEL

Ravi S. Sandhu and Gurpreet S. Suri

Center for Secure Information Systems

and

Department of Information and Software Systems Engineering

George Mason University, Fairfax, VA 22030-4444

ABSTRACT The Transform access-control model is based on the concept of transformation of access rights. It has previously been shown that Transform unifies a number of diverse access control mechanisms such as amplification, copy flags, separation of duties and synergistic authorisation. It has also been shown that Transform has an efficient algorithm for safety analysis of the propagation of access rights (i.e., the determination of whether or not a given subject can ever acquire access to a given object). In this paper we propose a distributed implementation of Transform. Our design is based on capabilities with identities of subjects buried in them. This ensures unforgeability of capabilities as well as enables enforcement of “mandatory” controls on propagation of capabilities from one subject to another. The design provides for immediate, selective, partial and complete revocation on a temporary as well as permanent basis.

Keywords: Distributed Systems, Secure Architectures, Capabilities

1 INTRODUCTION

The need for access controls arises in any computer system that provides for controlled sharing of information and other resources among multiple users. Access control models (or protection models) provide a framework for specifying, analysing and implementing security policies in multi-user systems. These models are typically defined in terms of the well-known abstractions of subjects, objects and access rights with which we assume the reader is familiar. A wide variety of access-control models have been described in the literature [3,4,10,12,16, for instance]. Unfortunately very few have been implemented or have even influenced implementations of actual systems.*

In this paper we take a step towards closing this gap between theory and practise. Our principal contribution is the outline of a distributed implementation of the recently proposed Transform model [17]. Transform derives its name from its central concept of transformation of access rights. The idea is that access rights get transformed as they are propagated from one subject to another, e.g., a security-officer who has the review right for a document may propagate the release right for the document to the document's author. It has previously been shown [17] that Transform elegantly unifies a number of seemingly different access control mechanisms such as amplification [5], copy flags [12], separation of duties [4] and synergistic authorisation [14]. It has also been shown [17] that there are efficient algorithms for the safety problem in Transform (i.e., the determination of whether or not a given subject can ever acquire access to a given object).

Thus Transform incorporates practically useful expressive power while allowing for safety analysis. Transform is actually a special case of the Schematic Protection Model (SPM) [16]. Like Transform, SPM also exhibits strong safety properties. This is in contrast to the weak safety properties of the access-matrix model commonly known as HRU [10]. Both HRU and SPM have undecidable safety in general [10,18]. In HRU safety becomes undecidable under very weak assumptions, notably

*The notable exception is the Bell-LaPadula model [3] whose strong influence on military systems has been formally incorporated in evaluation criteria [8].

the bi-conditional monotonic case of [11]. On the other hand safety in SPM remains decidable under very strong assumptions, notably the acyclic attenuating case of [16]. In particular Transform falls outside the known decidable cases for HRU but well within the known decidable cases for SPM [17].

Our implementation proposal for Transform is strongly influenced by the identity-based capability architecture proposed by Gong [9]. The concept of embedding the identity of a subject in a capability in distributed systems has been known for some time [6]. It ensures that capabilities cannot be forged or propagated from one subject to another without intervention of trusted software. Gong's architecture is based on the familiar client-server model of services in a distributed system and includes mechanisms for revocation which were missing in earlier proposals such as [6]. We have extended Gong's proposal to accommodate Transform. In particular the concept of strongly typed subjects and objects, which is essential to Transform, has been incorporated.

The paper is organised as follows. Section 2 reviews the Transform model to the extent required for our objectives in this paper. Section 3 discusses distributed capability-based architectures in general and motivates our choice of building on Gong's approach. Section 4 describes our proposed implementation for Transform. The protocols involved in creation, propagation and revocation are presented. An example of the implementation is presented in section 5. The paper is concluded in section 6 with a discussion and proposals for future research.

2 THE TRANSFORM MODEL

The Transform model [17] was obtained by identifying the common foundation underlying a variety of different access-control mechanisms proposed in the literature. These include amplification [5], copy flags [12], separation of duties [4] and synergistic authorisation [14]. Considered in isolation these mechanisms are diverse and were largely proposed independently of each other. They all appear to be desirable and should be supported by any system which claims generality. However simply lumping them together results in a complex system with many unrelated mechanisms.

Transform introduces the unifying concept of transformation of rights which can occur in two different ways.

1. *Self transformation* or *internal transformation* allows a subject who possesses certain rights for an object to obtain additional rights for that object.
2. *Grant transformation* or *external transformation* occurs in the granting of access rights by one subject to another. The general idea is that possession of a right for an object by a subject allows that subject to give some other right for that object to another subject.

In addition Transform is based on the *strong typing* of subjects and objects, i.e., subjects and objects are classified into types when they are created and their type cannot change thereafter. Much of the power of transformation derives from predicating the ability to transform on the types of subjects and objects involved.

A security policy is stated in Transform by specifying the following (finite) components.

1. Disjoint sets of subject types TS object types TO and rights R.
2. A can-create function $cc : TS \rightarrow 2^{TO}$.
3. Create-rules $cr : TS \times TO \rightarrow 2^R$.
4. An internal transformation function $itrans : TS \times TO \times 2^R \rightarrow 2^R$.
5. A grant transformation function $grant : TS \times TS \times TO \times 2^R \rightarrow 2^R$.

The notation 2^X denotes the power set of X , i.e., the set of all subsets of X . These components of a Transform specification are explained in turn below.

The sets TS and TO define the subject types and object types respectively. For example subject types might be faculty, student, guest, etc., and object types might be file, mail-message, bulletin-board, etc. R defines the set of rights or privileges in the system, e.g., read, write, execute, etc.

There are two issues involved in object creation.[†] Firstly subjects need authorisation to create objects. Secondly the rights obtained as a result of creation must also be specified. Transform authorizes creation by means of the can-create function cc . The interpretation of

$$cc(u) = \{o_1, o_2, \dots, o_k\}$$

is that a subject of type u is authorized to create objects of type o_1 and objects of type o_2 , etc. The effect of creation is defined by create-rules. The interpretation of

$$cr(u, o) = \{r_1, r_2, \dots, r_p\}$$

is that when a subject U of type u creates an object O of type o the creator U obtains the rights r_1, r_2, \dots, r_p for O . For example if $cc(\text{user}) = \{\text{file}\}$ and $cr(\text{user}, \text{file}) = \{\text{own}\}$ the creator of a file gets the own right for it. For readability we will usually drop the set parenthesis around singleton sets, for instance by writing $cc(\text{user}) = \text{file}$ and $cr(\text{user}, \text{file}) = \text{own}$.

Authorisation for internal transformation is specified by the internal transformation function $itrans$. The interpretation of

$$itrans(u, o, \{x_1, \dots, x_n\}) = \{y_1, \dots, y_m\}$$

is that a subject of type u who has *all* the x_i rights specified on the left hand side for an object of type o can obtain the rights y_1, \dots, y_m for that object by internal transformation. For example, the policy that possession of the w (write) privilege for a file implies possession of the r (read) privilege is easily stated as follows.[‡]

$$itrans(\text{user}, \text{file}, w) = r$$

Another example of internal transformation occurs in situations described as synergistic authorisation in [14]. For instance consider a situation where a scientist (abbreviated as *sci*) needs approvals from a security officer and a patent officer before he can release a document (abbreviated as *doc*) for publication. Say these two approvals are respectively signified by possession of the a_s and a_p rights. We can express this policy as follows.

$$itrans(\text{sci}, \text{doc}, \{\text{own}, a_s, a_p\}) = \text{release}$$

That is, a scientist who owns a document and possesses the two approvals can acquire the release right for that document.

Grant transformations are authorized by the grant transformation function $grant$. The interpretation of

$$grant(u, v, o, \{x_1, \dots, x_n\}) = \{y_1, \dots, y_m\}$$

[†]There must be provision for creation of subjects in any realistic system. In practise creation of subjects is often strictly controlled by some distinguished system administrator or security officer. Such creation can be considered as occurring outside the normal scope of the system.

[‡]In multilevel systems this policy would amount to prohibiting write-up.

is that a subject of type u who has *all* the x_i rights specified on the left hand side for an object of type o can grant *one or more* of the rights y_1, \dots, y_m for that object to a subject of type v . A common example of grant transformation occurs with the copy flag c which controls whether the granted privilege can itself be further granted or not. For instance the following

$$\begin{aligned} \text{grant}(\text{user}, \text{user}, \text{file}, xc) &= \{xc, x\} \\ \text{grant}(\text{user}, \text{user}, \text{file}, x) &= \phi \end{aligned}$$

defines the (unlimited) copy flag. Here a user who has the xc privilege for a file can grant the xc privilege or the x privilege to another user, whereas a user with the x privilege for the file cannot grant x any further. Other variations of the copy flag, such as 1-step or n -step copy flags can be similarly defined [17].

The expressive power of Transform is illustrated by the following policy specification.

$$\begin{aligned} cc(\text{sci}) &= \text{doc} \\ cr(\text{sci}, \text{doc}) &= \{\text{own}, \text{read}\} \\ \text{grant}(\text{sci}, \text{security-officer}, \text{doc}, \text{own}) &= \text{review} \\ \text{grant}(\text{sci}, \text{patent-officer}, \text{doc}, \text{own}) &= \text{review} \\ \text{grant}(\text{security-officer}, \text{sci}, \text{doc}, \text{review}) &= a_s \\ \text{grant}(\text{patent-officer}, \text{sci}, \text{doc}, \text{review}) &= a_p \\ \text{itrans}(\text{sci}, \text{doc}, \{\text{own}, a_s, a_p\}) &= \text{release} \end{aligned}$$

The first two equations specify that (i) a scientist can create documents, and (ii) the scientist who creates a document obtains the *own* and *read* privileges for it.¹ The next two equations specify that a scientist who owns a document can ask for it to be reviewed by a security-officer and by a patent-officer. These officers can respectively return the a_s and a_p rights to the scientist signifying the respective approvals. The scientist can then release the document. This example is further elaborated in section 5.

This completes our description of the Transform model. Further motivation for Transform and additional examples of policies are given in [17].

3 DISTRIBUTED CAPABILITY SYSTEMS

Capability-based architectures have had a strong appeal ever since the concept was first proposed [7]. They are viewed as providing a sound and common basis for providing both reliability and security. In the context of conventional centralised systems a number of such machines have been built [13]. Some have even achieved moderate commercial success. Nevertheless today's popular CPUs are not capability based. In retrospect one can argue that using capabilities to solve the memory protection problem is an overkill. The marginal advantages of capabilities over memory segmentation and protection rings (which are available in the latest generation of microprocessors such as the Intel 80386) do not justify the extra costs and performance penalties. In other words the initial application of capabilities was at too low a level.

It is expected by many researchers [15, for instance] that in the 1990s distributed operating systems will dominate the computing environment. These systems will appear to users as a single centralised system with complete location transparency. To achieve this, reliability and security must be addressed as part of the basic design of these systems. Attempts to graft security features

¹Once a document has been created it can no longer be written. This is necessary in order to freeze the contents of the document. If revisions are required a new version of the document needs to be created.

later in the design cycle will surely fail, much as they are failing in conventional centralised systems. The capability-based framework continues to offer an attractive approach to these problems. In a distributed operating system capabilities are introduced at a much higher level than memory addressing. Capabilities need to be incorporated into the remote procedure call mechanism rather than the memory addressing mechanism. This offers the hope that the additional overhead will not severely degrade performance. Capabilities can moreover be integrated into the basic client-server structure of distributed systems to provide transparency.

There are three basic issues which must be confronted by the designer of a distributed capability-based system. These issues are complicated relative to conventional centralised capability-based systems because capabilities are dispersed in individual workstations and can no longer be assumed to be under tight control of a security kernel.

1. *Unforgeability*. It must be guaranteed that capabilities cannot be modified or manufactured by subjects. This requires some form of cryptographic sealing.
2. *Propagation*. It must be guaranteed that capabilities cannot be copied from one user to another. This requires some means of embedding the identity of a subject in a capability.
3. *Revocation*. It must be guaranteed that capabilities which have been granted can be withdrawn or revoked in a timely manner. This requires some means of invalidating existing capabilities and accounting for cascaded revocation.

Various solutions to one or more of these problems have been proposed in the literature. For instance Amoeba [15] uses "sparse capabilities" with cryptographic protection to ensure unforgeability. Unfortunately Amoeba does not address capability propagation or revocation. Davies [6] discusses mechanisms to embed the identity of a subject in a capability. This ensures that capabilities cannot be forged or propagated from one subject to another without intervention of trusted software. Davies, however, does not address the revocation issue. Gong's proposed architecture [9] is the first attempt to address all three issues in a distributed context. It is based on the familiar client-server model of services in distributed systems and therefore is a suitable foundation for us to build upon. However, Gong does not incorporate the notion of types which is basic to Transform. His architecture therefore needs to be extended for this purpose.

4 IMPLEMENTATION OF TRANSFORM

We now describe a distributed capability-based implementation of the Transform model. We assume that objects are encapsulated within object servers. The basic computation model is that of remote procedure calls involving the following sequence of events: (i) a client sends a request to a server to manipulate one or more objects, (ii) the server accepts and services the request, and (iii) the server sends back a reply. The object server runs on a trusted host which guarantees that the server cannot be bypassed. For ease of exposition we visualize each object server as running on a separate host. However, we allow multiple object servers on the same trusted host provided the security kernel on the host can enforce separation among these servers. If we have sufficient confidence in the security kernel we can also allow untrusted clients to coexist with object servers on a single trusted host.

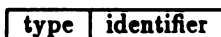
Each object server acts as the reference monitor (or access mediator) for the set of objects it manages. In other words the object server is part of the trusted computing base (TCB). The object server is responsible not only for access mediation but also for ensuring semantic correctness of the objects with respect to the abstract operations exported from the server. The object server itself has the ability to access all objects within its control. We emphasise that the object server is not a subject in the system but is rather a part of the TCB.

For simplicity, we require that each object server manage exactly one type of object. In practise this rule would probably be relaxed to allow a single server to manage multiple object types, particularly if they are closely related. On the other hand the same type of object may be managed by multiple object servers. For instance a given system may have numerous file servers. An individual file server manages some subset of the total collection of files in the system. We assume there is no replication of files, i.e., each file resides at exactly one file server.

Finally we assume there is an access decision facility (ADF) which can be consulted by object servers to determine the security policy. In the context of Transform the ADF will be consulted by object servers for finding out appropriate values of *cc*, *cr*, *grant* and *itrans*. Pieces of the ADF may actually reside at each object server while other pieces are remotely accessed. The reason for this is to allow quick local access to well-established and relatively static aspects of the policy while at the same time allowing for new types etc. to be introduced.

4.1 Identity and Type

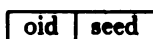
Each subject or object in the system has a globally unique identifier. Each subject or object also has a unique type which is determined when that subject or object is created. Thereafter the type cannot change. We assume the type of a subject or object is embedded in its identifier. Henceforth we refer to a subject identifier by *sid* and a object identifier by *oid*. These identifiers have the following structure.



The type field denotes the type of the object while the identifier field uniquely identifies each subject or object among instances of the same type. Note that *sid*'s and *oid*'s can be generated at will by users.

4.2 Capability Seeds

A capability seed is a secret random number associated with each *oid*. The seed is known only to the object server which manages the object identified by *oid*. We can visualise this association by the following pair.[†]



The purpose of the seed is to facilitate revocation and prevent against replay of revoked capabilities, as will be discussed later.

4.3 Capabilities

A capability has the following structure.



where the seal is computed using a publicly known one-way function *f* as follows.

$$\text{seal} = f(\text{sid}, \text{oid}, \text{rights}, \text{seed})$$

[†]Gong [9] calls this pair an "internal capability." We feel the name "internal capability" is a misnomer and prefer to call the secret random number a capability seed because its principal use is in cryptographically sealing capabilities exported from the object server.

The oid and rights components of a capability are exactly as one would expect even in a conventional centralised system. The seal cryptographically embeds the subject identifier (sid) in the capability using the capability seed for that purpose.

4.4 Access Mediation

Access mediation must be incorporated into the RPC (Remote Procedure Call) mechanism of the client-server architecture. The object server must authenticate the source of every RPC request. For this purpose, we assume that each subject has the means to place its digital signature on every RPC communication to a object server. The RPC also carries within it the relevant capabilities for the operation being requested. The object server first verifies that the sid on each capability is authenticated by the digital signature, otherwise the RPC is immediately rejected. Then the object server looks up the capability seed for oid, computes the seal using the above formula and compares the computed seal with the seal submitted by the subject. If these match the capability is known to be authentic and the operation is performed provided the rights are sufficient to authorise it. Digital signatures for the reverse communication from object servers to subjects can also be incorporated. The details of these protocols are beyond the scope of this paper and can readily be found in the standard literature [1, for instance]. We envisage a implementation similar to the interface function box of Amoeba [15] which are placed between each processor module and the network.

4.5 Creation

For object creation the object server consults the access decision facility (ADF) to determine whether or not such creation is authorised by $cc(sid.type)$. If the creation is authorized a new object is created with a new oid and a new capability seed. The rights to be entered on the capability are determined from $cr(sid.type, oid.type)$. Finally the capability is sealed and returned to the subject.

4.6 Internal Transformation

Let subject sid request the following internal transformation for object oid.

$$itrans(u, o, \{x_1, \dots, x_n\}) = \{y_1, \dots, y_m\}$$

The object server must, of course, be a manager for objects of type o. The server checks that $sid.type=u$ and $oid.type=o$. It also checks that the RPC request includes a capability (or capability list) for object oid with the rights x_1, \dots, x_n . This check is performed by comparing the computed seal with the seal on the capability as discussed in section 4.4. Finally the object server creates a new capability sealed for sid with rights $x_1, \dots, x_n, y_1, \dots, y_m$. This capability is returned to the subject sid. Note that the original capability, with rights x_1, \dots, x_n continues to be valid. It is however redundant and can be discarded by the subject.

4.7 Grant Transformation

Let subject sid1 request the following grant transformation for object oid to subject sid2.

$$grant(u, v, o, \{x_1, \dots, x_n\}) = \{y_1, \dots, y_m\}$$

The object server should again be a manager for objects of type o. The server checks that $sid1.type=u$, $sid2.type=v$ and $oid.type=o$. It also checks that the RPC request includes a capability (or capability list) for object oid with the rights x_1, \dots, x_n . If the check is successful the object server creates a

new capability sealed for sid2 with rights y_1, \dots, y_m . This capability is returned to the subject sid1 who can then pass it on to subject sid2.

4.8 Revocation

Revocation has always been a problem in capability-based systems. In distributed systems the problem is further compounded, since the subjects are completely autonomous with no centralised authorities enforcing security. There are various issues against which the implementation of revocation can be compared [19].

1. Partial or Complete: Whether it is possible to revoke a specific right or whether all rights in a capability have to be revoked to get any sort of denial of access in the system?
2. Immediate or Delayed: If the implementation executes revocation immediately or it comes into force only the next time the subject tries to access the object?
3. Selective or General: Does the revocation process affect all users or a select group of users having access over the object?
4. Temporary or Permanent: Is access to be denied permanently or if once it is revoked, is it retrievable?

We provide revocation by a revocation list and a count field appended to the seed as shown below.

oid	seed	count	revocation list
-----	------	-------	-----------------

The revocation list contains entries of sids for whom the rights for that particular oid have been revoked. The list specifies for each sid which of its rights have been revoked. When the validity of the capability is checked during access mediation, the revocation lists are checked in parallel as well. Since access mediation is performed on every operation revocation is immediate. The owner of an oid always has the option to revoke partially or completely the capability of a sid for that oid. Partial or complete revocation of a sid in no way interferes with the access rights of other sids.

The count is a measure that determines the number of valid capabilities for that seed. The count is incremented during creation and propagation, but decremented during complete revocation (i.e. when all the rights of a subject for that object are revoked). Temporary or permanent revocation is carried out, depending on the value of the count. If the size of the revocation list becomes a significant fraction of the count the object server goes ahead with permanent revocation. The server deletes the seed associated with that oid, computes a new one and sends new recomputed capabilities to other associated sids. This of course requires that the object server keep a log of propagation of capabilities. However if the size of the revocation list is small in comparison to the count, the object server goes ahead with temporary revocation. In this case the object server appends the revocation information onto the revocation list associated with that oid.

5 EXAMPLE

The scientist and the security-officer example discussed earlier in section 2 is illustrated here using the protocols described above. A scientist (say Joe) creates a document (say SDI) on his workstation, but before he can release it he needs to have approval from a security-officer (say Sam) and a patent-officer (say Pat). The following is the sequence of protocols needed to complete the task.

1. Joe asks the server to create a document called SDI. This RPC is made by the kernel of Joe's workstation to the appropriate daemon responsible for the server's actions. The RPC contains the action requested, the sid, oid, the types of sid and oid involved, and the actual data to be stored in the created document; all signed under Joe's digital signature. In this case the sid=sci.Joe and the oid=doc.SDI. Joe and SDI are respectively of type sci and doc. On receiving the request, server checks the digital signature to authenticate Joe. The server then checks the cc policy, taking into account the sid, oid and their types provided. If it is in the affirmative it checks the cr policy, by which it determines what rights Joe gets for the document he is creating. The server then pulls out the seed say seed1 for that document and stores it in its internal tables with the following association:

doc.SDI	seed1
---------	-------

Then the object server manufactures the following capability and sends it to Joe (strictly speaking to the kernel of Joe's workstation):

doc.SDI	own, read	seal1
---------	-----------	-------

where $seal1 = f(\text{sci.Joe}, \text{doc.SDI}, \{\text{own}, \text{read}\}, \text{seed1})$

2. Now Joe is ready to release the document. His workstation sends the propagation requests to the server on his behalf. The RPC looks like this:

grant(Sam, review)

doc.SDI	own, read	seal1
---------	-----------	-------

The host when framing the RPC, appends to it the capability it possesses for SDI and signs the request under Joe's digital signature. The server on receiving the request verifies the digital signature and authenticates Joe. Then the server checks the validity of the capability by retrieving the seed of SDI, i.e. seed1, from its internal tables, and computing the seal using the one way function f . Then it computes seal1 from the capability provided by Joe and if the two seals match the validity of the capability is confirmed. The request is then checked against the *grant* policy of Transform. When the server determines Joe has sufficient rights, i.e. own, for SDI, it authorises the grant. The server then computes the capability for the security-officer Sam to have the review right for SDI. The capability

doc.SDI	review	seal2
---------	--------	-------

where $seal2 = f(\text{security-officer.Sam}, \text{doc.SDI}, \text{review}, \text{seed1})$

is sent to Joe. Joe then forwards this capability to Sam. Sam now has the capability for oid=doc.SDI with the review right. With this capability he can only access the document to review it. If he tries to get additional rights by internal transformation, the server will turn down his request because when it will check the set of rights he possesses, namely review, which is insufficient set for it to grant him additional rights. Sam now reviews the document, and if he approves of the action to release SDI he requests the server to grant Joe the approval (a_s) right.

grant(sci.Joe, a_s)

doc.SDI	review	seal2
---------	--------	-------

The server computes the following capability and sends it back to Sam who in turn sends it to Joe.

doc.SDI	a_s	seal3
---------	-------	-------

where $\text{seal3} = f(\text{sci.Joe}, \text{doc.SDI}, a_s, \text{seed1})$

3. Exact similar protocol steps are executed to get the approval (a_p) from the patent-officer Pat. At the end of this session Joe possesses the following capability.

doc.SDI	a_p	seal4
---------	-------	-------

where $\text{seal4} = f(\text{sci.Joe}, \text{doc.SDI}, a_p, \text{seed1})$

4. Now the scientist Joe possesses the capabilities giving him the approval to get the release right by internal transformation. Joe presents these capabilities to the server with the following request:

	doc.SDI	own, read	seal1
<i>itrans</i> (release)	doc.SDI	a_s	seal3
	doc.SDI	a_p	seal4

Like before, the server carries out the authentication and the validity tests on the capabilities presented to it by Joe. Then the server checks that Joe has the rights own , a_s and a_p for SDI which are required to get the additional release right. The server sends him a new capability:

doc.SDI	own, read, a_s , a_p , release	seal5
---------	------------------------------------	-------

where $\text{seal5} = f(\text{sci.Joe}, \text{doc.SDI}, \{\text{own}, \text{read}, a_s, a_p, \text{release}\}, \text{seed1})$

This completes the example.

6 CONCLUSION

In this paper we have proposed a distributed capability-based implementation for the Transform model. The system is based on object servers who act as access-mediators on any attempt by a subject to create, use, acquire, grant or revoke capabilities. We assume a digital signature facility which authenticates the originating subject on each remote procedure call. The capabilities are cryptographically sealed to tie together the identity of the subject, the identity of the object, the rights and a secret cryptographic seed. Strong typing of subjects and objects has also been incorporated.

Our long term goal is to arrive at a practical distributed implementation for SPM (and its recent extension called ESPM [2]). Our first step towards this goal is the implementation of Transform described here. Transform is a sufficiently interesting and non-trivial special case of SPM. At the same time Transform is a sufficiently simplified version of SPM for which a realistic near-term implementation can be contemplated.

Acknowledgment

We are indebted to Howard Stainer and Sylvan Pinsky for their support and encouragement, making this work possible. The opinions expressed in this paper are of course our own and should not be taken to represent the views of these individuals.

References

- [1] Akl, S.G. "Digital Signatures: A Tutorial Survey." *Computer* 16(2):15-24 (1983).
- [2] Ammann, P. and Sandhu, R.S. "Extending the Creation Operation in the Schematic Protection Model." *Proc. Sixth Annual Computer Security Applications Conference*, 340-348 (1990).
- [3] Bell, D.E. and LaPadula, L.J. "Secure Computer Systems: Unified Exposition and Multics Interpretation." MTR-2997, Mitre, Bedford, Massachusetts (1975).
- [4] Clark, D.D. and Wilson, D.R. "A Comparison of Commercial and Military Computer Security Policies." *IEEE Symposium on Security and Privacy* 184-194 (1987).
- [5] Cohen, E. and Jefferson, D. "Protection in the Hydra Operating System." *5th ACM Symposium on Operating Systems Principles*, 141-160 (1975).
- [6] Davies, D.W. "Protection." In Lampson, B.W., Paul, M. and Siegart, H.J. (Editors). *Distributed Systems: An Advanced Course*. Springer-Verlag, 211-245 (1981).
- [7] Dennis, J.B. and Van Horn, F.C. "Programming Semantics for Multiprogrammed Computations." *Communications of ACM* 9(3):143-155 (1966).
- [8] *Department of Defense Trusted Computer Systems Evaluation Criteria*. DoD 5200.28-STD, Department of Defense National Computer Security Center (1985).
- [9] Gong, L. "A Secure Identity-Based Capability System." *IEEE Symposium on Security and Privacy*, 56-63 (1989).
- [10] Harrison, M.H., Russo, W.L. and Ullman, J.D. "Protection in Operating Systems." *Communications of ACM* 19(8):461-471 (1976).
- [11] Harrison, M.H. and Russo, W.L. "Monotonic Protection Systems." In DeMillo, R.A., Dobkin, D.P., Jones, A.K. and Lipton, R.J. (Editors). *Foundations of Secure Computations*. Academic Press, 337-365 (1978).
- [12] Lampson, B.W. "Protection." *5th Princeton Symposium on Information Science and Systems*, 437-443 (1971). Reprinted in *ACM Operating Systems Review* 8(1):18-24 (1974).
- [13] Levy, H.M. *Capability-Based Computer Systems*. Digital Press (1984).
- [14] Minsky, N. "Synergistic Authorisation in Database Systems." *7th International Conference on Very Large Data Bases*, 543-552 (1981).
- [15] Mullender, S.J., van Rossum, G., Tanenbaum, A.S., van Renesse, R. and van Staveren, H. "Amoeba: A Distributed Operating System for the 1990s." *IEEE Computer*, 23(5):44-53 (1990).
- [16] Sandhu, R.S. "The Schematic Protection Model: Its Definition and Analysis for Acyclic Attenuating Schemes." *Journal of ACM* 35(2):404-432 (1988).
- [17] Sandhu, R.S. "Transformation of Access Rights" *IEEE Symposium on Security and Privacy*, 259-268 (1989).
- [18] Sandhu, R.S. "Undecidability of Safety for the Schematic Protection Model with Cyclic Creates." *Journal of Computer and System Sciences*, to appear.
- [19] Siberschatz, A., Peterson, J., and Galvin, P. *Operating System Concepts*. Addison Wesley (1991).

EMPLOYEE PRIVACY AND INTRUSION DETECTION SYSTEMS: MONITORING ON THE JOB

Lorrayne J. Schaefer¹
The MITRE Corporation
7525 Colshire Drive M/S Z 268
McLean, VA 22102

Abstract

The area of intrusion detection systems and privacy has always had a conflict of interest. Intrusion detection systems are designed to help the System Security Officer detect malicious or unauthorized use of a computer system by both unauthorized and authorized users. These systems protect our computer systems from abuse, yet in doing so, it violates our privacy. This paper discusses the legal and ethical issues involved in using an intrusion detection system to monitor the computer system.

Introduction

Down in the street little eddies of wind were whirling dust and torn paper into spirals, and though the sun was shining and the sky a harsh blue, there seemed to be no colour in anything except the posters that were plastered everywhere. The black-mustacio'd face gazed down from every commanding corner. There was one on the house front immediately opposite. BIG BROTHER IS WATCHING YOU, the caption said, while the dark eyes looked deep into Winston's own...

Behind Winston's back the voice from the telescreen was still babbling away...The telescreen received and transmitted simultaneously. Any sound that Winston made, above the level of a very low whisper, would be picked up by it; moreover, so long as he remained within the field of vision which the metal plaque commanded, he could be seen as well as heard. There was of course no way of knowing whether you were being watched at any given moment. How often, or on what system, the Thought Police plugged in on any individual wire was guesswork. It was even conceivable that they watched everybody all the time. But at any rate they could plug in your wire whenever they wanted to. You had to live - did live, from habit that became instinct - in the assumption that every sound you made was overheard, and, except in darkness, every movement scrutinized [1].

George Orwell's 1984 [1] presents a shocking view of a future where everyone's behavior is carefully scrutinized. The feeling that "Big Brother is watching you" is clearly as unsettling now as it was in 1949, and yet intrusion detection technology now allows computer systems to be monitored by electronic "Big Brothers." This raises many legal

¹ This paper reflects work performed while Ms. Schaefer was an employee of Trusted Information Systems, Inc.

and ethical questions as to exactly what privacy rights employees have, and what lengths companies can go to ensure the security of their computer systems.

Computer security is required for enforcing privacy laws. "At the same time, the process of detecting threats, vulnerabilities and abuses may result in violations of privacy and other human rights, leading to a conflict between the use of computer security to guarantee privacy and its use to invade privacy." [2] One area where this conflict is obvious is in the use of intrusion detection technology. This paper will discuss the legal and ethical issues associated with the use of intrusion detection technology in the work place.

Definitions

Intrusion detection systems (IDS) are System Security Officer (SSO) tools, which aid in the identification of malicious or unauthorized use of a computer system by normal system users (insiders) and unauthorized users (outsiders). In other words, the IDS is used to monitor the computer system.

Intrusion detection systems usually get information from raw audit data retrieved from the observed operating system. Typically, the audit data is then reduced for ease of use. This reduction may involve searching for audit records corresponding to specific events that have been previously deemed important, or simply reorganizing all of the audit records into a more generalized format and disposing of fields that are not needed for further analysis.

The raw content of the audit trail may be system accounting information as well as security relevant events. Generally audit records contain such information as subject (e.g., terminal user, process running on behalf of user), object (e.g., file, device), action performed, time stamp, resource measures, indication of any uses of privilege, and an error code. Most intrusion detection systems are designed to observe abnormal patterns of system use such as failed login, unusual user performance (perhaps an unauthorized user masquerading as a legitimate user), Trojan horses, viruses, or an insider attempting to access unauthorized files [3].

Privacy is extremely important to people, yet its meaning, especially for policy purposes, is often unclear. Privacy represents concerns about autonomy, individuality, personal space, solitude, anonymity, and a host of other related concerns [4]. There have been many attempts to define a "right to privacy." Warren and Brandeis defined it as "the right to be let alone." [5] Webster's dictionary defines it as "one's right to freedom from unauthorized intrusion." Dean Prosser wrote that privacy is "in one form or another...declared to exist by the overwhelming majority of the American courts." [6] Prosser identified four types of privacy invasions: intrusion, disclosure, false light, and appropriation. Each of these types depends on physical invasion or requires publicity, and thus offers minimal protection for privacy of personal information.

The Privacy Act of 1974 protects personal data collected by the government. Any individual can request what data has been collected on him/her, for what purpose, and to whom such information has been disseminated. An additional use of the law is to prevent one government agency from accessing data collected by another agency for another purpose. The Privacy Act requires diligent efforts to preserve the secrecy of private data collected [7].

Webster's Dictionary defines *ethics* as "the discipline dealing with what is good and bad and with moral duty and obligation; the principles of conduct governing an individual or a group."

Of course, "good" and "bad" cannot be precisely defined, since they are relative terms that refer in many cases to personal opinion. Consider two co-workers Jim and Mike. Jim does not think it is wrong to take office supplies:

"I am just taking some pens and floppy disks. It's not going to break them."

"It probably won't," Mike replied, "but it's still wrong. It's company property."

Jim did not think this was wrong, but many others feel it is. We are taught in school, by our parents and by our peers that it is morally wrong to take things that do not belong to us; yet many of us still take "a few pens and pencils."

This is also true with monitoring people on the job. Some think it is acceptable to monitor others because it informs individuals as to who is doing their job properly. Others feel it is only acceptable if there is suspicion that a job is not being properly done. Still others feel that any surveillance at all is ethically wrong. The ethics of what should or should not be monitored is discussed later.

The Use of Intrusion Detection Systems and Privacy Rights

The Privacy Act of 1974 made the individual's right to privacy both a legal and ethical issue. There is an ongoing debate now over where an individual's right to privacy ends and a company's right to protect itself begins.

The use of IDS in the workplace has both advantages and disadvantages. A significant advantage is that it can help detect outsiders breaking into the computer system. It can also help detect insiders abusing company resources (e.g., using company time to develop software for personal profit or committing insider fraud or abuse). Monitoring can be quite useful in environments that have little or no protection of sensitive information, in that an intrusion detection system can help detect unauthorized access to the sensitive information. Some disadvantages employee monitoring can create are low employee morale, reduced productivity, destructive countermeasures, and resentment [8], [9]. While security officials or management may believe monitoring the system protects both individual data and company resources, (i.e., it is not meant to watch over the "good guys" but rather to keep the "bad guys" out) programmers, system developers, and other users of the system may feel that they are automatically an "under suspicion" employee. A middle-of-the-road approach states that if IDS operators were carefully restricted and administered, "monitoring of computer activity could be viewed as a benefit by the user community in the same way as security monitoring of luggage at airports is viewed as a benefit by air travellers" [2].

Monitoring on the Job

An example that makes the dilemma between individual and company's rights painfully clear was published in *Information Week* [10] and the *Washington Post* [11]. Alana Shoars was fired from Epson America, Inc. when she questioned management about its

monitoring and reading of electronic messages between employees². There is a question of whether this is a violation of the employees' right to privacy. In 1986, the Electronic Communications Privacy Act (18 U.S. Code 2511) was passed to protect users of telephones and other communications equipment from wiretapping and similar invasions of privacy. The Act also included electronic mail (E-mail), cellular phone service, and other new forms of electronic communication. The Act also extended to communications other than those carried over public networks. It is not clear, however, what rights companies have to monitor the traffic on internal E-mail networks.

There is little question that, at least in the United States, monitoring people without good reason is regarded as socially and ethically unacceptable. Nonetheless, many users of computer systems regard their use of the computer as a personal matter, and a system that watches over their activity could be seen as a violation of privacy. Ironically, people do accept video cameras in banks, airports, and hallways at the workplace. Also, in a shared computing environment, all but novices know that "private" files are not truly private; unscrupulous system administrators and users can examine any cleartext file, and in some cases may be able to read encrypted files. Thus, users generally do not maintain sensitive Privacy Act information on shared systems that lack adequate protection measures. Perhaps the main reason intrusion detection systems appear threatening is that they are designed to judge user activity, specifically to determine whether or not a user is behaving normally or violating some security policy [13].

What to Monitor?

An audit or intrusion detection tool is designed to detect anomalous behavior. Generally, it is intended to aid the SSO in locating the "bad guys" who are circumventing the system. But what about the "good guys"? Exactly how much system activity should an intrusion detection system monitor? In other words, when does this start going beyond a tool and begin invading someone's right to privacy? The IDS will not invade a person's privacy rights if it is monitoring at the node level (login failures). If the IDS is monitoring every keystroke of an individual, this would be an example of invading a person's privacy. On the other hand, the tool could point out that an individual has been poking around files to which she has no access rights. The SSO can then take preventive or preemptive action. There certainly are enough cases of employee fraud where extensive auditing would be deemed not only appropriate but imperative by management. Financial institutions could hardly expect to be insured if a strong audit program were not in place.

As mentioned earlier, people are monitored all the time -- in airports, banks, supermarkets, and department stores, to name a few common places. This usually does not upset people. It is expected that a camera will monitor activity in these areas to help protect both the public and the company assets, as well as to offer a warning to potential trouble-makers.

But what about being monitored on the job? All forms of surveillance and supervision are accepted in factories. Factory workers owe 100% work time when they are on the job in the factory. Whatever workers build in the factory is the factory's property.

² This case was dismissed January 1991 by a superior court judge who said the California wiretapping statute does not apply to E-mail. That suit, however, is pending appeal [12].

This should also be true with white-collar jobs. All scientific discoveries made at work are the company's property. Employees should not spend company resources making several personal calls or revising résumés [14].

There is, however, an unspoken ethic that it is morally wrong to rifle through fellow employee's drawers or files, or eavesdrop on phone conversations.

An employee can consciously protect her files from being monitored. She can do this by either calling an important document something meaningless or by putting file protections on the document to prevent wandering eyes from seeing it. Even so, it is well known that these hurdles can be brought down with little or no effort. The difference between these examples and the knowledge that your system is being monitored by an IDS is that in the former scenario the employee is still comforted with the unknown -- she really is not sure that she is indeed being monitored. The latter case can change employee behavior with the knowledge of being monitored.

As described earlier, employee monitoring may result in low employee morale, reduced productivity, destructive countermeasures, fear, and resentment. As a real-life example, take the boss who automatically has a file sent to him each time someone first accesses their electronic mail. The boss uses the time stamp to determine when that employee has arrived for work; that is, if the employee reads his mail as soon as he arrives. This is a classic example of organizations analyzing patterns of E-mail. Employees can, of course, purposely not read their mail until the afternoon.

As another example, suppose your supervisor, John, approaches you and asks why you can't do an additional task to those currently assigned. You tell John that you don't have enough time. Without your knowledge, John starts monitoring your daily activities using an IDS. John notices that you spend more time reading personal mail than you should. John approaches you later and accuses you of spending an average of two hours reading mail per day and that if you spent less time reading personal mail, you would have plenty of time to do the additional task. How would you feel in this situation? Most people would probably be outraged, resenting the fact that John monitored them without prior permission.

Even if employees know that extensive intrusion detection systems are used, the two examples above illustrate the use of monitoring tools being abused by the unethical. The examples illustrate extreme uses of intrusion detection systems in a "Big Brother is watching" fashion.

Conclusions

Yet examples such as these happen often in the workplace. Using an IDS to monitor the system is an excellent tool to aid the SSO in detecting attempted system breakins or employee abuse of company resources. But this tool also makes it possible to abuse moral issues such as spying on individuals or using the IDS to calculate employee performance. A company using intrusion detection systems must face many legal and ethical questions that to date have not been completely answered. Thus, each organization planning to use such a system should consider these issues.

There are at least two major legal issues that need to be identified. First, whether or not companies have the legal right to monitor computer use, and second, at what level could such monitoring occur. Companies demand the right to monitor computer use to protect

proprietary information and to prevent abuse of computer resources. Companies should have a written policy that describes the extent to monitoring the system.

Even though the legal issues are not well-defined today, these issues should be better understood in the near future. With the *Shoars v. Epson* E-mail case, many companies are becoming more aware of the legal and ethical issues. This case has prompted many organizations to review their policies on system security and employee monitoring, and some companies that previously had no policy on system monitoring have created one.

Companies who do not have a policy could have problems if they have to go to court to defend themselves concerning the monitoring of employees. It is clearly wise for companies to develop a policy regarding the use of IDS. The policy should cover issues such as limits of IDS use, use of the results obtained from monitoring, obtaining informed consent of users, and providing due notice of intent to monitor. The development of this policy should not be limited to security experts, but should involve system users, as well as psychologists, sociologists, constitutional lawyers, and human rights groups [2]. This security policy should be openly available to employees. Each employee should read and sign the policy indicating that they understand and will abide by the rules within. Employees should be advised that they are being monitored when they are using company computing resources. It should be very clear as to what exactly is being monitored and how that information will be used.

Bad policy can certainly become a reality within a company's use on intrusion detection systems. There is also a possibility that the IDS operator can abuse the tool to monitor anything and everything employees do, thereby becoming a kind of Big Brother. Who should or shall oversee that companies do not, in fact, abuse this technology, which is otherwise a great benefit for information security, should be explored to prevent the workplace from being under the constant surveillance of Big Brother and the Thought Police.

References

- [1] Orwell, George, 1984, Harcourt, Brace, and Company, Inc., NY, 1949, pp 6-7.
- [2] Denning, Dorothy E., Peter G. Neumann, and Donn B. Parker, "Social Aspects of Computer Security," *Proceedings of the 10th National Computer Security Conference*, September 1987.
- [3] Schaefer, Lorrayne J. and J. Noelle McAuliffe, "Intrusion Detection Technologies," Trusted Information Systems Technical Report #334, February 1990.
- [4] Federal Government Information Technology, *Electronic Record Systems and Individual Privacy*, OTA-CIT-296, U.S. Office of Technology Assessment, Washington, D.C., June 1986.
- [5] Warren, S.D. and L.D. Brandeis, "The Right to Privacy," *Harvard Law Review*, December 1890, pp 193-220.
- [6] Prosser, Dean, "Privacy," *California Law Review*, vol. 48, 1980, pp. 383, 386.
- [7] *Privacy Act of 1974*, Public Law 93-579, U.S. Code 552(a), December, 1974.

- [8] Irving, R.H., C.A. Higgins, and F.R. Safayeni, "Computerized Performance Monitoring Systems: Use and Abuse," *Comm. ACM*, August 1986, pp 794-801.
- [9] Marx, Gary T. and Sanford Sherizen, "Monitoring on the Job: How to Protect Privacy as well as Property," *Technology Review*, November/December 1986, pp 63-72.
- [10] Caldwell, Bruce, "Big Brother is Watching," *Information Week*, June 18, 1990, pp 34-36.
- [11] Richards, Evelyn, "Privacy at the Office: Is There a Right to Snoop?," Business Section, *Washington Post*, September 9, 1990, pp. H6, H8, H9.
- [12] Eckerson, Wayne, "E-mail Privacy Issue Gains Momentum With Second Suit," *Network World*, February 11, 1991, p. 33.
- [13] Bauer, David S. and Dorothy E. Denning, "Social and Privacy Issues of Intrusion Detection," *Proceedings of the 1st SRI Intrusion Detection Workshop*, March 1988.
- [14] Garson, Barbara, *The Electronic Sweatshop: How Computers are Transforming the Office of the Future into the Factory of the Past*, Simon and Schuster, New York, NY, 1988, pp 205-224.

EXPERIENCE OF COMMERCIAL SECURITY EVALUATION

© Secure Information Systems Limited 1991

Peter Fagan & Julian Straw

Secure Information Systems Limited, Sentinel House, Harvest Crescent, Ancells Park, Fleet, Hampshire, GU13 8UZ, England

ABSTRACT

Considerable experience has been gained in Government funded or controlled facilities in the United States, in the UK and elsewhere in the evaluation of systems and products. This paper discusses experiences gained from the operation and management of a UK Commercial Licensed Evaluation Facility (CLEF), and highlights the issues involved in the marketing of certification to security product vendors.

INTRODUCTION

Two licensed commercial evaluation facilities have been operating in the UK since June 1989. The contracts to operate the CLEFs were granted to two parent companies, Logica Space and Defence Limited and Secure Information Systems Limited (SISL), as the result of a competitive tender process.

In contrast to the existing UK Government funded evaluation facilities, the CLEFs operate on a commercial basis, seeking evaluation work from product suppliers and project sponsors. The CLEFs have now been in operation for two years and have provided unique experience in the area of commercially funded formal evaluations. This paper outlines the procedure for conducting such evaluations, and discusses the issues raised under headings of licensing, staffing, management and marketing. The views expressed are those of the SISL CLEF only.

CONDUCT OF EVALUATIONS

Evaluations are carried out in the SISL CLEF against the evaluation criteria developed by the UK Communications Electronic Security Group (CESG) [1] and also the harmonised European IT Security Evaluation Criteria (ITSEC) [2]. The UK criteria define levels of assurance, describing how confidence is obtained in the design, implementation and operation of a product or system, but without applying constraints to the functionality of any item submitted for evaluation. Because of this the criteria can be applied to systems and products with limited and specific features but which meet high assurance requirements. The ITSEC are compatible with the approach in [1], while at the same time being designed to provide a link to evaluations of products using functionality as defined in the DOD Trusted Computer System Evaluation Criteria (TCSEC) [3]. The ITSEC are the result of an initiative by the UK, France, Germany and the Netherlands, and are based on existing European criteria.

Because functionality and assurance are split, a target evaluation level (e.g. ITSEC E3) is insufficient in itself to define the duration and extent of an evaluation. The information required to control a UK evaluation is provided in two documents: the evaluation baseline and the evaluation work programme.

The baseline document defines the scope of the evaluation work. It states the target assurance level but also states the extent of the functionality of the item under evaluation. For ITSEC evaluations this statement will either refer to or contain a Security Target, which in the case of

a system will describe the security policy for the system. In the case of a product the Security Target will contain a set of claims, describing the security features provided by the product, and a rationale which enables a prospective purchaser to assess whether the product will meet his requirements.

The work programme lists all the work packages making up the evaluation, and states the amount of effort assigned to each.

The baseline and work programme are issued in parallel and both are approved by the certification body before the commencement of an evaluation. This ensures that necessary and sufficient work is planned to allow the product to be evaluated, given the functionality claimed for the product, and that the quality of any evaluation remains unaffected by the competitive situation.

Evaluation then proceeds according to the requirements stated in the ITSEC or the UK criteria, guided by an evaluation manual issued by the certification body, which provides a rationale for standard work packages and describes contents and layout for the mandated reports.

This paper discusses the commercial nature of the relationships between the parties involved in such an evaluation and the issues raised in operating such a facility on a commercial basis. Figure 1 shows the major parties involved in a UK commercial product evaluation and illustrates the flows between them. The roles of these parties are described fully in the UK IT Security Evaluation and Certification Scheme Publication No 1 [5].

The relationship between the facility and the certification body requires commercially sensitive information to be passed from the facility. This is to enable the certification body to monitor progress and assist with the resolution of any problems which arise. The certification body also refers government projects with a requirement for evaluation to the evaluation facilities, in an equitable manner. In the UK both the commercially operated facilities and the certification body are committed to the success of the CLEF scheme and work in concert, given the constraints of their different roles.

LICENSING

On 1st May 1991, the single UK IT Security Evaluation and Certification Scheme was launched in the UK, replacing the scheme under which the CLEFs had been initially established. The new scheme is managed jointly by CESG and the UK Department of Trade and Industry (DTI), under the direction of a management board made up of representatives of a number of UK Government departments. The scheme provides for a single UK Certification Body, reporting to the board.

The SISL CLEF is licensed under the scheme, to carry out evaluations using the methodology common to all UK evaluation facilities. The licence is granted by the certification body under the terms described in UK IT Security Evaluation and Certification Scheme Publication No 2 [6].

The terms of the licence place constraints on the operation of the facility in the areas of security procedures and management. In particular it is a requirement that a quality system which has been accredited by the UK National Measurement Accreditation Service (NAMAS) be in place at the facility. This fact is appreciated by CLEF clients since it increases their confidence in the quality of the work performed. NAMAS is a service operated by the UK National Physical Laboratory (NPL). The criteria used by NAMAS assessors are primarily reliability, quality

and traceability of results. The certificates awarded by NAMAS are recognised widely within the UK, and mutual recognition agreements are in place with a number of European countries.

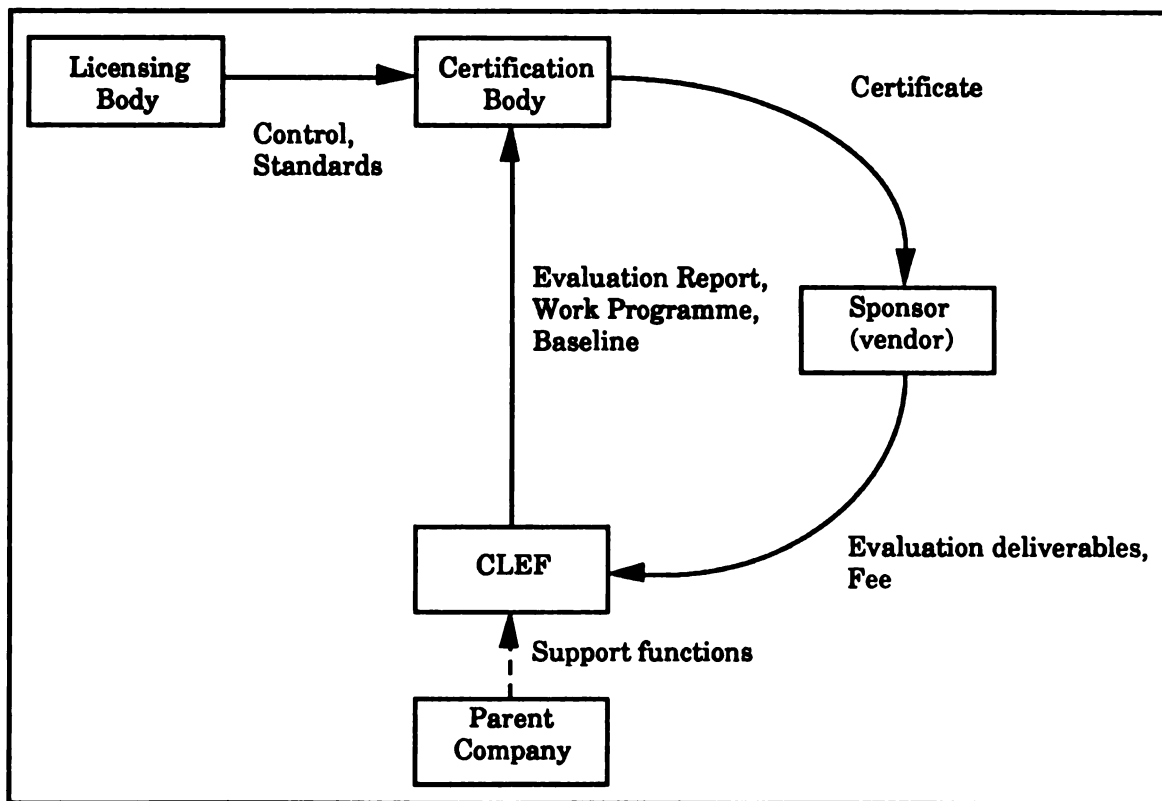


Figure 1 : Commercial Relationships

The licensing terms also require an appropriate management structure to be in place. In general terms this comprises a facility controller, responsible for the overall operation, a business manager (reporting to the facility controller and therefore keeping control on commercially sensitive information), a technical manager (responsible for day to day operation), and an administration manager (whose responsibilities lie mainly in the area of day to day security). In addition there are potentially a number of specialist roles such as methods advisor (responsible for advising on the use of formal methods and associated static and dynamic analysis tools). While it is possible for one individual to hold more than one of these posts, two other posts exist which act as an internal check on the operation of the facility, and which therefore cannot be combined with any of the other roles. These are the posts of quality manager and security manager. It is an important aspect of these two positions that they are independent of the facility controller, in order to assure their impartiality.

The licensing terms for the UK facilities ensure very high standards of work, and the standards are coupled with an official endorsement of the work carried out. Unlicensed companies offering similar services may in some cases be able to undercut the facilities in terms of price; however their work will not carry the authority required for a vendor to achieve the desired marketing benefits provided by a government certificate awarded after evaluation in an approved facility.

STAFFING

In addition to licensing the facilities, CESG operate a separate scheme through which individual evaluators are licensed on the basis of their training and experience. Training courses can be run by a facility subject to approval from the certification body, which reviews the content and quality of the course. This again increases customer confidence and allows the licensing body to ensure that suitably qualified staff are used on evaluations.

The requirement for licensed staff creates a problem for a commercially-operated facility, where the flow of work may be irregular. Training and licensing of individuals constitutes an investment which must be used in the facility if it is to bring benefits. Therefore at the end of an evaluation, when qualified staff potentially become unassigned, there is a need to retain them in the CLEF and not to return them to the parent company, where they may be assigned to long term projects and thus become unavailable to the facility. A stable and self-contained community of evaluators is to be desired. However, only limited overheads in terms of low staff utilisation levels can be tolerated in a commercial environment. These two conflicting requirements can only be reconciled if there is a stable and reasonable flow of evaluations into the facility, which in turn requires a commercially-oriented approach to operation and marketing.

In order to minimise costs and risks to the CLEF, personnel are assigned for the duration of an evaluation, and only very exceptionally are they removed for other work within the facility.

The possible consequences of breaches of commercial confidentiality affect staffing. While document security, procedural security and physical security can be adequately addressed by the means usually adopted by defence contractors, personnel security is an area requiring increased attention. Non-disclosure agreements are made on an individual basis, so as to confine the spread of information to those with a need to know. This agreement continues beyond the lifetime of the evaluation. In addition, constraints are placed on the management of facility staff, so that their deployment outside the facility will not place them in positions where they could use information gained during the evaluation to the commercial disadvantage of the vendor. Monitoring of staff who leave the facility and the parent company remains a problem.

Staff motivation is an issue within CLEFs. Since one aim of evaluation and licensing is to achieve standardisation, a danger exists that staff can be left with a feeling of insufficient autonomy. This issue is considered to be an important one within the SISL facility since staff motivation is a prerequisite for high quality work.

Autonomy and feedback on performance are two major factors affecting motivation, secondary issues being task significance and task integrity. Autonomy needs to be a feature of a facility, with early responsibility and customer contact. While commercially desirable technical skills are gained during the evaluation of a product, these tend to be knowledge of the construction of the product rather than knowledge of its use. Also there are constraints on the use of the knowledge gained during the execution of an evaluation. Autonomy can provide experience which compensates for this. Similarly, early and frequent feedback to staff on performance and problems must be a feature of a primarily participatory management style. By incorporating the commercial aspects of evaluation (e.g. proposal preparation, presentations), into all positions in the facility, task significance and task integrity can be achieved. In the experience of the authors this is best carried out by sharing the marketing work and administration tasks.

MANAGEMENT

Commercial confidentiality is a major issue in the management of a facility as well as in the licensing.

The key benefit which vendors see in obtaining certification for a product is that of obtaining a marketing asset. This is particularly true in the case of certification to the ITSEC. In contrast to the TCSEC, increased emphasis is placed on the development environment for a product, successful evaluation reflecting upon the company and its development process just as much as on the product. Therefore the timing of the announcement of certification, and the confidentiality of the results of evaluation are important factors in UK evaluations.

This requirement for commercial confidentiality arises in part from the commercial nature of CLEF work. Knowledge of any corporate action expected shortly to provide an improved market share might be considered by many vendors to be sensitive information. Where longer timescales are expected, as is the case in the US, the early announcement of formal evaluation may be beneficial. Since UK commercial evaluations are conducted with the minimum evaluation effort commensurate with the maintenance of the enforced standards, there are in contrast, potential benefits for a vendor choosing confidentiality.

This means that great care must be taken with the handling of customer identities within the facility, and also within the parent company where facilities such as accounts, sales and marketing are used. Identities of prospective and actual customers are disguised by an internal numbering scheme, with only the minimum number of staff knowing for whom the work is being carried out.

Strict measures are put in place within the facility to provide commercial confidentiality. Primarily this is a matter of physically separating teams working on separate evaluations, and providing secure storage facilities for each. Preferably teams should use dedicated computer equipment which is flushed between evaluations, since working on two evaluations simultaneously on the same computer places an increased dependence on logical separation of user groups.

The SISL facility is physically separated from its parent company, with a separate entrance. This separation reduces the risk of accidental disclosure via documents or conversations. In addition, prospective customers can be seen without the knowledge of personnel in the parent company. A log of visitors to the facility is kept, and managed in a way which prevents one prospective customer from seeing that another has visited. The same is true of any document recording the identity of more than one customer (e.g. facsimile log, business reports). Visitor passes are issued by the facility and not by the parent company so that no record is kept of the visit by the parent company. Separate telephone and facsimile lines are provided so that customers and prospective customers can be assured of the confidentiality of their project.

For all tasks, individual registers are kept of all deliverables supplied to the facility whether or not there is a requirement to handle classified information. At the end of the evaluation the deliverables provided to the CLEF are either returned or destroyed, as agreed between the facility and the client. Appropriate destruction procedures are among those defined by the Security Operating Procedures (SOPs) under which the facility operates. The SOPs, approved by the certification body, define the procedures for day to day management of the facility and for the handling of the client evaluation deliverables. The existence of documented procedures is essential for consistent application of confidentiality and quality requirements.

From the point of view of confidentiality the facility can be seen to comprise a number of operating groups: those with knowledge of prospective customers (the facility controller and the business manager, together with any staff involved with sales support); those with knowledge of a particular evaluation; and those with overall knowledge of the CLEF operations. In fact this last group consists of a single individual, the facility controller.

While it can be seen that there are management problems in operating a CLEF as an arm of a parent company, there are advantages also. It is unlikely, for example, that any one evaluation will be a significant fraction of parent company turnover; therefore cash flow on one task is unlikely to be a significant factor for the overall health of the parent.

The primary problem in the management of CLEF evaluations is one of maintaining control on costs. During the evaluation this is exemplified by the problem of evaluating a product in which minor faults may be found which must be corrected before certification. Clearly this will require some re-evaluation, and a suitable strategy must be chosen during contract negotiation which will allow this to take place within the constraints of what is usually a fixed price contract.

In entering a contract with a vendor the facility is in the unusual position of performing a service without guaranteeing a result, since it does not itself award the certificate. The evaluation results cannot currently be provided directly to the vendor, and the same is true of information concerning faults which may be found in the system or product. These are sent instead to the certification body who can release them (or not) to the vendor. This can lead to situations where vendors may attempt to impose unacceptable constraints on the evaluators, such as penalty clauses in the event of the certification body not responding within defined timescales.

Commercial risks to the CLEF in entering into a fixed price contract are naturally a management issue. The availability of deliverable items such as design documents and source code has an impact on timescales and costs. In order for the evaluation to proceed the deliverables must be provided at an early stage, and it is usual for a contractual clause to exist which will protect the facility in the case of these items being delayed or being unavailable. To guard against the effects of this situation, clear lists of required deliverables are provided to vendors at the time of submission of a proposal by the CLEF.

The commercial liability which a CLEF is prepared to accept is defined by the terms of its insurance cover and by the status of the reports which it produces. The SISL CLEF is covered for example for security evaluations, but not for safety critical uses. The legal status of an evaluation report is that of a statement that the product has been compared against a certain standard; not that the CLEF is guaranteeing the product to be secure. This is obviously essential to protect against third party claims for consequential loss.

A final management issue concerns the use of tools in the CLEF. For a commercially operated facility the use of tools in areas such as source code analysis is justified where a saving will be made or where the quality of the evaluation will be improved. For example at higher levels of assurance it may be necessary to use a tool to achieve the required confidence level. Previously this area has to some extent been one of academic research, and the tools which will be necessary to derive commercial benefit for a CLEF may be different to those which are currently being developed.

PRE-EVALUATION CONSULTANCY

Vendors consider the price of an evaluation as speculative investment, and an internal marketing case may have to be provided before senior management will allocate a budget for an evaluation. Evidence may have to be provided by the technical department that it is confident of a successful result. To meet this need the facility offers pre-evaluation consultancy. The aim of this activity is to highlight areas which should be addressed before committing to an evaluation proper. A review period is sometimes beneficial, so that any corrective action can be verified.

The aims of pre-evaluation consultancy depend on the requirements of the vendor. Frequently the vendor will wish to understand more fully the evaluation process and the risks which are being accepted. To meet this requirement the facility produces a vendor report which describes the deliverables required for a target level, and assesses the available deliverable items against those requirements. It may be that as part of the consultancy, a vendor will authorise an evaluation at a level below the desired level, as a cost-effective check on the evaluatability of the product, or just to confirm the target level. Vendors may also wish to compare the requirements of an NCSC evaluation against those of an ITSEC evaluation, to determine the effectiveness of an ITSEC evaluation in terms of addressing the European market. An important form of pre-evaluation consultancy is in the preparation of a baseline and work programme. The agreement of the certification body is required before an evaluation can commence, and the controlling documents are required before such agreement can be given. Therefore where a CLEF enters into an evaluation contract without having agreed the baseline and work programme, it does so at its own commercial risk. A clear contractual and licensing distinction is drawn between evaluation and this form of consultancy.

Therefore the common aim of all pre-evaluation consultancy can be seen to be to reduce risk in the evaluation phase, both to the facility and to the vendor.

During the management of any form of pre-evaluation work it is important for the CLEF to maintain impartiality. It has been suggested that CLEFs should be debarred from performing evaluations where they have provided pre-evaluation consultancy. The basis of this argument is that a conflict of interest can arise if a facility first determines the suitability of a product for an evaluation which it then subsequently carries out. There are dangers in this view for all parties. It is unlikely that other organisations offering such services will be licensed or policed in the same way as CLEFs, and undoubtedly to protect their commercial interests there will be disclaimers attached to the results. Such consultancy will be provided in the absence of experience of evaluation itself and in the absence of up to date knowledge of the remit of the approved facilities. Most importantly the consultancies will be taking on the role of the CLEF during the period in which a CLEF would be gaining experience in the product and building a relationship with the vendor. If a CLEF were to come in at the evaluation stage without having reduced their own risks beforehand, the net effect would be higher prices for evaluation, reflecting a higher contingency, in the light of possible contractual and quality problems arising from the previous stage.

MARKETING

Evaluation is currently considered to be primarily a vertical market, in that the skills sold by the evaluation facilities are narrow in range and are applied in a similar way to varying sizes of project. However the SISL CLEF has found that skills are gained during evaluation which should allow a broader base to be established, and which would enable evaluation expertise to be applied across a wider range, possibly by applying subsets of the skills (e.g. source code analysis services, secure product design reviews).

There are some unique issues to be addressed in marketing these, and other, CLEF services.

In order to understand the marketing issues in any industry it is useful to split the market into appropriate segments (e.g. by customer type, contract size or geographical area). Segmentation of the evaluation market, and the parameters which could be used, are issues yet to be addressed in detail by the UK evaluation community. The primary reason being that the low level of activity means that there is a limited amount of information to gather and thus analyse. However it will soon be necessary for answers to be supplied to questions such as 'how is the market split?' and 'how does our CLEF expertise map onto that split - what market share will that give us?'. Initially however it can be assumed that at higher levels the evaluation market is predominantly for certification of products for use in Government systems. The price of evaluation for these products may be considered by vendors to be the price of admission to the market.

Publicity following certification is another marketing issue for vendors. Press releases are effective to a degree in alerting the public to a product undergoing evaluation, and in the UK this has been employed at the lower end of the market. This is useful for the facilities since it also alerts other vendors to the expected benefits. However, the facilities are still bound by their agreements on confidentiality and this is a constraint on their marketing operations.

Currently the UK market for evaluation is a latent one (the market is considered to have potential but currently it is not running at a very high level of activity). In these circumstances a pro-active approach is required to identify and stimulate market areas. This is made more important to the CLEFs by the vendor requirement to reduce through-life costs. In short, when a vendor has undertaken an evaluation with a facility, and the quality of the original work has met expectations, the staff of that facility will have been trained in the design of the product. It will make commercial sense for the vendor to return to the same facility for subsequent re-evaluations. Thus it is important from the point of view of the facilities to be pro-active, since repeat business is generally considered to be cheaper to obtain than new business, and since there are a finite number of product vendors.

In a competitive environment a CLEF must decide on a marketing stance. Although performing the original work has been stated to be a factor in winning repeat business, it can be expected also that the so-called 'marketing approach' will be the optimum stance in the long term. In this the CLEF seeks to understand specific vendor requirements and to match the work to the requirements, providing customer satisfaction as an internal goal. In a field in which it is clearly possible to provide a service on a basis of 'take it or leave it', the marketing approach can be expected to provide distinction to a CLEF.

Publication of the ITSEC has undoubtedly raised awareness of the process of certification. The harmonised European criteria are increasingly relevant to the marketing activities of European subsidiaries of US companies. When the planned framework for the ITSEC has been put in place, the validity of evaluations will be accepted throughout a number of countries, with the evaluation scheme recognising fully the TCSEC functionality which many vendors will have incorporated. The use of the ITSEC will provide a number of benefits for commercial evaluation facilities, primarily removal of the requirement for published interpretations of the criteria in particular circumstances or for particular applications. The UK facilities have not for example been delayed in database evaluations by the absence until recently of an accepted version of the Trusted Database Interpretation [4]. However, in comparison to the US market the UK market is small, and therefore the flexibility of criteria such as the ITSEC has not been exploited on a scale necessary to achieve significant marketing benefits as yet for the facilities.

The issue of cost-benefit analysis as performed by a vendor must be considered. It has already been stated that in simple terms a vendor will see evaluation costs as a speculative investment which can be expected to reap rewards in terms of increased sales.

This is nowhere more true than in situations where a vendor is aiming for a low level of assurance in a simple product. A statement of assurance gained by an approved facility in the correctness of a product, coupled with a statement from the vendor of his security claims, provides a marketing asset sufficient to distinguish a product from its competitors. The total cost of evaluation for a Personal Computer (PC) security add-on at the lowest assurance level might be in the region of \$7,000-\$12,000. The elapsed time would be a matter of a very few weeks. Therefore even distributors (rather than manufacturers) can and do consider this as a feasible investment. For a comprehensive PC security product, moving up to a higher level of assurance could cost \$40,000-\$60,000 and would run for perhaps 12-15 weeks. This is a different sector of the market, and different reasons for acquiring certification apply.

MAINTENANCE OF CERTIFICATION

Aside from the initial costs, through-life costs are an issue for vendors, and are therefore a marketing issue for CLEFs. For product vendors at any level there is an overriding commercial benefit in being able to control the costs of certificate maintenance. It may for example prove more cost-effective for the vendor to decouple a certificate maintenance programme from the usual product release cycle. The ITSEC take into account the quality of the development environment, and a vendor may be content to run through two or three bug fixes or releases before going for recertification, relying on customer confidence in the certification of the development environment. Strong configuration control requirements for a product enable the impact of changes on product security to be closely monitored and assessed. A commercially acceptable scheme for maintenance of certification will provide control to the vendor over the timing and size of expenditure.

The separation of assurance from functionality has allowed small companies to gain certification for simple products at low levels of assurance. Maintenance of certification at these levels is generally accepted to be almost a re-evaluation. The costs of maintaining certification as the product evolves will probably be significant in terms of the vendor company turnover. Nonetheless, in the UK scheme the costs and timings for re-evaluation are under the control of the sponsor. There is no requirement for open ended support or liaison with the evaluation facility or certification body, which would be inappropriate for a small organisation.

For larger products or systems, the nature of re-evaluation is different. Work must be done during the evaluation to allow the certification maintenance to proceed, by constructing a database of security relevant areas. After certification, changes are notified to the evaluators by the vendor, and the evaluators assess the impact on certification, providing where necessary, third party evidence that certification remains unaffected. However at this level also the costs and timings of re-evaluation remain under the control of the sponsor.

SUMMARY

The relative advantages and disadvantages of commercial security evaluation are summarised in Figure 2 below.

The market for commercial evaluation in the UK remains in its infancy. The UK CLEF scheme has established a model for its development which has now been tested and refined,

and which will be sufficient to meet an expanded market. The ITSEC provide a widely applicable basis upon which to build, and are expected to generate significant interest from international markets in the use of UK facilities.

The establishment of the CLEFs has required considerable effort in the definition of procedures for licensing and certification. It has also called for the resolution of problems concerning commercial confidentiality and staffing. Market development has called for careful examination of the requirements and motives of potential customers, to determine how best their needs may be satisfied.

It is now believed that the UK model for commercial evaluation facilities, with its emphasis on quality management and responsiveness to market needs, can provide the basis for a significant expansion in the supply of evaluated products available internationally.

Advantages		Disadvantages
Commercial (CLEFs)	reduced timescales no queues reduced client restrictions adaptable to customer needs	cost to vendor not well known in US
Government (NCSC)	free widely known	long timescales not development env only US systems queues

Figure 2 : Comparison Summary

REFERENCED DOCUMENTS

1. **CESG Computer Security Memorandum Number 3**
UK Systems Security Confidence Levels, Issue 1.1, February 1989
2. **Information Technology Security Evaluation Criteria (ITSEC)**
Harmonised criteria of France, Germany, the Netherlands, the UK
Version 1, dated 02 May 1990
3. **Department of Defense Trusted Computer System Evaluation Criteria,**
DoD 5200.28-STD, December 1985
4. **Trusted Database Management System Interpretation of**
Trusted Computer System Evaluation Criteria, NCSC-TG-021, Version 1,
April 1991
5. **UK IT Security Evaluation and Certification Scheme Publication No 1**
Description of the Scheme
Issue 1.0, dated 1 March 1991
6. **UK IT Security Evaluation and Certification Scheme Publication No 2**
The Licensing of Commercial Licensed Evaluation Facilities
Issue 1.0, dated 1 March 1991

EXPERIENCES IN MULTI-LEVEL SECURITY ON DISTRIBUTED ARCHITECTURES

Karl A. Siil
AT&T Bell Laboratories
1 Whippany Road, Rm 14E-218
Whippany, New Jersey 07981

ABSTRACT

This paper describes the port of a Multi-Level Secure (MLS) operating system to a multi-processor distributed architecture. The implementation involved porting AT&T's B1 Rated System V/MLS to the AT&T 3B4000 super-minicomputer. Although originally a port of the System V/MLS *operating system*, the 3B4000 port provided valuable experience in solving problems associated with MLS networking. This type of experience is required for the creation of future secure system solutions. Because, just as it is unlikely for a modern computer not to be networked, it is equally unlikely for an MLS computer not to have need of MLS networking.

INTRODUCTION

This paper describes the port of AT&T's System V/MLS to the AT&T 3B4000 super-minicomputer. During this port, the 3B4000 distributed architecture's resemblance to a network unexpectedly provided answers to and performance data on MLS networking issues that were not part of the original goals of the porting project.

MLS networking is a relatively new and unexplored territory that is in demand by customers in both the government and commercial realms. Theoretical pursuits, although constructive for determining avenues of endeavor, are limited by the many real-world issues that, as yet, cannot be expressed as equations, proofs, or algorithms. Worked examples of MLS networking are needed to confront and resolve such issues. The System V/MLS 3B4000 port has acted as one such worked example.

The paper starts with brief overviews of System V/MLS, the 3B4000 distributed architecture, and UNIX[†] System V on the 3B4000 concentrating on those elements that are relevant to multi-processor and network security. It then goes on to present a set of porting requirements determined before starting the port. The paper then discusses network security issues encountered and tackled during the port. It goes on to discuss the impact of the requirements and network security issues on the System V/MLS kernel modules, commands, and libraries. The paper then shows how achievement of the requirements was verified. Lastly, the results are extended to network security in general showing how the porting experience can be applied to the development of MLS networking products.

SYSTEM V/MLS AND 3B4000 OVERVIEWS

This section presents overviews of System V/MLS and the 3B4000 distributed architecture. This is to provide the reader with the basis for the discussion of what was undertaken in the port and how the results of the port can be applied to MLS networks.

System V/MLS

System V/MLS (SV/MLS) is an NCSC B1 Rated version of the UNIX System V operating system. The first

[†] UNIX is a Registered Trademark of UNIX System Laboratories.

UNIX system to achieve a B1 Rating, SV/MLS is fully compliant with the System V Interface Definition (SVID) and introduces no more than 4% performance degradation with full auditing enabled. For this paper, the most important components of SV/MLS are label management and audit subsystems.

The SV/MLS Mandatory Access Control (MAC) policy is a modified version of the policy described by Bell and LaPadula.^[1] Subjects and objects are labeled via an overloading of the conventional UNIX GID field. Unlike conventional UNIX systems, this field is not a dimensionless number on SV/MLS. Instead, it is an index into a file. Each element in the file is composed of the data necessary to form an SV/MLS *privilege*. A privilege can be thought of as an instance of a conventional UNIX group at a given MAC label. An example of two privileges is shown in (Figure 1).

Privilege = Discretionary Group + MAC Label

For example, in the *ls(1)* output:

```
-rw-r----- 1 karl X[TS] 31056 Jun 11 13:19 Mission_Data
-r--r----- 1 karl X[S] 58547 Apr 17 11:03 Design_Doc
```

The privileges, X[TS] and X[S] could be:

Privilege	Group	Label
X[TS]	Project_X	Top Secret
X[S]	Project_X	Secret

Figure 1. System V/MLS Privilege Components

When an access check is performed, privilege information is required by the access control software. SV/MLS uses a cache to hold frequently referenced privileges. The cache is checked before any privilege information is read from the disk file. The file and cache have become known as the *labels file* and *labels cache*, respectively. Technically, both contain more than labels. But, to maintain a smooth flow in the paper, the commonly used terms shall also be used here. Also, the act of getting information from either of the labels cache or labels file has become known as, *getting a label*. That terminology will be used here, as well.

There are three (3) components to the SV/MLS audit subsystem. These are:

- A kernel-resident audit trail data buffer,
- An audit trail daemon process, and,
- A set of audit trail data files.

The SV/MLS audit trail is composed of binary records collected from 25 probe points throughout the kernel, as well as 16 trace devices accessible only by trusted processes (e.g. *login(1S)*, *passwd(1S)*, etc.) via nodes in */dev*. The binary data is sent from the probe points and trace devices to the kernel buffer. The buffer is periodically read by the trusted daemon process which then sends the audit data to the files.¹ When a System Security Officer wishes to review the audit data, he/she passes the binary data through a filter program to convert the data to a human-readable format.

1. The audit trail daemon can write to *any* form of writable media (files, printers, network ports, etc.). Most SV/MLS sites use files, however. So, the remainder of this paper only discusses files as targets for audit data.

AT&T 3B4000 Distributed Architecture

The 3B4000 super-minicomputer is a multi-processor system composed of up to 16 Processing Elements (PE's). The system is composed of a 3B15 Master Processor (MP) with up to 15 Adjunct Processing Elements (APE's) physically connected by a network fabric known as the A-bus.^[2] Though the hardware is a star topology, the 3B4000 kernel software supports logical point-to-point and broadcast type messaging between PE's. Each PE runs its own UNIX kernel. The APE's depend on the MP to provide services for many operating system functions (e.g., process creation, clock synchronization) and cannot run autonomously for any great length of time. However, each APE maintains data structures for its local files, inodes, mounted file-systems, devices, etc. These data structures are similar to those of uniprocessor UNIX systems.

When a PE requires a resource/service from another PE (e.g., a remote file access), it issues an A-bus message requesting that service. If the server PE can satisfy the client PE's request immediately, the client waits for the results. If not, the client process requesting the information goes to sleep and gives control of the CPU to the next runnable process. When the information requested from the server arrives, the requesting process is made runnable and, when given the CPU, it retrieves the data and continues its work.

One main goal of the designers of the 3B4000 version of UNIX System V was to maintain application compatibility at the object code level. The internals of many system calls were augmented to handle the need for sending and/or receiving messages over the A-bus. But, most of these changes are invisible to applications programs.

PORTING REQUIREMENTS

Before the SV/MLS 3B4000 port began, several documents were referenced to determine the porting requirements. The three (3) primary sources used were: The Orange Book,^[3] the uniprocessor SV/MLS Design Documents,^[4] ^[5] and the 3B4000 UNIX System V Design Documents.^[2] Using these sources and a few others, requirements were determined for the SV/MLS 3B4000 port. In the list that follows, the requirement itself is in *italics*. Additional information is provided to explain the requirement and its purpose.

- <A1> *SV/MLS on the 3B4000 must maintain the same system call interface and operation as the uniprocessor version of SV/MLS.* From the onset, it was known that SV/MLS would require non-trivial changes for the port. This requirement was designed to localize the SV/MLS changes to the kernel components. Any kernel change that did not alter the system call interface from that of the uniprocessor SV/MLS would aid in reducing the number of changes to SV/MLS user-level commands and libraries.
- <A2> *Performance degradation on the 3B4000 because of SV/MLS must be no more than 5% compared to a non-SV/MLS 3B4000 system as measured by industry accepted benchmarks.* This requirement is the same as the SV/MLS uniprocessor performance requirement. It is present to prevent the creation of a *secure brick*. Performance degradations greater than 5% will cause users and system administrators to disable the security features of SV/MLS and, therefore, make its presence on the system useless.
- <A3> *The 3B4000 SV/MLS audit trail must be able to be written to any PE on the system.* An extension of a uniprocessor SV/MLS requirement, this is to assure that the audit trail can be written to any writable device no matter which PE that device is on.
- <A4> *The 3B4000 SV/MLS audit trail must contain data to allow the determination of the PE(s) that a given event occurred on.* Since PE's are addressable objects in the 3B4000 UNIX System V, successful and failed accesses to them and the resources they control must be audited.
- <A5> *A System Officer must be able to boot and shutdown APE's without shutting down the entire 3B4000 system. The booting and shutting down of APE's must be audited.* This stems from an original 3B4000 requirement. A goal of the 3B4000 is to provide long spans of uninterrupted service. As such, the shutdown of a given PE must be allowed without shutting down the rest of the system. Also, booting (shutting down) APE's adds (deletes) objects from the users' address spaces. As such, the booting (shutting down) of an APE must be audited.

The above requirements are listed to show the basis for design decisions described later which impacted various components of SV/MLS. As necessary, particular requirements will be called out at the relevant sections.

SECURE NETWORKING ISSUES ENCOUNTERED

When extending the concepts of secure computing to multiple CPU's, many concepts that are simple in a single processor system become somewhat more complex. Two issues that had to be dealt with in the SV/MLS 3B4000 port were distributed auditing and label management.

Distributed Auditing

On a single processor system, a single thread of control is guaranteed since there is only one CPU executing instructions and, therefore, creating auditable events. In a multi-processor system or network, each CPU is generating auditable events. Also, each CPU may have its own clock. This is especially likely in a network. As such, when auditing events on a multi-processor system or network, the time-stamp of an audit record must be associated with the clock of the CPU that generated it.

An audit record from a given CPU must be ordered with respect to the records of the other CPU's. Determining the skew between the clocks or synchronizing them becomes important if the audit data is to have meaning. Also, since multiple CPU's can cause *simultaneous* events, there must be a way to determine which events are actually caused by other events and which are totally unrelated.

In addition to having correct time-stamps and proper ordering of the audit data from multiple CPU's, secure multi-processor and networking products must decide where the audit data is to be stored. Audit data could be stored at the processor local to the events until review of the data is required. Or, all the audit data could be sent to a central collection point as the audit records are being generated. While centralized auditing provides ease of access to the whole networks audit data, it could increase network traffic to the point where the entire network grinds to a halt. Localized auditing keeps the network clear, but each processor is required to have a substantial amount of local storage for the audit data. The latter solution is unacceptable in a diskless workstation environment.

Label Management

Similar to the issue of distributed auditing, label management has to do with global access to a set of globally significant data. In auditing, this data is almost entirely *write-only*. Conversely, the data and operations associated with label management are almost entirely *read-only*. As in distributed auditing, centralized versus localized label repositories can be used. The two primary concerns are:

- network traffic associated with distributing labels around the network, and,
- synchronization of labeling information between label repositories.

A centralized label repository requires no synchronization, but all but one processor must do non-local I/O to get a label. On the other hand, localized label repositories eliminate most, but not all, network traffic associated with label passing. Some traffic must still occur to update the label repositories of processors not local to where a new label has been defined for the overall distributed system or network. In addition, a communications scheme must be developed to provide *strong* assurances that the label repositories are always synchronized for access decisions to be made correctly.

SV/MLS has a file that is used, among other things, to translate the privilege associated with every subject and object to a human-readable label. When dealing with a multi-processor system or network, however, a mechanism for managing labels between CPU's must be implemented. For most distributed systems or a network where all the hosts have come to agreement on a labeling standard, the mechanism could be as simple as a shared resource that is accessible by all processors. In that case, an implementation similar to that of SV/MLS could suffice. For large or heterogeneous networks, the label management mechanism might have to be complex with labeling domains and mapping functions to translate between domains. This might be required if only subsets of the systems on a network could come to agreement about what label convention to use.

When communicating labeled information out of one of these clusters of systems to another cluster, a label mapping algorithm would have to be used to translate the labels of one domain to another.

3B4000 SV/MLS IMPLEMENTATION

This section presents the changes implemented in SV/MLS to meet the requirements and achieve workable solutions to the network security issues presented above.

Audit Subsystem

In analyzing the 3B4000, it was discovered that the MP synchronizes all its APE's clocks at least once per second. As such, for the SV/MLS 3B4000 port, no additional work was required in the area of clock synchronization.

AT&T has already tackled the data ordering issue in its securing of UNIX Remote File Sharing (RFS) for use with SV/MLS.^[6] The SV/MLS RFS strategy is to audit accesses on the *server* that contains the *object* being accessed. As the SV/MLS 3B4000 port progressed, other similarities between the 3B4000 distributed architecture and RFS were noted. As such, it was decided to use the AT&T RFS/MLS auditing strategy for the 3B4000. This became known as the "Object PE Records Audit," or OPERA rule. Given the single-threaded, non-preemptable nature of the 3B4000 PE kernels, and because auditing at the object PE assures that audit records relevant to the object remain ordered, implementing the OPERA rule in the SV/MLS 3B4000 audit subsystem was enough to meet the SV/MLS auditing requirements.

For placement of the audit data, it is possible for the three (3) audit subsystem components described earlier to be resident on up to three (3) different PE's. This capability was implemented to meet <A3>. However, for performance reasons, it was later found that all three components should reside on the same PE. The remainder of this paper assumes that all three components are on the same PE, called the Security Audit Trail Processing Element (SAT PE).

The SV/MLS 3B4000 port implements a centralized audit trail. The SAT PE is the central repository for audit data. This solution was chosen because some 3B4000 PE's do not support local mass storage. As such, these PE's would have to send their data to a remote audit trail repository anyway. Therefore, the centralized audit trail was chosen in the interests of a simple solution applicable to all PE's. All PE's send their audit data to the SAT PE via the A-bus. As such, only the SAT PE needs to allocate a storage area for audit trail data. However, A-bus bandwidth is consumed by audit data traveling to the SAT PE from all the other PE's.

The question arises, "What should the other PE's do when the SAT PE goes down?" In theory:

- a. all subsequent reference monitor requests on all other PE's should fail/hang, and/or,
- b. no more auditable events of any kind can occur on the entire system.

In determining whether reference monitor requests should fail or user processes should be suspended, it was found that it didn't matter. If the SAT PE goes down, it cannot be rebooted without tripping an audit point. Granting access to and executing the program to reboot a PE are auditable events.

Under normal circumstances (i.e., the SAT PE is up), the SAT_START records that audit the boot of a PE are cut by the MP and sent to the SAT PE. Under the SAT-PE-crashed case, the MP cannot audit the (SAT PE) boot event and will therefore fail or hang, depending on the implementation chosen. With this the whole 3B4000 will eventually suspend, since the MP runs the A-bus. Sooner or later each APE will request an MP service and hang waiting for the request to complete. In light of this, the "if there's no SAT PE, shutdown" solution was chosen. If a PE generates an auditable event and can't send it out over the A-bus to the SAT PE, the originating APE attempts to send a "go to single-user mode" request to the MP. If this request succeeds, the APE waits for the MP to shut down the whole system. If the request fails, the APE panics.

A final note on the auditing requirements. To meet <A4>, all 3B4000 SV/MLS audit trail records have fields that identify the PE's involved in the auditable event that generated the record. The PE numbers stored in the records depend on the auditable event type and the OPERA rule. To meet <A5>, a new audit trail record,

SAT_STOP, was implemented for the auditing of the crash or intentional shut-down of an APE. Also, the meaning of the **SAT_START** record was expanded to include the booting of APE's in addition to the booting of the MP.

Choosing a SAT PE

The choice of SAT PE can make a difference in the performance of the whole 3B4000 SV/MLS system. This section provides guidelines determined through review of the 3B4000 design documents and by experimentation on the live development and production systems.

The first thing determined was that, if possible, the SAT PE should not be the MP. If any APE has mass storage capabilities, it should be used as the SAT PE, over the MP. The MP is best left free to move traffic around the A-bus.

Given the OPERA rule, the SAT PE should be the one that *contains the greatest number of objects accessed most frequently*. This set of objects includes files, directories, pipes, and System V IPC structures. The placement of the audit trail files local to the bulk of the auditable activity greatly reduces the amount of A-bus traffic, offloading the MP and generally improving throughput.

As a final note on SAT PE placement, it was found that making the SAT PE and the PE that contains **/bin**, **/usr/bin**, and **/tmp** one and the same helps throughput significantly. Actions on objects in these three file-systems can amount to the bulk of the auditable events on a UNIX system.

Label Management Subsystem

The uniprocessor SV/MLS maintains all label information in use on the system in the labels file. This was not changed for the SV/MLS 3B4000 port. There were concerns at first that such a file on only one PE would greatly increase A-bus traffic, causing the product not to meet <A2>. But, by using per-PE caches similar to the cache of the uniprocessor product, it was believed that <A2> could be met. A single file with label information also eliminates the need to synchronize many such files on multiple PE's. The MP was chosen as the keeper of the labels file for a variety of reasons. The most significant reason was that since the MP is always the first PE to be booted and it must have access to the labels file to boot itself and the other PE's, then, if there is only one labels file, it must be on the MP.

Since label information is centrally stored on the MP, there must also be a way for APE's to get labels. The algorithm used is a convenient extension of the uniprocessor SV/MLS version. In the uniprocessor version, if the access control software requires a label, it first looks in the labels cache. If the label is not there, one or more reads of the labels file are performed. When the label is retrieved, it is placed in the cache for future reference. If the cache is full, the least used label (determined by a per-cache-entry hit count) is overwritten by the newly gotten one.

On the 3B4000, the same algorithm is used on the MP, since the labels file is local. However, if an APE's access control software requires a label, it first checks an APE-local cache. If the label is not found, the APE sends an A-bus message to the MP requesting it. The MP first checks its cache and if the label is not there, the MP gets the label from the labels file.

The MP cache check is important because the odds are that the label is already there from a previous access check. When users log in, among the first things they access is **/etc/profile**, which is on the root file-system of the MP. By virtue of the OPERA rule, the access checks are performed on the MP and the label is deposited in the MP labels cache. Even if a user never accesses another MP object, the label remains in the cache (within the constraints of the replacement algorithm). This saves a considerable amount of disk accesses which are much more expensive than A-bus messages.

Any distributed or network label management system must account for the deletion of labels. When a label is deleted, instances of the label in caches throughout the 3B4000 system must be rendered invalid. The SV/MLS 3B4000 port uses a global reset message which is sent by the MP to all APE's when a label is deleted. This message causes each APE to invalidate *all* entries in its cache. The caches must then be reloaded over time as

labels are required. This is not the most elegant of techniques. But, given that label deletion is a relatively rare event, the technique was acceptable.

System V/MLS Commands and Libraries

Because of <A/>, few user level routines had to change. Those that did change were augmented to handle additional capabilities and/or features of the 3B4000.

The most evident difference in the SV/MLS user interface on the 3B4000 versus a uniprocessor system is the *prlbl(1S)* command. The interface to and output of *prlbl -s* had to be changed to accommodate multiple labels caches (one per PE). An administrator can use *prlbl -s* to display label cache hit statistics. Based on this information, the administrator can "tune" the cache for optimal performance. Since each PE has its own labels cache, the interface used by *prlbl(1S)* to get the statistics from the SV/MLS kernel modules had to be changed. Also, the display of the cache information was changed. The old display is shown in (Figure 2).

```
# /mls/bin/prlbl -s
cache hits = 2638826
one read from disk = 52
more than one read from disk = 9
cross product found on disk = 4356
```

Figure 2. Output of a uniprocessor *prlbl -s*

For the 3B4000, the *prlbl(1S)* display has been augmented to show PE specific cache hit information. The new display is shown in (Figure 3).

```
# /mls/bin/prlbl -s
```

PE	chit	dh1t1	dh1t2	rh1t	conlbl
121	10482652	14970	20347		10788
0	1010			16	0
80	2918229			16785	6076
120	31919			3781	0

Figure 3. Output of a 3B4000 *prlbl -s*

The columns specified are:

chit - cache hits,

dh1t1 - retrieved label from disk with one read,

dh1t2 - retrieved label from disk, required more than one read,

rh1t - retrieved label via A-bus transaction,

conlbl - *cross-product* privilege constructed from group of one privilege and label of another.

ACHIEVEMENT OF REQUIREMENTS

The previous sections have shown the implementation choices made to meet the requirements and network security issues of the SV/MLS 3B4000 port. This section presents the steps taken to assure that these implementation choices were valid.

Among the best ways to determine if <A1> was met was to examine the amount of change to the system test software/procedures. AT&T offers the *AT&T System V/MLS Security Test Package*,^[7] (STP) which is intended to determine whether a particular implementation of SV/MLS meets the SV/MLS security requirements and interface definition. This package is primarily for source customers to evaluate a port of SV/MLS to their particular platform(s). It was expected that STP would have to be changed to test the PE fields in the audit data, plus the new SAT_STOP and enhanced SAT_START records. No other changes were expected before starting the testing of the port. After testing was completed, the only changes, except for those expected, were to accommodate the 3B4000 package installation mechanisms, device names, and directory hierarchy, which differ slightly from those of the AT&T 3B2 or 6386, the platforms STP was originally designed for.

Early on in the port, the cache hit statistics showed that <A2> would be met. From the data in (Figure 2), the uniprocessor cache hit ratio comes out to show that 99.8 percent of all label references are resolved with cache hits. Using the data from (Figure 3), the average 3B4000 SV/MLS labels cache hit ratio was determined to be 99.5 percent.²

As the port neared completion, the Neal Nelson benchmark was used to compare the 3B4000 SV/MLS performance figures against *vanilla* 3B4000 UNIX System V figures, taken on the same system, before the port began. The performance degradation was not measurable using the Neal Nelson benchmark, beyond differences attributed to statistical error. This showed that <A2> had been met.

In addition to data provided by the Neal Nelson benchmark and the Security Test Package, the SV/MLS 3B4000 port currently has over 6500 system-hours of hands-on use. Since the final load, no SV/MLS related shut-downs have taken place. Therefore, a good deal of confidence exists that the implementation is sound.

LIMITATIONS OF 3B4000 PORT RESULTS

Although the SV/MLS 3B4000 port provided many insights into network security issues, two shortcomings in the solutions were found. These two shortcomings are presented in this section.

Clock Synchronization

The 3B4000's existing clock synchronization simplified the implementation of the audit trail considerably. Had the synchronization not been there, it would have had to have been implemented. This surely would have made things much more complex. Most networks, however, *do not* have synchronized clocks between the hosts on the network. As such, for a usable audit trail to be created based on the SV/MLS design and the 3B4000 porting work done to date, some type of synchronization mechanism must be developed. Given that, the lessons learned from the 3B4000 port can be applied to the remainder of the network audit trail implementation.

Limitation of the OPERA Rule

The OPERA rule has one shortcoming. Although the OPERA rule implementation allows system administrators to answer questions like:

- Who was the last person to open /etc/passwd for writing?
- What happened first; did Lucy write to the file, xyz or did Andy read it?

Where the OPERA falls short is in answering questions like:

- Did Audrey write to the file, ABC on PE80 first, or did she read the file, QXR on PE121, first?

The OPERA rule cannot be used to answer this question. And, if the two events mentioned above are close in time (within the granularity of one system clock tick), the ordering cannot be accurately determined in the

2. Both the 3B4000 and uniprocessor data come from configurations with 10 entries per labels cache.

3B4000 SV/MLS audit trail. An analogous rule, "Subject PE Audits Request," or SPEAR, could answer a question such as the previous one. But, given the same timing conditions, SPEAR could not answer the questions that OPERA could. A combination of OPERA and SPEAR must be implemented to completely order the audit trail for all cases. Because of other facets of the 3B4000 architecture beyond the scope of this paper, SPEAR was not implemented, as the events that it is needed to distinguish are rare.

APPLICATIONS TO SECURE NETWORKING

Building on the experiences of the SV/MLS 3B4000 port, one could apply the results of the port to produce an MLS network based on the SV/MLS design. This section applies the SV/MLS 3B4000 port design decisions to the secure networking issues, distributed auditing and label management, described earlier.

Distributed Auditing

The Neal Nelson benchmarks and day-to-day use of the SV/MLS 3B4000 system show that the implementation of centralized auditing and the associated network traffic for audit data necessary to achieve a B1 evaluable network is possible without sacrificing performance. The centralized approach also provides a solution for networks that have components, such as diskless workstations, that don't have the ability to store audit data locally.

The effort to determine which PE of a 3B4000 should be the target for the centralized audit files showed that the choice of a target for audit trail data does make a difference in performance. Given the OPERA rule, the audit trail should be collected *on or as near as possible to the network component(s) that contain the greatest number of objects that are accessed most frequently*. The goal is to limit network audit trail messages as much as possible. The SPEAR rule mentioned earlier, adds the additional constraint that auditing data should be collected *on or as near as possible to the network component(s) that contain the greatest number of subjects that produce the most audit data*. OPERA and SPEAR need not be conflicting rules. In client/server networks with server processors containing database objects and server process subjects, OPERA and SPEAR work together well. The database accesses account for a large amount of the auditing affected by OPERA and the actions of the servers account for a large amount of the auditing affected by SPEAR. Since both types of auditing come from the same network component, the audit files could be centralized on or near the most active server(s), thereby satisfying both OPERA and SPEAR. Of course, real-world solutions are never so cut-and-dry. But, the use of OPERA and SPEAR as guidelines, if not rules, can help determine where to place auditing in a secure network.

Label Management

Like distributed auditing, the performance data and system usage experience show that centralization of label information is practical from a performance perspective. With proper use of caches, the network traffic associated with distributing labels from a central repository is not prohibitive. For distributed systems and networks with all elements in agreement on labeling, efficient centralized label management proves to be realizable. For those (larger) networks mentioned earlier that cannot come to agreement, a central label repository per *agreeable* cluster is also shown to be realizable by the SV/MLS 3B4000 results.

A subtle, but important concept that emerged from the label distribution algorithm was the checking of a labels cache local to the label repository before going to the repository itself. If a user is likely to access an object on the network component containing the centralized label repository, then that component's labels cache will contain the label in question when the access control software of another component requires it. One real-world occurrence of this phenomena is a network where a centralized database is searched to determine if a user has access to the given network. If that database is on the network component that also contains the label repository, then the cache searching technique presented above will decrease the number of times the repository (rather than the cache) is searched for labels. Such a configuration would exist in a network architecture with a central *security server*.

CONCLUSIONS

This paper described the port of AT&T's System V/MLS to the AT&T 3B4000 super-minicomputer. During this port, the 3B4000 distributed architecture's resemblance to a network unexpectedly provided answers to and performance data on MLS networking issues that were not part of the original goals of the porting project. The SV/MLS 3B4000 port acted as a worked example to show that practical and efficient MLS networks can be built and that the SV/MLS design is a practical base for an MLS network. With the increasing demand for secure networking products and services, this provides welcome evidence that secure systems don't have to be unusable systems.

ACKNOWLEDGEMENTS

I thank Bill Leighton and Pete Dinsmore for letting me "find" the time to prepare this paper, as well as providing the funding and support. I thank Glen Gordon for sanity-checking potential designs and helping me understand the internal operation of the 3B4000 system. I thank Cheri Dowell for her work in system testing the SV/MLS 3B4000 port. I also thank the Department 46291 system administration staff for letting the port loose in their lab. And, finally, I thank all the members of Department 46291. By acting as an (involuntary) Beta site, they helped iron out the last kinks in the system.

REFERENCES

1. Bell, D. E., LaPadula, L. J., *Secure Computer Systems: Unified Exposition and Multics Interpretation*, MTR-2997 Rev. 1, MITRE Corp., Bedford, Mass., March 1976.
2. Fish, R. W. (ed.), et al., *DSB-720300CD - AT&T 3B4000 Kernel, Component Design Specification, Issue 3*, AT&T Information Systems, May 20, 1988.
3. National Computer Security Center, *Department of Defense Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD, December 1985.
4. *System VIMLS Multilevel Security (MLS) Module - Requirements, Design, and Implementation*, AT&T Bell Laboratories, August 9, 1990.
5. *The System VIMLS Security Audit Trail - Requirements, Design, and Implementation*, AT&T Bell Laboratories, June 27, 1990.
6. Johnson, E. M., *Secure Remote File Sharing on SVIMLS*, AT&T Bell Laboratories, June, 1, 1989.
7. *AT&T System VIMLS Security Test Package User's Guide*, AT&T Bell Laboratories.

AN EXPERT SYSTEM APPLICATION FOR NETWORK INTRUSION DETECTION*

Kathleen A. Jackson, David H. DuBois, Cathy A. Stallings

Computer Network Engineering Group, MS B255
Computing and Communications Division
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

This paper describes the design of a prototype intrusion detection system for the Los Alamos National Laboratory's Integrated Computing Network (ICN). The Network Anomaly Detection and Intrusion Reporter (NADIR) differs in one respect from most intrusion detection systems. It tries to address the intrusion detection problem on a network, as opposed to a single operating system. NADIR design intent was to copy and improve the audit record review activities normally done by security auditors¹. We wished to replace the manual review of audit logs with a near realtime² expert system. NADIR compares network activity, as summarized in user profiles, against expert rules that define security policy, improper or suspicious behavior, and normal user activity. When it detects deviant (anomalous) behavior, NADIR alerts operators in near realtime, and provides tools to aid in the investigation of the anomalous event.

1 Introduction

The authentication and access control system in any network is the first defense against intruders from outside. At Los Alamos, we define authentication as the identification of a user with reasonable assurance that the user is who he or she claims to be. Access control is defined as a mechanism of restricting access by authenticated users to those parts of the network consistent with their clearance and need-to-know. It is clear, given the industry-wide frequency of break-ins by outsiders, that authentication and access control mechanisms can be compromised or bypassed. They alone cannot supply assurance against penetration by outsiders. Also, outside "hackers" are not the only source of security problems. Far more often they are a result of abuse by the privileged insider. Even the most secure system is vulnerable to abuse by insiders who misuse or try to misuse their privilege. This is obvious from well publicized reports of incidences of unauthorized access and removal of classified information by insiders from otherwise secure computer systems.

In a large, complex, and rapidly changing computer network such as the ICN it is not realistic to expect to identify all security loopholes and vulnerabilities. Even if identified, it is not a given that they can be closed, since it may be impossible or impractical to do so. A primary reason for this is the need to strike a balance between security and the provision of convenient services to network users. Given the acknowledged doubt in the completeness of current security measures, we are tasked to identify and implement new technologies that support network security.

An auxiliary line of defense against both intrusions by outsiders and insider misuse is the maintenance and review of an audit record of important network activity. In our case, maintenance of an adequate audit record presents few problems. This has been a required activity at Los Alamos for many years. However, attempts at audit record review result in security auditors wading through huge quantities of output in an ineffective attempt to spot invalid activity. The sheer vol-

*The Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36. This work was performed under auspices of the United States Department of Energy.

¹ Los Alamos security auditors are specialists whose responsibility is to ensure the security of the ICN. They include security officers such as the CSSO and CSRM, and their staffs.

² For our purposes, we define a near realtime application as one that responds to data or user input in one to 30 seconds.

ume of data makes it nearly impossible to detect suspicious activity that does not conform to a few obvious intrusion or misuse scenarios. Even these may be missed. To make audit review effective, the auditors need the capability for automated analysis of the audit record. This capability combines the knowledge of security experts with a computer's capability to process and correlate large quantities of data. When done in near realtime, the auditors can be notified of suspicious activity quickly, and direct action taken to trace and stop an identified penetration attempt or other misuse. This is the essence of an intrusion detection system.

2 Target System

The Integrated Computing Network (ICN) is Los Alamos National Laboratory's main computer network. It includes host computers, file storage devices, network services, local and remote terminals, and data communication interfaces. The core of the ICN includes the main host supercomputers and their support devices. Through the ICN, any user inside the Laboratory may access any host computer (with authorization to do so and use of an approved access path) from office workstations or terminals. Outside users typically access the ICN through telephone modems, leased lines, or one of multiple world-wide networks. The core ICN has more than 8,000 validated users.

The ICN consists of a unique arrangement of four "partitions," in which resources are dedicated to specific levels of processing. Each partition limits access to only those users cleared for the most sensitive information processed in the partition. A system of dedicated, special function, ICN nodes enforce partitioning throughout the network. These *service nodes* perform specific services in the ICN, such as user authentication, access control, job scheduling, file access and storage, file movement between partitions, and hardcopy output. They are physically protected, have tightly restricted access, run only that software needed to perform a specific service, and do not execute user programs. Only these dedicated nodes may service multiple ICN partitions. Each of these nodes must produce and maintain an audit record of its activity. They are the ICN systems targeted for our intrusion detection effort.

3 Overview

Until recently, security auditors manually reviewed ICN audit records to identify potential security violations. Given the size of the audit records, manual review was limited to a small sampling or a cursory scanning. The auditors found many security violations, but there was no way to evaluate the general success or completeness of their effort. Also, the Laboratory's Internal Security (ISEC) office often requests audits that cover weeks of audit data from months or years in the past. As there was no automated way to do these audits, considerable effort was expended in completing them. It was for these reasons that development of an automatic audit record analysis, or intrusion detection, system was undertaken at Los Alamos.

The early research of Dorothy Denning and her colleagues, and the IDES research and development at SRI International, has heavily influenced intrusion detection development at Los Alamos. Denning proposed monitoring standard operations on a target system for deviations in usage. Her early research tried to define the activities and statistical measures best suited to do this [1, 3], and continued with the development of an IDES prototype [4]. Teresa Lunt and her colleagues continue this research with the development of the IDES system [5, 6, 9, 13]. They expanded the original concept by adding an expert system component that addresses known or suspected security flaws in the target system. IDES research has served to demonstrate two things. First, that statistical analysis of computer system activities provides a characterization of "normal" system and user behavior, and that activity deviating beyond normal bounds is detectable. Second, that known intrusion scenarios, exploitation of known system vulnerabilities, and violations of a system's security policy are detectable through use of an expert system rule base. The IDES approach puts a primary emphasis on the statistical detection of deviations from normal user and

system behavior. The expert system is intended to catch those invalid activities missed by the first means [10].

Several intrusion detection systems have in recent years adapted the Denning model to their particular problem [7, 8, 11]. However, where the Denning model and most intrusion detection systems target specific operating systems, our effort addresses a *network* connecting many host systems, but not the hosts themselves [15]. Where Denning addressed the standard operations on a specific operating system (system logon, program execution, file and device access) we wished to address the standard operations on our network. The problems are similar in many respects, but with some important differences. While the ICN contains many standard functions such as those found on an operating system (authentication, access control, file access and storage, job control), these functions are distributed across the network. Also, the ICN implements a *distributed* multi-level secure system (the system of partitions and the controls over them), that must be monitored closely by any intrusion detection system. Nonetheless, if we view the ICN as one large distributed operating system, then the Denning model applies well to the problem of network intrusion detection.

Current network intrusion detection efforts have taken one of two approaches. One approach is to target network traffic at the service and protocol levels [12]. The second approach collects data from separate hosts on a network, for processing by a centralized intrusion detection system [14]. Although NADIR does not capture network traffic, it targets service level activity by targeting the service nodes that handle and log standard ICN service operations. We decided to target the *service nodes* because of their critical nature, to keep the quantity of data to be processed at a manageable level, and because their audit record is sufficient to support an effective intrusion detection system.

4 Working Prototype

Once we decided to apply intrusion detection to the ICN service nodes, we adopted three basic technical goals. These goals support development of a flexible system that we could expand to multiple target systems. The first goal was to limit the audit record to that currently supplied by the target systems. The second, to keep target system changes to a minimum. The third, to avoid degradation of target system performance.

Because the ICN is a large, long-established network that has changed constantly over the last fifteen or so years, we had to take the following peculiarities into account:

- The Los Alamos developed network protocols are non-standard, so are not compatible with off-the-shelf software.
- The ICN service nodes comprise several different hardware configurations, that run a variety of operating systems.
- The software on most service nodes has been subject to many changes and upgrades, and is programmed in several different languages.
- While each service node must maintain an audit record of its activity, the format and content of the audited data differ greatly from system to system.

To support expansion to these various multiple target systems, we made three design choices. First, to use dedicated workstations for intrusion detection processing. Second, to use flexible off-the-shelf interface and database software, that supports data translation between different operating systems and enables the merging of data into a single extended database. Finally, to limit required target system changes to the capability to collect the proper audit record of user activity, transform the data into a specified canonical format, and transmit it to NADIR. Also, we designed NADIR software in a modular fashion, so that new target system expansions can be handled with a minimum of effort.

NADIR is to be implemented on a set of dedicated workstations, each of which will receive and correlate data from multiple target systems. As we add more target systems to NADIR, we plan a network of workstations, each contributing to a distributed database. This approach minimizes the impact on target system performance, enables the collection of data from multiple diverse systems, and provides for maximum security. Ethernets will connect the workstations to the target systems and to each other, and we will implement a standard network protocol.

The NADIR prototype consists of one workstation, a SUN SPARCstation³ with two 327 MByte disks. It uses the Sybase⁴ relational database management system and a Los Alamos designed expert system. Sybase provides tools used to structure, maintain, and display all data on the system. The expert system is programmed almost entirely in Transact-SQL, an enhanced version of the SQL database language supplied by Sybase. Transact-SQL provides such capabilities as stored procedures, triggers, system administrator tools, and control flow language features, used extensively in NADIR. NADIR communicates with each target system over a dedicated secure ethernet link.

The prototype NADIR currently monitors Network Security Controller (NSC)⁵, Security Assurance Machine (SAM)⁶, and Common File System (CFS)⁷ activity on the ICN. The NSC is a DEC-8250⁸ machine, which runs the VMS operating system. The SAM is a DEC-730 machine, which runs the UNIX⁹ operating system. The CFS is a IBM 3090 mainframe. NSC and SAM data is transmitted directly to NADIR, while CFS data is passed to an intermediate VAX/VMS system before transmission to NADIR. The changes called for on each target system were minimal. Communication with NADIR by a target system calls for only the installation of Sybase supplied interface software, and the use of a standard DECnet or TCP/IP protocol. DB-Library packages for Fortran and C provide the interface to Sybase. The Multinet¹⁰ software package provides an implementation of TCP/IP under VMS. We changed each target system code as little as possible. The target system must only format the audit record for NADIR and transmit it immediately after its occurrence. NADIR required data processing has not resulted in any measurable degradation in system performance on any target system.

5 System Design

We are applying NADIR to the ICN service nodes in a sequence of planned phases. Each phase includes analyzing a node individually, processing its data separately, then integrating it into the NADIR system. As we add new nodes to NADIR, we correlate their user activity record with earlier included nodes to produce more complete profiles of ICN activity. Eventually, this will allow the tracking of users from the time they enter the ICN, until they leave the network. With the addition of each node, we define new expert rules that use the expanded information available. The rules describe more elaborate scenarios of invalid or suspicious user activity, and will, over time, improve the discrimination and judgement of the system. We have integrated the NSC, the SAM, and the CFS into NADIR. Work is in progress to integrate the Facility for Operator Control and User Statistics (FOCUS)¹¹ and the Print and Graphics Express Station (PAGES)¹². These are all the nodes initially targeted for prototype development.

³ SUN SPARCstation and SUN workstation are trademarks of SUN Microsystems, Inc.

⁴ Sybase, Transact-SQL, and DB-Library are trademarks of Sybase Corporation.

⁵ The NSC is a dedicated, single-function computer through which all ICN user authentications must pass.

⁶ The SAM controls and audits the down-partitioning of unclassified files between partitions in the Common File System (CFS).

⁷ The CFS is a large, centralized file management and storage system that provides long-term file storage in all ICN partitions for ICN users.

⁸ DECnet, VMS, DEC-8250, and DEC-730 are trademarks of Digital Equipment Corporation.

⁹ UNIX is a trademark AT&T Bell Laboratories.

¹⁰ Multinet is a trademark of TGV, Inc.

¹¹ FOCUS provides operations control, batch job scheduling, and accounting control for the ICN.

¹² PAGES produces listings, graphics, and formatted document output for ICN users. Output is subject to partition and classification restrictions.

The NADIR system has six functional components; Data Collection, Data Processing, Anomaly Detection, Report Generation, Event Assessment, and the User Interface. Figure 1 illustrates their relationship to each other.

5.1 Data Collection

NADIR monitors target system activity as it happens. Each audit record describes a single event. Audit records from different target systems vary in format and contain mostly unique data, a result of the functionally different tasks done by those systems. Whatever the system, the audit record will contain a unique ID for the ICN user, the date and time of the user's activity, fields that describe the activity, and any errors that might have occurred.

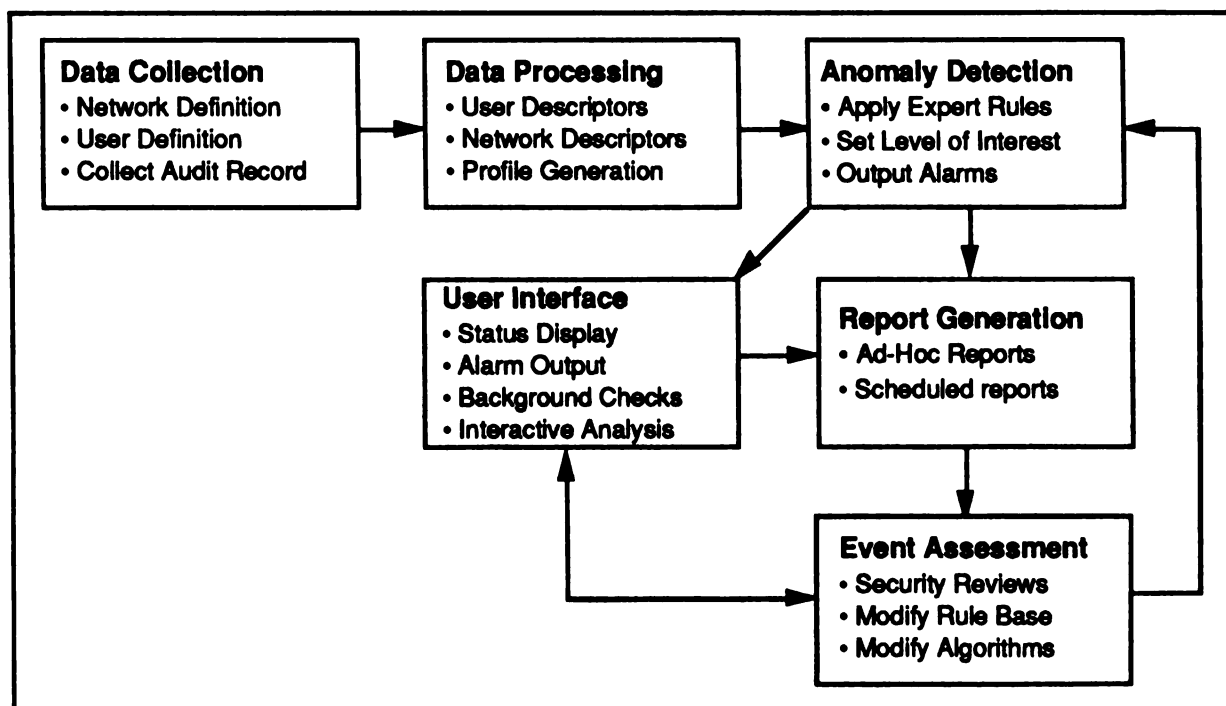


Figure 1: NADIR System Model

5.2 Data Processing

NADIR summarizes all user and system activities, as represented by audit records from the target systems, into statistical profiles. These profiles are a description of current behavior in a set of defined parameters. NADIR maintains profiles for both separate ICN users and for a composite or total of all ICN users. They contain measures (count statistics) that summarize user activity. These measures keep a record of the occurrences of a particular event during a specified time. NADIR updates the profiles when it receives an audit record. It parses the data from each audit record and increments the proper profile measures. NADIR maintains past profiles for comparison purposes and as a permanent record.

5.3 Anomaly Detection

NADIR finds events in either the contents of a single input audit record or from an examination of the user profiles. Single audit records define an event when any of the data fields in the record match a specified pattern. Events detected in the profiles represent activity that is spread across multiple audit records. They define an event when the profile measures match a specified pattern.

NADIR compares proper and expected activity to observed events within either the audit record or the profiles. It does this through the application of expert rules, and identifies deviations¹³. NADIR assigns each deviant event (or anomaly) a Level-of-Interest¹⁴. It bases the Level-of-Interest on the number and type of rule that the user's behavior has fired. NADIR applies the Level-of-Interest to each unique user, host system, or entry point into the network. Every fired rule increases the Level-of-Interest, though the firing of one critical rule may be enough to bring immediate attention to the event. The current security status for each user and system is provided in the combination of Level-of-Interest and record of fired events.

5.4 Report Generation

NADIR generates anomaly reports from deviant events. The frequency of reports is dependent on the Level-of-Interest associated with each event. All events are documented in routine weekly reports. Those events determined to be very interesting, but not critical, are output in daily reports. Very suspicious events of a critical nature, such as a probable attack under way, are output immediately. NADIR generates detailed follow-up reports as part of any investigation.

5.5 Event Assessment

Upon receipt of a NADIR report, whether critical or routine, security auditors review all anomalous activity. To process anomaly reports quickly, specific auditors investigate certain categories or types of ICN users. They review each anomalous user in detail, and decide whether to investigate further. This may include interviewing the user. If the user's activity warrants it, the user is blacklisted¹⁵ during the investigation. The auditors file a short report at the completion of each investigation, giving details of its resolution. They supply this information to us, so we may have immediate feedback on system performance. The auditors hold periodic reviews to evaluate NADIR effectiveness and to make recommendations for improvements. We use their feedback to change the expert rules on NADIR and improve the discrimination and judgement of the system.

5.6 User Interface

The user interface uses Sybase front end tools, graphics packages, and Los Alamos designed routines to provide a preliminary interface for the knowledgeable user. It provides warnings, alarms, and status displays. For users who have the proper access and privilege, the user interface allows a choice of built-in or ad-hoc queries against the raw audit data, the separate user and composite profiles, and status information. Data may be displayed in a variety of ways, including graphically, and reports generated. In addition, NADIR provides tools for interactive background analysis of current and past activity. It maintains indefinitely the audit data needed for this activity.

6 Expert Rules

An expert rule base has separate reasoning rules encoded in a condition-action form (if-then-else statements in the old days), that provide the criteria for end determination. The rules watch for unusual separate events and attempt to evaluate the meaning of a group or series of events. NADIR expert rules, whether they are rules that enforce security policy or result from a statistical determination of normal behavior, define an expected standard of behavior for all users.

¹³ The identification of a deviation by an expert rule is generally referred to as having "fired" or "triggered" the rule.

¹⁴ The Level-of-Interest is the calculated seriousness of an event.

¹⁵ A blacklisted user is denied access to the ICN by the NSC. Removal of the blacklist requires the prior approval of security personnel.

The NADIR rule base includes four logical filters; each designed to separate out certain types or levels of anomalous activities. Following a knowledge engineering approach successfully implemented at Textronic [2], the rule base definition started with the abstraction of the well-understood part of the problem. This included ICN security policy and well-defined invalid and suspicious behavior, which resulted in rules for the Characteristic Filter. Report requirements supplied rules for the Report Filter. From there evolved further refinements, implemented in the Misuse and Attack Filters. These rules involve heuristic associations that sometimes make intuitive leaps not always explicitly justified. NADIR activates the rule base filters in stages, as illustrated in Figure 2.

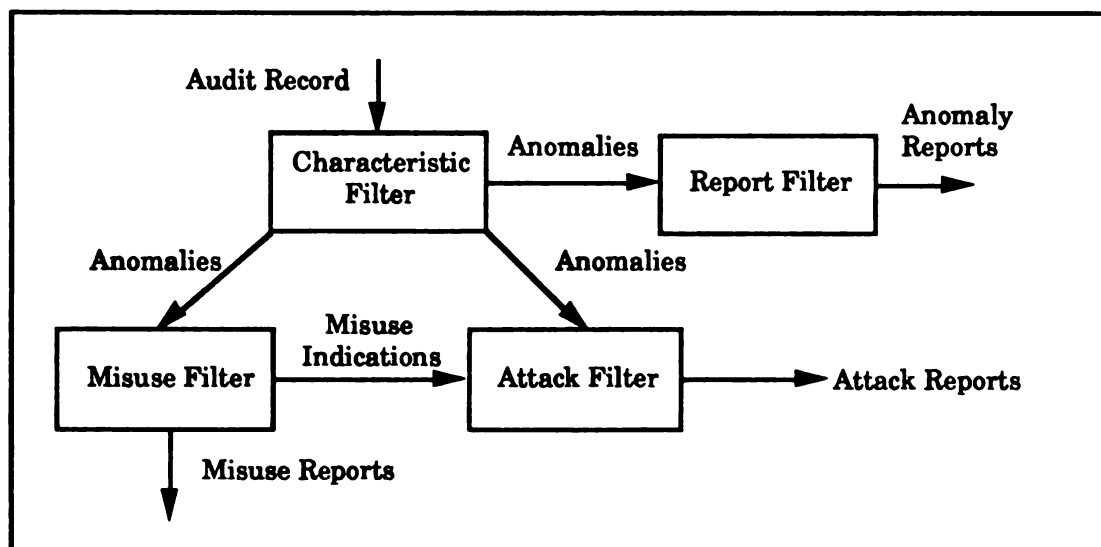


Figure 2: NADIR Rule Base Structure

- **Characteristic Filter** - applies rules that are straightforward descriptions of simple activities; each serving to distinguish a separate feature of anomalous behavior. NADIR applies these rules individually; it does not correlate one with another. It assigns a Level-of-Interest to each anomaly defined by these rules. This Level-of-Interest, as applied to each user or system, is incremental; with each rule fired it increases by a specified amount.
- **Report Filter** - applies rules to the anomalies output by the Characteristic Filter, to produce appropriate reports of anomalous behavior.
- **Misuse Filter** - applies rules to the anomalies identified by the Characteristic Filter. These rules try to identify patterns of anomalous activity that have a good chance of being systematic misuse. They specify what action to take when fired, such as the output of warning messages.
- **Attack Filter** - applies rules that try to correlate the recorded Characteristic anomalies and Misuse Indications with various Attack Scenarios. Attack Scenarios identify patterns of anomalous activity that have a good chance of being attacks on the system. They specify what action to take when fired, such as the output of alarm messages.

6.1 Characteristic Rules

NADIR applies Characteristic rules to either the input audit record or to profile data. As it finds each anomaly, it either generates or updates the Anomaly Record, whichever is appropriate. The

Anomaly Record includes a Level-of-Interest for the involved user or system, and an indication of the fired rule. Characteristic rules fall into three basic categories:

1. Security Policy - These rules are the implementation of ICN security policy. They result from interviews with security personnel and documentation reviews. They detect and immediately report potential or certain security violations. An example of a security violation rule:

IF NADIR has detected an "Improper Location" error,
 AND the terminal used is in the Open Partition,
 AND the password used is classified,
THEN update the Anomaly Record; assign the user a high Level-of-Interest.
EXPLANATION: Use of a classified password from an unprotected terminal is reason enough to consider the password compromised. The password will be immediately invalidated.

2. Individual Anomaly - NADIR applies these rules to separate user profiles, to detect when a user's behavior departs from that which is normal and valid ICN user behavior. They result from statistical analysis of the past behavior of ICN users, and interviews with security personnel. An example of an individual anomaly rule:

IF the Failure Ratio¹⁶ of a user is >n1,
 AND the user has logged on >n2 and ≤n3 times,
THEN update the Anomaly Record; assign the user a Level-of-Interest.
EXPLANATION: If a user has logged onto the ICN at least n2 times then the user is not new to the ICN. Since the average ICN user has a Failure Ratio that is much less than n1, then a Failure Ratio of n1 is significant. NADIR applies a sliding scale of concern, balanced between the total number of logons and the Failure Ratio, to this rule.

3. Composite Anomaly - NADIR applies these rules to composite user profiles, to detect when that activity departs from that which is normal and valid for the system. They result from statistical analysis of the past behavior of the composite of ICN users. An example of a composite anomaly rule:

IF "Unknown User" errors are >n3/hour, **OR** >n4/day, **OR** >n5/week,
THEN update the Anomaly Record; assign the system a Level-of-Interest.
EXPLANATION: The normal number of attempted authentications that contain a user number that is not valid for the ICN is statistically very consistent. Extreme variations from this expected activity could be a sign of a break-in attempt. NADIR applies a sliding scale of concern to this rule, that depends on the variation from normal.

6.2 Report Rules

These rules do periodic checks of anomalous user activity levels, and define what reports to generate after specific intervals. Designated report intervals may be daily, weekly, or any other period. They analyze the Anomaly Record for the indicated interval, and generate reports that summarize and detail anomalous activity.

6.3 Misuse Indication Rules

NADIR fires these rules when it receives a sequence or combination of Characteristic anomalies that have a low chance of happening. They suggest possible serious misuse of the network. They do

¹⁶ Failure Ratio = $\frac{\text{Invalid_Logons}}{\text{Successful_Logons} + \text{Invalid_Logons}}$

not try to define anything as specific as an attack, but their firing shows something is seriously amiss. The following simplified Misuse Indication rule examines overall ICN user activity:

```
IF the Level-of-Interest for >n6 ICN users is >0,  
  OR the Level-of-Interest for >n7 ICN users is >x,  
  OR the Level-of-Interest for >n8 ICN users is >x + x/2,  
  OR the Level-of-Interest for >n9 ICN users is >2x,  
THEN output an immediate report, that includes an urgent warning message to  
the user interface.  
EXPLANATION: The number of ICN users who reach a particular Level-of-Inter-  
est is statistically very consistent. Extreme variations from the normal  
level of anomalous activity could be a sign of some type of organized mis-  
use of the network. NADIR applies a sliding scale of concern to this rule,  
that depends on the users involved and their Level-of-Interest.
```

The following simplified Misuse Indication rule examines the Anomaly Record of a separate user:

```
IF Characteristic rule 003 is set,  
  (a separate user has many logons this week)  
  AND Characteristic rule 056 is set,  
    (the user has an unusual distribution of logon tries during the swing  
    and weekend shifts).  
  AND Characteristic rule 053 is set,  
    (the user has only unsuccessful ICN logon tries during the night shift).  
  AND Characteristic rule 043 is set,  
    (the user has an unusual distribution of unsuccessful logon tries on the  
    weekend).  
  AND Characteristic rules 040, 041, 044, 045, 046, and 047 are not set,  
    (the user does not show a like pattern of failures during the day shift  
    or on weekdays).  
THEN output an immediate report, that includes a message to the user inter-  
face.  
EXPLANATION: The fired Characteristic rules show a greater than normal  
usage of the ICN, combined with abnormal usage during off hours. Also, the  
user has had an abnormal number of failures during off hours while not  
showing a like pattern of failure during normal working hours. This could  
be a try at masquerading, and is surely suspicious.
```

6.4 Attack Scenario Rules

These rules may define one Characteristic anomaly or Misuse Indication, or a combination of these, that have a low chance of happening. They suggest a known or postulated attack. It is the sequence and combination of these rules that make for an increasing certainty that an attack may be proceeding. Attacks are events that could lead to the compromise or bypass of authentication and access control mechanisms, destruction or compromise of data, or denial of service. Attack Scenario rules are in the definition stage for NADIR.

7 Results

The NADIR working prototype has been in operation since June of 1990. During this time NADIR identified and aided in the investigation of invalid activity by unknown users, and in the investigation of many cases of misuse or suspicious behavior by insiders. It has helped identify unanticipated network vulnerabilities, that have been remedied where possible or are being closely monitored. NADIR development has resulted in the identification of unanticipated misuse conditions, that have led to the definition of new expert rules. It has supported background analyses during investigations of several current and past ICN users. NADIR has also supplied unanticipated net-

work management benefits. It has enabled us to detect hardware and software problems with some nodes of our network. It has also supplied detailed, statistical reports of network activity that were useful in such areas as accounting and network planning.

8 Future Directions

Anomaly and event notice now consists of terminal messages and periodic reports. For serious security events, the ultimate goal is to give notice on a near realtime basis.

Some kinds of invalid user activity, if allowed to continue, could lead to break-ins or denial of service to legitimate users. As a result, another goal is the notification of the proper ICN node of extremely suspicious activity, and the development of effective responses by that node. This would consist of taking direct action to stop an identified penetration attempt. The node's actions must be proportional to the extent that the monitored activity has deviated from valid behavior, what damage could result from allowing an invalid activity to continue, and denial of service considerations. We have not determined the criteria for such a response.

Finally, we would like to identify and use a rigorous method by which to validate and verify the performance, consistency, and completeness of the NADIR expert rule base.

9 Summary

NADIR shows the feasibility of the automation of security auditing on a distributed environment such as the ICN, and the benefits of applying an expert system to the problem. It shows the benefits of a phased approach to applying intrusion detection in a distributed environment. The working prototype is a start to a longer-range goal of expanding the system to more ICN nodes, and correlating their information to produce complete profiles of user activity on the ICN.

10 Acknowledgments

We wish to acknowledge the contributions of Jimmy McClary, who introduced us to the basic ideas, organized our funding, contributed enormously to our expert rule base, and supported us throughout the project. Valuable contributions to our rule base were made by members of the Operational Security Division. We are indebted to Harry Martz for his knowledge of statistics, and to Steve Ruud and Dorothy Merrigan for their contributions to the implementation of the NADIR system.

11 References

- [1] D. Denning and P. Neumann. *Requirements and Model for IDes - A Real-Time Intrusion Detection Expert System, Final Report* (Computer Science Laboratory, SRI International, August 1985).
- [2] M. Freiling, J. Alexander, S. Messick, S. Reh fuss, S. Shulman. *Starting a Knowledge Engineering Project: A Step-by-Step Approach* (The AI Magazine, Fall 1985).
- [3] D. Denning. *An Intrusion Detection Model* (Proceedings of the IEEE Symposium on Security and Privacy, April 1986).
- D. Denning. *An Intrusion Detection Model* (IEEE Transactions on Software Engineering, Vol. 13, No. 2, February 1987).
- [4] D. Denning, D. Edwards, R. Jagannathan, T. Lunt, P. Neumann. *A Prototype IDes: A Real-Time Intrusion Detection Expert System* (Computer Science Laboratory, SRI International, August 1987).

- [5] T. Lunt and R. Jagannathan. *A Prototype Real-Time Intrusion-Detection Expert System* (Proceedings of the IEEE Symposium on Security and Privacy, April 1988).
- [6] T. Lunt, R. Jagannathan, R. Lee, S. Listgarten, D. Edwards, P. Neumann, H. Javitz, A. Valdes. *IDES: The Enhanced Prototype A Real-Time Intrusion Detection Expert System* (SRI International, October 1988).
- [7] M. Sebring, E. Shellhouse, M. Hanna, R. Whitehurst. *Expert Systems in Intrusion Detection: A Case Study* (Proceedings of the 11th National Computer Security Conference, October 1988).
- [8] L. Halme and B. Kahn. *Building a Security Monitor with Adaptive User Work Profiles* (Proceedings of the 11th National Computer Security Conference, October 1988).
- [9] T. Lunt. *Real-Time Intrusion Detection* (Proceedings of COMPCON, Spring 1989).
- [10] T. Lunt, R. Jagannathan, R. Lee, A. Whitehurst. *Knowledge-Based Intrusion Detection* (Proceedings of the 1989 AI Systems in Government Conference, March 1989).
- [11] G. Tsudik and R. Summers. *AudES - An Expert System for Security Auditing* (Proceedings of AAAI Conference on Innovative Applications in AI, May 1990).
- [12] L. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber. *A Network Security Monitor* (Proceedings of the IEEE Symposium on Research in Security and Privacy, May 1990).
- [13] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. Neuman, C. Jalali. *IDES: A Progress Report* (Proceedings of the 6th Annual Computer Security Applications Conference, December 1990).
- [14] J. Winkler. *A UNIX Prototype for Intrusion and Anomaly Detection in Secure Networks* (Proceeding of the 13th National Computer Security Conference, October 1990).
- [15] K. Jackson, D. DuBois, and C. Stallings. *A Phased Approach to Network Intrusion Detection* (Proceedings of the DOE Computer Security Group Conference, May 1991, LA-UR-91-334).

FORMAL VERIFICATION OF

A NETWORK SECURITY DEVICE: A CASE STUDY

Hicham N. Adra
CGI Information Systems
& Management Consultants
275 Slater Street
19th Floor
Ottawa, Ontario, Canada
K1P 5H9
Tel: (613) 234-2155
FAX: (613) 234-6934

William Sandberg-Maitland
CGI Information Systems
& Management Consultants
275 Slater Street
19th Floor
Ottawa, Ontario, Canada
K1P 5H9
Tel: (613) 234-2155
FAX: (613) 234-6934

ABSTRACT

An automated formal verification study of a commercial network security device, the SmartCrypto™, is described. A high level view of relevant formal verification techniques using the m-EVES environment is given. A description of the SmartCrypto™ is provided, as well as a brief overview of the m-EVES system. The uses and roles of Verification plans, environmental and device-specific models, and other planning techniques are discussed in the context of this case. Observations are made concerning the proof process and the problem of tractability which may apply to similar projects.

1.0 INTRODUCTION

This paper completes and extends the work initially reported in [ADRA91]. It describes some aspects of the formal verification of a commercially available Network Security Device (NSD). The NSD under study was the SmartCrypto™ of the CryptoNet™ product line by Intellinet. The study involved a selection of several source code modules, and the development of a formal verification of these target modules against specified properties using the m-EVES environment, described below. The purpose of this study was to establish that basic properties of

functionality and security hold for the NSD design. This account extends that of [ADRA91], promoting the view of formal verification as a software engineering process. The conclusions attempt to summarize and generalize the experience gained. As in [ADRA91], technical details are suppressed.

The paper begins with a presentation of the target NSD system and a brief overview of the verification environment. A brief treatment on the theory of operations of the NSD is given. A more comprehensive treatment can be found in [ADRA91]. A description of the m-EVES verification environment follows. The emphasis is on the user view of the environment, rather than its internal design or technical details. The use of a verification plan is covered, with examples from the project. The role of modelling techniques is an important aspect of formal verification. Examples of modelling drawn from the project are discussed. The paper includes sections on the proof process, some techniques found to be of interest in this domain, and a set of general observations on formal verification issues. The summary draws together some opinions of the authors based on their experience. The role of formal verification within the context of the systems design and development process is highlighted.

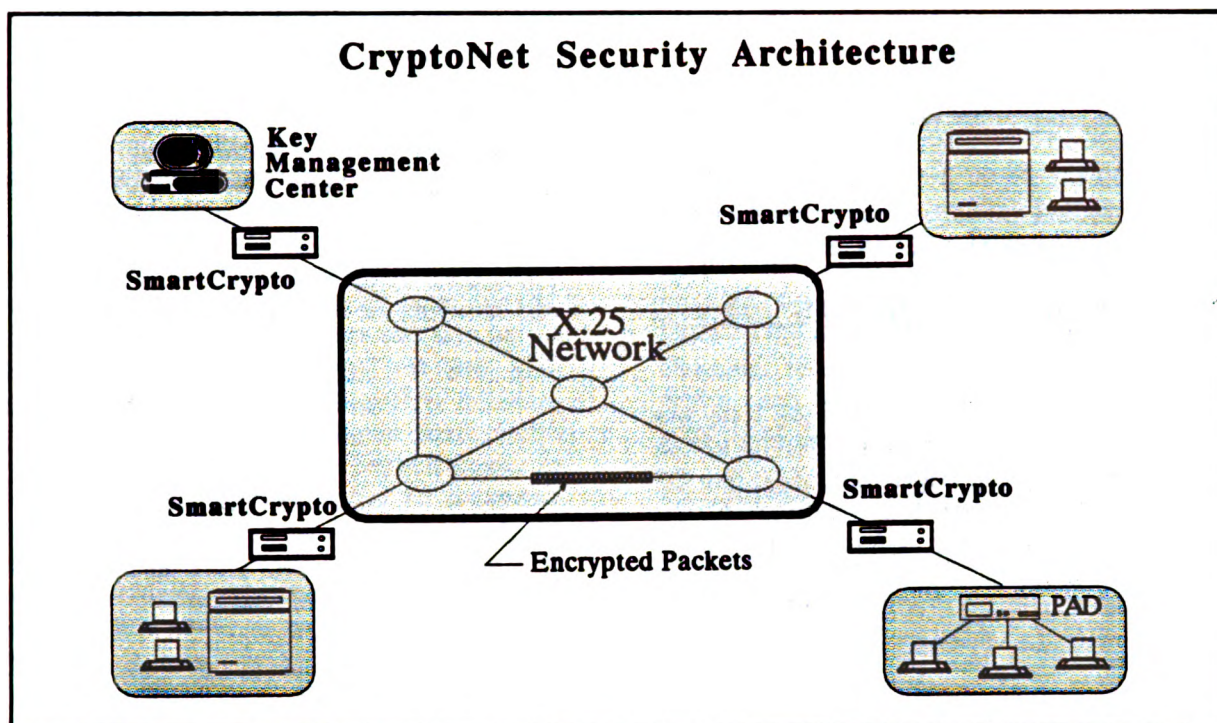


Figure 1: The NSD Environment

2.0 NSD DESCRIPTION

2.1 Theory of Operations

The NSD operates as an end-to-end encryption device functioning in an X.25 packet switching network [X.25]. The DES (Data Encryption Standard) is used in Cipher FeedBack (CFB) mode to achieve confidentiality of the information in the User Data field of X.25 data packets. The NSD is located between the Data Terminal Equipment (DTE) and the Data Circuit-terminating Equipment (DCE) or network. The NSD filters the traffic between the "host" and the network (see Figure 1). The host may be a computer system or a collection of terminals that are connected to an X.25 PAD (Packet Assembler/Disassembler). The Key Management Center (KMC) is responsible for the management of the network, including the distribution of keys and the remote monitoring and control of the network sites.

Several internal states are supported, including a Secure Normal State where encryption/decryption processing is performed on all data packet traffic between protected hosts. Since the control or header information is transmitted in its plaintext

form and remains accessible to the intermediate nodes in the network, this method of applying encryption is called end-to-end; the nodes within the network do not need to be trusted to protect the security or secrecy of the information, as discussed in Section 4.

2.3 Security Protocol and Communication Requirements of NSD

Terminology related to CCITT Recommendation X.25 [X.25] will not be defined in detail here. Some discussion of X.25 issues is necessary to form the context for the inter-relation between security and communications functionality.

The NSD operates at the Network layer of the OSI model. Within the X.25 packet level DTE/DCE (interface (Level 3) frame of reference, the NSD acts as a data filter which intercepts data packets in either the DCE to DTE, or the DTE to DCE direction, and transforms them by decryption or encryption of user data fields. Most X.25 control packets are also recognised.

A Security Protocol is followed when a DTE call request packet initiates the setting up of a call. The NSD ensures that certain requirements are

met; the security protocol relates to such elements of the X.25 call as the addresses, the logical channel numbers, the NSD network security groups, the encryption variables, and a security checksum.

Similar functions are performed in the case where the call is incoming from the DCE.

3.0 VERIFICATION TOOLS: m-EVES SYSTEM

The principal software tool used in this study was the m-EVES version 4 formal verification system running on a Sun 3/80 workstation under Sun OS version 4.0.3. m-EVES (an Environment for Verifying and Evaluating Software), of Odyssey Research Associates, is a prototype formal verification system [CRAI88] [EVES89]. m-EVES contains two main components:

- m-Verdi, a specification and implementation language,
- m - N E V E R , a n interactive/automated theorem prover.

An automated style of proof is possible with certain high level commands which invoke proof heuristics. m-EVES maintains an internal database of proven theorems. m-EVES has a soundness proof for its logic [EVES89]. For a more complete description of the m-EVES environment, see [CRAI88].

3.1 m-EVES Proof and Verification

This section defines some verification terminology referred to in the rest of the paper. A more detailed view of these topics will be found in [EVES89].

The term 'formula' will refer to a first order logical expression which is obtained from entering an m-Verdi target text in an m-EVES session. Informally, m-EVES reliably translates m-Verdi text into an equivalent and purely logical format which contains no occurrences of commands or other algorithmic language constructs. The resulting (initial) formula may be transformed into other formulas, using EVES Command Language (ECL) commands. Two particular formulas are of

note: TRUE, which designates the universally valid formula, and FALSE, its negation.

Informally, a proof in m-EVES is a successful attempt to reduce the initial formula for a given m-Verdi target text to TRUE, through a finite number of steps. This generates a sequence of formulas, each formula derivable from its predecessor by application of an ECL command, and satisfying the following conditions:

- the first formula is obtained by m-EVES from the m-Verdi target,
- the final formula is TRUE.

The terms 'formula' and 'proof' are used in this paper exclusively in the sense given above.

3.2 Application of m-EVES to the NSD Verification Study

The NSD study involved a code verification of sample modules of the NSD system. As both specifications and implementation were known, a method was required to translate both into m-Verdi in the most reliable way. The implementation source code for the NSD is in the C language (with some assembler code). A number of hardware components, such as the encryption chip, also needed representation. Fortunately, the translation of target C code to m-Verdi was a relatively efficient manual operation. Some other software may not be as easily translatable, however, due to the use of pointers in C code which have no built-in support in m-Verdi. Hardware and environmental elements were translated by modelling techniques discussed below, and by the use of the m-Verdi "environment" construct.

A standard theory of history sequences is easily implemented in m-Verdi from examples in the literature [CRAI88]. The NSD study required some form of discrete temporal reasoning to prove that basic liveness properties hold. A simple history sequence theory is needed for this. Any system whose specifications include time dependencies between events will likely need a similar model. It was observed that certain theories expressed in terms of temporal logic have a natural embedding into first-order m-Verdi theories involving history sequences (see [FVR]).

This could be exploited in many general contexts.

Practicality may necessitate controlled modifications to the m-Verdi code. An example from this study illustrates this point. One of the target modules had interfaces with a large number of lower level sub-modules, each of which had extensive logical structure involving low level variables lying outside the general context of the verification module. In order to minimize the impact that such a code structure can have on the proof of high level structures, the low level modules were stubbed. This involves declaring them in m-Verdi without code, but possibly with logical annotations which describe their action. In addition, an array of flags, called an *occurrence array*, was defined. If a module is invoked, the boolean flag pertaining to it is set in the occurrence array. This can be done through specifying postconditions on the stubs.

Using this technique, it is possible to express general code-oriented specifications that say that under specific conditions, certain modules should be invoked. Proving this kind of specification provides assurance that the right sub-modules were called under various sets of conditions, and avoids the difficulty of contriving equivalent expressions employing low level variables. It is possible to return to the proof later and integrate the low level modules into the existing proof in a top-down manner, or employ some independent method to verify them. The use of in-line annotations, supported in m-Verdi as the "note" command, is also applicable to this problem.

4.0 VERIFICATION PLAN

4.1 Purpose of the Verification Plan

The role of the verification plan was to describe the scope of the verification effort in terms of an identification of general properties or areas of functionality within the NSD that were considered to be suitable objects of investigation in the next phase(s) of the project. The verification plan followed a phase where the architecture of the NSD, including the theory of operation, the hardware and software structure, and the functional requirements were analyzed. The next phase of the project was expected to involve the development of a design-level description of the verification targets, or modules to be specified,

verified, and implemented in the context of the m-EVES environment.

The main focus in the verification plan was on the selection of a subset of the NSD for the purposes of formal verification and on the identification of some of the main issues that characterized the technical concerns at that stage. The criteria that guided the selection of verification targets included: the need to define the scope of effort based on the available resources, the importance of choosing important/critical properties of the device, and the characteristics of the formal verification environment and process.

4.2 Selection of Verification Properties

The selection of properties for formal verification is of critical importance to the value and level of achievement of the project as a whole. If the properties chosen reflect an overly simplistic or trivial view of the system, little is achieved in formally verifying them. If, on the other hand, the properties are either complex or inconsistent, the prospects for obtaining positive results become minimal or nonexistent. In the interest of obtaining useful results from the verification process, a practical balance between meaningful system properties and provable verification goals was the prime motivation for this task.

The criteria that were used for selection of the functionality that would be addressed can be broadly expressed as the following two areas:

- Security properties and
- Basic Functionality properties.

It is worth recalling that the NSD implements end-to-end encryption in an X.25 network [X.25]. It acts as a "filter" and is situated between the DTE or host and the DCE or network.

The properties selected and some necessary assumptions regarding the system are described in the following sections.

4.2.1 High Level Network Security Properties

The high level security properties of the NSD are informally based on the main aspects of information security. In this section these properties are discussed briefly and related to the

major components of information security.

Security is generally considered to encompass the following three areas: Confidentiality, Integrity, and Availability. Threats to confidentiality relate to the unauthorized disclosure of information. Integrity refers to the properties through which the system or information meets one's expectations. Availability can be viewed in terms of the manifestation of its absence in the form of denial of service.

For the NSD, confidentiality is achieved through encryption. If encryption of data can be verified where it is required, then the confidentiality of data in the network is guaranteed. The property is largely dependent on the NSD processor which sends the data to the network. The assumption is made that no decryption activity can occur other than within another NSD. It is assumed that a DES-encrypted data packet cannot be read unless the key and initialization vector are known. With these assumptions, confidentiality is largely a byproduct of the basic CSP (see [HOAR85]) specifications that were developed during the Architecture Review stage. The specifications enforce rules regarding the conditions under which a data packet is encrypted. Confidentiality is compromised only if a data packet is not encrypted according to these rules.

While encryption contributes to a limited form of integrity, the data exchanged over the network can be manipulated and additional measures (at a higher communications layer) may be required to protect against threats to integrity. The use of distinct initialization vectors for each direction of an X.25 call aids in the detection of reverse direction replays. Also, the NSD supports a method for the authentication of the identities of the communicating parties. Through the logical design of the network and the ownership of the keys, an implicit form of authentication is achieved.

4.2.2 Basic Functionality Properties

The basic functionality of the NSD concerns properties relating to its role as a type of data encryption filter in a X.25 network, as well as the internal features which support this role, in particular, its security protocol. The term "basic functionality" is intended to describe the major NSD documented specifications around which the

system is designed. In some cases, problems resulting from conflicts between hardware, communications and security requirements resulted in non-trivial modifications to the network layer behaviour of the NSD. It was important to determine that the NSD implementation actually satisfies these requirements.

The basic functionality of the NSD was primarily expressed in the formal specifications. These were written in CSP and required reliable translation to m-Verdi. This immediately implies that the underlying models (and their m-Verdi theories) must be compatible with whatever form the translations of the CSP functionality requirements take. A theory in m-Verdi which adequately describes the input-output black box view of the NSD must therefore mimic the behaviour specified in the CSP formal specifications.

4.2.2.1 X.25 Protocol Properties

The basic X.25 protocol is embedded in the CSP formal specification of the NSD. It was observed during the architecture analysis that not all internal X.25 states are implemented in the NSD, and that some events are not treated in the expected way. The specification took much of this into account, although the Call Collision State is present in this specification, but is not implemented in the NSD (since the encryption process does not allow for this). It was recognized that some modification of the specifications may therefore be in order prior to the verification activities.

Proving that the NSD satisfied a modified subset of X.25 was one of the primary objectives of the verification tasks.

4.2.2.2 NSD Security Protocol Properties

The security protocol for the NSD is embedded in the lower invocation levels of the CSP formal specification, i.e. in the form of special processes which are triggered when certain security-sensitive events occur. The main areas include Call Initialization, encryption-related events, and handling the interaction with the KMC. The last area was not considered to be within the scope of the verification plan.

Important high level properties are based on the sufficiency of the NSD security protocol. However,

the strategy envisioned in the verification plan was not to establish the security protocol properties and then prove the high level properties. Rather, the NSD security protocol was seen as an inseparable part of the formal specification of the NSD. Its verification was seen as part and parcel of the verification of the basic functionality of the system. It would be possible in any case to draw on specific security properties obtained in the basic functionality verification in the establishment of high level security properties.

5.0 ROLE OF DEVICE-SPECIFIC MODELLING TECHNIQUES

Given a device with the level of complexity of the NSD, special models are often required to form a framework for stating certain specifications. Typically these models portray or simulate an environmental factor which the system must tolerate, a special theory or recognized standard which the system must satisfy, or possibly a subsystem or external entity whose characteristics are assumed and whose verification is considered beyond the scope of the project.

Implementation of a model in the verification language (e.g., m-Verdi) involves a design phase where decisions are made regarding the type and number of variables and data structures required. Some procedures may be designed and coded. In addition, a model will normally require purely logical components such as axioms and specification functions. A prototyping phase may be called for, in order to resolve design decisions. To finalize the initial model-building phase, theorems involving model constructs and relating them to the appropriate system interfaces are developed. Again, this may entail prototyping, as new model features may arise out of the attempt to prove the target theorems. In this way the body of theory involving the model is built up to the required level of depth.

Experience gained in this research indicates that even very simple models can entail significant costs in terms of time and effort over the verification phase. The effect of incrementally adding new models to a stable (i.e. proven) body of modules introduces the obligation to integrate all new variables and data structures into the old module proofs, and thus multiply their length. As more models are integrated in this way, the effect

appears to be significantly non-linear. Although too little quantitative evidence is available at this point, this growth effect may have a significant influence on the design and scope of verification projects similar to the one documented here.

The following three main models were required by the verification phase:

- **Non-determinism model**
A model which allows proof of certain fault tolerance and liveness properties of the NSD under uncertainty of success of certain packet processing tasks.
- **X.25 model**
A decision tree model of the state transitions of the X.25 protocol.
- **Encryption model**
An elementary DES encryption (CFB) model based on axioms obtained from [FIPS81].

In each case, challenges were encountered with the integration of the new model into the existing proof database.

6.0 THE PROOF PROCESS

6.1 Verification of an Existing System

The nature of this project, which has its basis in an existing system, does not lend itself to the traditional top-down approach. In a general verification project one may have control over models of both the specification and the implementation and the verification can involve the development of parallel or corresponding descriptions. Attempting to gain assurance about a system after it has been developed through formal verification entails great difficulties. Verification of a low level of specification (eg. the source code) against a higher level of abstraction (eg. specifications of the requirements) is almost impossible without intermediate levels of detail. Models that characterize the specifications of system behaviour and the implementation must be developed. The specifications and the implementation must share various correspondences that relate to their logical structure and to their semantical content. The

process of verification entails building the specification, the proof, and the implementation in tandem. For large systems this process is not easily managed, especially when an existing system is being examined. The implications for the certification of systems are serious.

6.2 Interaction with the Prover

Although provers such as that of the m-EVES environment are called automated provers, it should be remembered that they function as proof checkers and interactively assist in the proof process. The developer or user should ensure that the module or software being verified has been designed and implemented such that a proof would be forthcoming, and that the necessary conditions are met. The user will issue commands to effect certain proof steps, including the application of types of heuristics that the prover supports and the incorporation in the current proof of other theorems, lemmas, or assumptions. The user has to read the output of the prover at each step of the proof and be able to determine what parts of the current formula are of interest. The ability to see where conditions need to be strengthened or inconsistencies addressed is central to the process.

6.3 Modules and Preconditions

The verification of a module will show that if it is invoked when its precondition is satisfied it will terminate and its postcondition will be true. Even when the requisite proof is completed there remains the obligation to show that the precondition is satisfied in the calling module. Depending on the structure of the system and the time relationships between variables it may become very difficult to reason about the dependencies and to ensure the consistency of the various conditions when changes are made. When modules do not have side-effects and their preconditions are very simple this difficulty is reduced.

6.4 Use of Small Steps and Automation

The m-EVES prover has a number of 'macro' commands that may apply several basic or simple steps. These macro commands effect highly automated manipulations of a formula in an attempt to show that the condition it embodies is true. While it is desirable and sometimes easier to invoke these powerful commands to arrive at a proof, in cases where the formula is a complex or

long logical condition the highly automated capabilities of the prover were not found to be very effective. The time required for a powerful command to execute becomes too long and the prover cannot be guaranteed to find a proof. The interactive application of a larger number of smaller steps was found to be more productive and allowed the developer greater flexibility in finding a proof. Although this required closer examination of the formula being verified at each step of the proof and was a very demanding process the ability to gradually simplify the formula and the higher likelihood of arriving at a proof made such a strategy necessary. Such a strategy is especially needed when an existing system is being studied since the verification team has less control over the software structure and design and the proof has to be adapted to the general architecture of the system.

6.5 Iterative Flow of Activities

The general view of formal verification as a top-down process of defining specifications and then implementing the software that demonstrably satisfies the specifications as evidenced by the proof is an abstraction in search of a reality. Formal verification necessitates a close match between the specifications and the implementation and between various parts of each. Changes to any part of the system descriptions may require changes to other parts depending on the dependencies that exist. Since it is unlikely that initial descriptions will be complete, changes and extensions must result. In an automated environment a considerable amount of the formal descriptions or theories are developed to support the verification effort or in order for the prover to deal with leaps of abstraction and are not strictly part of the system. Thus the likelihood of the discovery of the need for additional assertions, properties, and relationships is very high, and semantical changes often require substantial concern with aspects of the verification environment and language. The result is a process that defies simplistic depictions and which requires both anticipation of what is needed and the ability to recognize that changes will be necessary. A developer should expect a considerable amount of both planning and refinement.

6.6 Gradual Building of Proof Requirements

Despite the high penalty for changes to a system's

description (specification or implementation), in some cases it is useful to experiment with a module to arrive at the proper form and conditions. In such cases the weaker or simpler forms of the (post) condition that must be shown may be used to gain confidence in the correctness of the module's code or structure and to quickly discover any flaws, which would be easier to detect since any inconsistencies will be more apparent in the simpler formula. The general strategy that was used in the proof may also be applicable to the stronger or final condition.

7.0 OBSERVATIONS AND FINDINGS

7.1 Meaning of Verification Results

The mathematical nature of formal methods and the benefits of (automated or computer assisted) formal verification do not obviate the need for a critical assessment of the level of assurance provided by the verification effort. The system descriptions and documentation -- in the form of such constructs as axioms, data declarations, executable code, and theorems -- may encompass several assumptions and models, the appropriateness or validity of which cannot be determined solely within the steps and proofs of the formal verification effort or environment.

The formal verification results may not guarantee the absence of inconsistencies in the system specifications. In addition, the strength and completeness of the assertions that are shown is central to the value of the verification effort. The verification team is free to choose the form of a module and the statements of the precondition and postcondition. The assurance, about the behaviour of a module, that is provided by a proof (based on the precondition and postcondition) is not always clear, especially with respect to intermediate occurrences of conditions and states. The role of the results of an individual proof must be carefully considered in relation to other proofs and within the general description of the system. The results of the verification effort as a whole, in turn, must be assessed with a recognition of any assumptions and limitations that exist and to determine the implications for the behaviour and trustworthiness of the system.

7.2 Engineering Side of Formal Verification

In addition to the mathematical nature of formal verification, the development of formal and executable specifications and descriptions involves many of the choices and decisions that characterize engineering processes. Many of the engineering issues do not manifest themselves in the formal verification of a simple or small application, partly because the implications of the decisions may not be critical to the success of the software effort. When large and complex systems are being built, however, the number and difficulty of the choices that the project team faces are increased. The quantitative aspects may become qualitative in that gradual accumulation of complexity may represent unsurmountable obstacles to the successful completion of the project. The ability to operate within an integrated project support environment is expected to form a key requirement of formal specification and verification tools.

While most likely there is no general recipe for building systems using formal methods and automated verification environments, there are various approaches that are effective in dealing with the complexity that faces software designers and developers. Although it is beyond the scope of this paper to describe design or development methods, some of the observations in this section may hint at some properties that such methods should have.

7.3 Relative Size of the Formal Descriptions

In specifying and implementing modules in m-Verdi it was observed that the size of the resulting software was considerably larger than the original C source code. This may reflect a basic difference between software development that uses third generation languages and that which is based on formal verification. In the latter case, some of the information that would traditionally reside in the various requirements and design documents has to be represented in the formal specifications that are developed within the verification environment. Also, formal verification may require or be facilitated by the specification of supporting models and theories that capture the functionality of the software and bridge the gaps between different levels of abstraction. The general observation was that the size of the formal product far exceeded the C language source code. The use of graphical depiction techniques may aid in

addressing the difficulties associated with the long expressions and formulas; graphical nested structures may be presented to the analyst in order to increase the communications bandwidth of the user interface between the environment and the user [TAR]. However, the implications for a large project of the large size of formal descriptions are very serious when the effect of software size on the required effort and schedule, as discussed below, is recognized.

7.4 Effect of System Growth on Schedule and Effort

The relationship between software size and the effort or time required to develop that software is considered to be non-linear, and in fact many estimation models represent effort as an exponential function of size. In this project, the addition of more functionality and the attempt to integrate these modules with the existing system descriptions required considerable effort. The nature of formal verification as an effort and time intensive process and the need for ensuring proof consistency indicate an even steeper form of the curve for effort as a function of system size. The consequences for the direct application of formal verification to large projects are serious and seem to entail considerable limitations.

7.5 Intermediate Levels of Abstraction

While the step-wise refinement of system descriptions is a generally known technique for design and development, the use of specifications at different levels of abstraction has special implications for formal methods and for attempts to achieve a high level of assurance as required in secure systems. The use of a several levels of abstraction facilitates the mapping between levels. It also increases the effort required to manage the system descriptions and may become less effective when too many levels are used. In formal verification efforts the project team may find the development of intermediate specifications (or implementation layers) to be necessary in interacting with an automated prover which can not be expected to deal with wide abstraction gaps. The recognition of: the need to maintain the consistency of the software descriptions, performance considerations, and the ripple effect of changes to one module on other parts suggests a trade-off between these factors and the number of levels in the abstraction hierarchy. The careful

introduction of intermediate layers remains an effective strategy for simplifying formal proofs.

7.6 Propagation of Properties to Higher Levels of Abstraction

As modules at the lower levels of the module hierarchy, including those that have a relatively self-contained function, are developed and integrated with modules at a higher level their properties need to be reflected in the properties of the upper layers. This process of making the function of lower-level modules known to a higher module involves the upward migration of properties within the hierarchical structure of the software. The preconditions and postconditions at adjoining layers need to be closely linked. Even after achieving the proper correspondence between the modules and their properties, there is a strong likelihood that changes to a lower level will be necessary and that a large part of the time that was expended in the verification of the existing modules will be required again.

7.7 Understanding the Automated Prover Output

In interacting with an automated prover, the software developer or user is faced with several characteristics of that tool and environment. One such aspect is the length and complexity of verification conditions that are generated by the prover. The automated capability of the prover includes the generation of a formula that formally describes the module and its corresponding proof obligation and the execution of user commands that represent steps in verifying that the formula is true. The formulas may be long and complex. The developer has to read the formula and determine what steps are necessary for its proof or whether a proof can be found at all; the formula may be amenable to proof with the right sequence of commands, or it may be inconsistent or lacking some necessary assertions in which case it cannot be proven.

Thus, although the prover automates certain capabilities the user person has to interact with it and is expected to 'process' its output. This seems to tie the person to the technological tool. It is not clear to what extent expecting a developer to read the long and intricate product of a tool (at every intermediate step) represents the best form of an activity's automation. This may be one stage in the evolution of verification technology and

further advances in software engineering may be expected to alter this cooperative process.

7.8 Focus on Critical Functionality

For several reasons that include the observed growth effect in terms of the relationship between the size of the system descriptions and the required effort, as described earlier, the choice of the appropriate scope for formal verification is considered to be essential to the success of a formal verification project. It is important to limit the targets of formal verification to the "critical" functionality or components of the system under consideration. While determining the critical nature of a component is a matter of judgement and may be based on the area of interest or depend on the design of the system, the choice of the proper scope for formal verification is necessary to the management of the complexity of the verification process.

8.0 SUMMARY

This paper has identified several aspects of formal verification as applied to a network security device. The use of certain modelling techniques was described in the context of the goals of the verification effort. The role of a verification plan within the life cycle of a formal verification project was shown through a case study approach to the communication of findings. The results and observations described in this paper, and in Sections 6 and 7 in particular, were part of this attempt to share the project team's general experience in the application of formal verification to a target having substantial scope of functionality.

In the verification of the NSD, the general areas that were addressed included, in addition to the general X.25 functionality, both safety and liveness properties. Safety properties within the context of the NSD were examined in terms of the controlled application of encryption. The use of a model of nondeterminism to support liveness stemmed from a choice to characterize the rich behaviour of a communications system and to avoid extreme simplifications. While it is recognized that abstraction is an essential part of the use of formal methods and is often necessary in describing systems, the need remains for formal verification efforts to address the complex or flexible behaviour

that is inherent in some systems.

The logical separation between "security" and "functionality" or "liveness" is sometimes detrimental to the evaluation of a system. (Among other limitations: it seems to reflect a bias towards viewing security as secrecy or confidentiality. Furthermore, the assumption that properties of a system are independent or even that different areas of security can be examined in isolation is not justified.) While a system that does nothing can be considered safe with regards to confidentiality its trustworthiness is of very little value. It is in the area of complex systems and rich behaviour that security and trust are of special interest and importance.

The relationship between the general functionality of a system and its security policy, especially in their implementations, often involves many subtle dependencies and provides a challenge in any attempt to gain the high degree of assurance that is necessary for the system to be considered trustworthy. The formal proof process and the use of an automated environment contribute to both the challenges and the solutions. The tractability of the verification problem, as manifested by the effort and time required, is a serious concern. It is hoped that this paper will contribute to the recognition of the role of formal verification and specification within the systems engineering process and of the need for verification design methods and techniques that support the management of the complexity of the formal development process. Towards such a view, it is important to recognize the role, limitations, and benefits of formal verification within an integrated systems design and development process.

ACKNOWLEDGEMENTS

The work described in this paper was funded by the Canadian Department of National Defence under contract No. W8477-9-CC07/01-QD. The support and insights of Mr. Milan Kuchta and Mr. Vincent Taylor are gratefully acknowledged. This paper has also benefited from the comments and suggestions of the referees. The authors would also like to thank Intellinet Corp. for their help and for providing access to the CryptoNetTM/SmartCryptoTM technical documentation.

REFERENCES

- [ADRA91] Adra, H.N. and Sandberg-Maitland, W., "Formal Verification Techniques for a Network Security Device", Proceedings of the Third Annual Canadian Computer Security Symposium, May 15-17, 1991.
- [CRAI88] Craigen, Dan, "An Application of the m-EVES Verification System", Proceedings of the Second Workshop on Software Testing, Verification, and Analysis", IEEE, July 1988, pp. 21-36.
- [EVES89] *m-EVES Collected Papers*, Odyssey Research Associates, Inc., Ottawa, Ontario, Canada, September 14, 1990.
- [FIPS81] U.S. Department of Commerce/National Bureau of Standards, FIPS Publication 74, *Guidelines for Implementing and Using the NBS Data Encryption Standard*, April, 1981.
- [FVR] *Verification of a Network Security Device: Final Verification Report (FVR)*, Technical Report SE-R91.009, CGI Group, Ottawa, Ontario, Canada, March 1991.
- [HOAR85] Hoare, C.A.R., *Communicating Sequential Processes*, Prentice-Hall International, 1985.
- [INT87] *CRYPTONET / SMART CRYPTO Theory of Operation and Functional Specification*, Technical Report INT-87-37, Intellitech Canada Limited, Ottawa, November 12, 1987.
- [TAR] *Verification of a Network Security Device: Technical Assessment Report (TAR)*, Technical Report SE-R91.008, CGI Group, Ottawa, Ontario, Canada, March 1991.
- [VPR] *Verification of a Network Security Device: Verification Plan and Rationale (VPR)*, Technical Report SE-R90.29, CGI Group, Ottawa, Ontario, Canada, May 1990.
- [VR1] *Verification of a Network Security Device: Verification Report 1 (VR1)*, Technical Report SE-R90.43, CGI Group, Ottawa, Ontario, Canada, September 1990.
- [X.25] *Interface between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE) for Terminals Operating in the Packet Mode and Connected to Public Data Networks by Dedicated Circuit*, CCITT Recommendation X.25, 1984.

A FRAMEWORK FOR ADVANCING INTEGRITY STANDARDIZATION

Terry Mayfield
Stephen R. Welke
John M. Boone
Catherine W. McDonald

Institute for Defense Analyses
1801 N. Beauregard St.
Alexandria, VA 22311
(703) 845-3500

Abstract

This paper deals with the issue of preserving and promoting integrity within computer and automated information systems. It is intended to serve as the starting point for defining those expectations and standardizing integrity properties of systems. The paper discusses the difficulty of developing a single definition of the term integrity as it applies to data and systems. Integrity has multiple definitions in the dictionary and the application of those definitions to data and systems using a single attribute within the expectation set has led to definitions that could not achieve consensus. Concluding that a single definition is not needed to advance our understanding, the paper develops a more appropriate operational definition, or framework, that encompasses various views of the issue. This framework includes the two distinct, yet interdependent, contexts for integrity: data and systems. The framework reinterprets, within these two contexts, a general integrity protection goal to derive three specific integrity goals. The framework also interprets the integrity properties and relationships of active and passive entities in a system using the conceptual constraints of "adherence to a code of behavior," "wholeness," and "risk reduction." The paper concludes that it is possible to begin to standardize integrity properties. We acknowledge that gaps in understanding exist, but recommend that further studies be undertaken. We conclude that such studies can be accomplished concurrently with standardization and that both efforts could be mutually supportive.

1. Introduction

As public, private, and defense sectors of our society have become increasingly dependent on widely used interconnected computers for carrying out critical as well as more mundane tasks, integrity of these systems and their data has become a significant concern. The purpose of this paper is not to motivate people to recognize the need for integrity, but rather to motivate the use of what we know about integrity and to stimulate more interest in research to standardize integrity properties of systems. This paper provides a framework for examining the issue of promoting and preserving integrity in computer systems. It is intended to be used as a general foundation for further investigations into integrity and a focus for debate on those aspects of integrity related to computer and automated information systems (AISs).

One of the specific further investigations is the development and evolution of product evaluation criteria to assist the U.S. Government in the acquisition of systems that incorporate integrity preserving mechanisms. These criteria also will help guide computer system vendors in

The work reported in this paper was conducted as part of Institute for Defense Analyses Project T-AA5-459 under Contract No. MDA903-89-C-0003 for the Department of Defense. It is based on portions of IDA Paper P-2316, *Integrity in Computer and Automated Information Systems*, which is in preparation at this time. The publication of this paper does not indicate endorsement by the Department of Defense or the Institute for Defense Analyses, nor should the contents be construed as reflecting the official positions of those organizations.

producing systems that can be evaluated in terms of protection features and assurance measures needed to ascertain a degree of trust in the product's ability to promote and preserve system and data integrity. In support of this criteria investigation, we have provided a separate document [1] that offers potential modifications to the Control Objectives contained in the Trusted Computer Systems Evaluation Criteria (TCSEC), DoD 5200.28-STD [2]. The modifications extend the statements of the control objectives to encompass data and systems integrity; specific criteria remain as future work.

2. Background

For some time, both integrity and confidentiality have been regarded as inherent parts of information security (INFOSEC). Confidentiality, however, has been addressed in greater detail than integrity by evaluation criteria such as the TCSEC. The emphasis on confidentiality has resulted in a significant effort at standardizing confidentiality properties of systems, without an equivalent effort on integrity. However, this lack of standardization effort does not mean that there is a complete lack of mechanisms for or understanding of integrity in computing systems. A modicum of both exists. Indeed, many well-understood protection mechanisms initially designed to preserve integrity have been adopted as standards for preserving confidentiality. What has not been accomplished is the coherent articulation of requirements and implementation specifications so that integrity property standardization can evolve. There is a need now to put a significant effort on standardizing integrity properties of systems. This paper provides a starting point.

The original impetus for this paper derives from an examination of computer security requirements for military *tactical* and *embedded* computer systems, during which the need for integrity criteria for military systems became apparent. As the military has grown dependent on complex, highly interconnected computer systems, issues of integrity have become increasingly important. In many cases, the risks related to disclosure of information, particularly volatile information which is to be used as soon as it is issued, may be small. On the

other hand, if this information is modified between the time it is originated and the time it is used (e.g., weapons actions based upon it are initiated), the modified information may cause desired actions to result in failure (e.g., missiles on the wrong target). When one considers the potential loss or damage to lives, equipment, or military operations that could result when the integrity of a military computer system is violated, it becomes more apparent why the integrity of military computer systems can be seen to be at least as important as confidentiality.

There are many systems in which integrity may be deemed more important than confidentiality (e.g., educational record systems, flight-reservation systems, medical records systems, financial systems, insurance systems, personnel systems). While it is important in many cases that the confidentiality of information in these types of systems be preserved, it is of crucial importance that this information not be tampered with or modified in unauthorized ways. It is especially important that unauthorized tampering not occur in embedded computer systems. These systems are components incorporated to perform one or more specific (usually control) functions within a larger system. They present a more unique aspect of the importance of integrity as they often may have little or no human interface to aid in providing for correct systems operation. Embedded computer systems are not restricted to military weapons systems. Commercial examples include anti-lock braking systems, aircraft avionics, automated milling machines, radiology imaging equipment, and robotic actuator control systems.

Integrity can be viewed not only in the context of relative importance but also in the historical context of developing protection mechanisms within computer systems. Many protection mechanisms were developed originally to preserve integrity. Only later were they recognized to be equally applicable to preserving confidentiality. One of the earliest concerns in the development of computers was that programs might be able to access memory (either primary memory or secondary memory such as disks) that was not allocated to them. As soon as systems began to allocate resources to more than one program at a time (e.g., multitasking, multiprogramming, and time-

sharing), it became necessary to protect the resources allocated to the concurrent execution of routines from accidentally modifying one another. This increased system concurrency led to a form of interleaved sharing of the processor using two or more processor states (e.g., one for problem or user state and a second for control or system state), as well as interrupt, privilege, and protected address spaces implemented in hardware and software. These “mechanisms” became the early foundations for “trusted” systems, even though they generally began with the intent of protecting against errors in programs rather than protecting against malicious actions. The mechanisms were aids to help programmers debug their programs and to protect them from their own coding errors. Since these mechanisms were designed to protect against accidents, by themselves or without extensions they offer little protection against malicious attacks.

3. Defining Integrity

Integrity is a term that does not have an agreed definition or set of definitions for use within the INFOSEC community. Recent efforts to define and model integrity have raised the importance of addressing integrity issues and the incompleteness of the TCSEC with respect to integrity. They also have sparked renewed interest in examining what needs to be done to achieve integrity property standardization in computing systems. However, the INFOSEC community’s experience to date in trying to define integrity provides ample evidence that it doesn’t seem to be profitable to continue to try and force a single consensus definition. Thus, we elect not to debate the merits of one proposed definition over another. Rather, we accept that the definitions generally all point to a single concept termed “integrity.”

Our position is reinforced when we refer to a dictionary; integrity has multiple definitions [3]. Integrity is an abstract noun. As with any abstract noun, integrity derives more concrete meaning from the term(s) to which it is attributed and from the relations of these terms to one another. In this case, we attribute integrity to two separate, although interdependent, terms (i.e., *data* and *systems*). Bonyun made a similar observation in discussing the difficulty of arriving

at a consensus definition of integrity [4]. He also recognized the interdependence of the terms systems and data in defining integrity, and submitted the proposition that “in order to provide any measure of assurance that the integrity of data is preserved, the integrity of the system, as a whole, must be considered.”

Keeping this proposition in mind, we develop a conceptual framework or operational definition which is largely derived from the mainstream writing on the topic and which we believe provides a clearer focus for this body of information. We start by defining two distinct contexts of integrity in computing systems: *data integrity*, which concerns the objects being processed, and *systems integrity*, which concerns the behavior of the computing system in its environment. We then relate these two contexts to a general integrity goal developed from writings on information protection. We reinterpret this general goal into several specific integrity goals. Finally, we establish three conceptual constraints that are important to the discussion of the preservation and promotion of integrity. These definitions, specific goals, and conceptual constraints provide our framework or operational definition of integrity. A diagram of this framework is given in Figure 1.

3.1. Data Integrity

Data integrity is what first comes to mind when most people speak of integrity in computer systems. To many, it implies attributes of data such as quality, correctness, authenticity, timeliness, accuracy, and precision. Data integrity is concerned with preserving the meaning of information, with preserving the completeness and consistency of its representations within the system, and with its correspondence to its representations external to the system. It involves the successful and correct operation of both computer hardware and software with respect to data and, where applicable, the correct operations of the users of the computing system (e.g., data entry). Data integrity is a primary concern in AISs that process more than one distinct type of data using the same equipment, or that share more than one distinct group of users. It is of concern in large scale, distributed, and networked processing systems because of the diversity and interaction of information with which such systems must

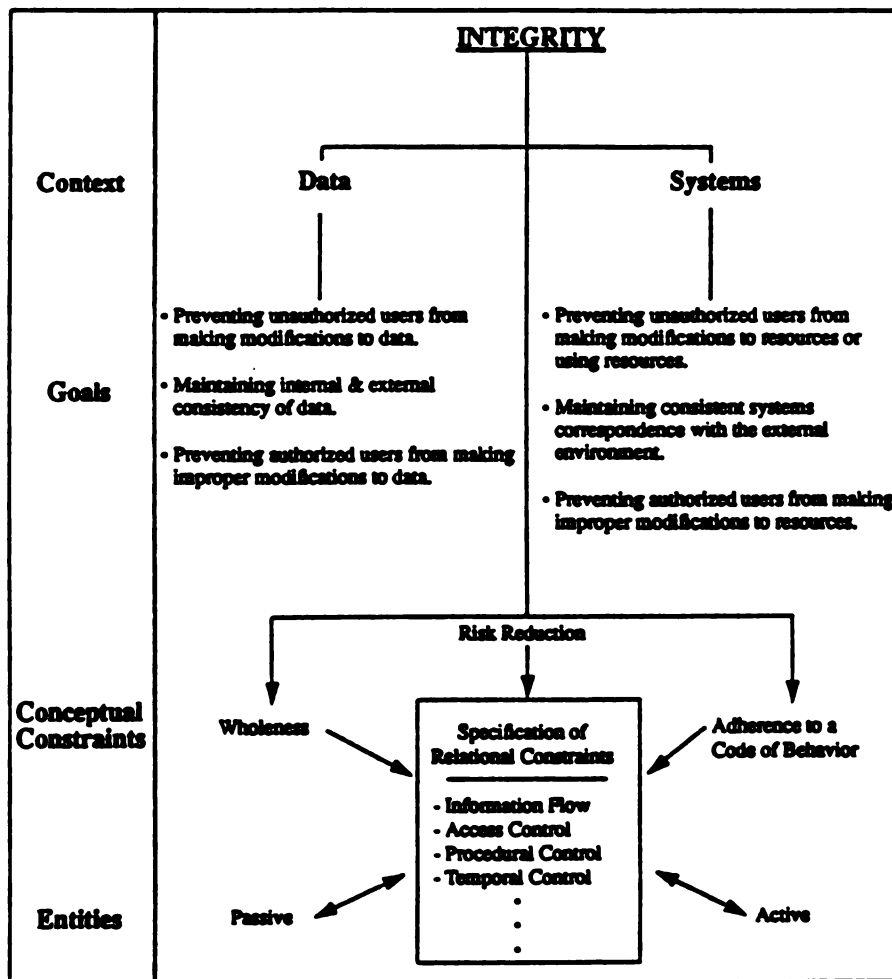


Figure 1. Integrity Framework

often deal, and because of the potentially large and widespread number of users and system nodes that must interact via such systems.

3.2. Systems Integrity

Systems integrity is defined here as the successful and correct operation of computing resources. Systems integrity is an overarching concept for computing systems, yet one that has specific implications in embedded systems whose control is dependent on system sensors. Systems integrity is closely related to the domain of fault tolerance. This aspect of integrity often is not included in the traditional discussions of integrity because it involves an aspect of computing, fault tolerance, that is often mistakenly relegated to the hardware level. Systems integrity is only superficially a hardware issue, and is equally

applicable to the AIS environment; an embedded system simply has less user-provided fault tolerance. In this context, it also is related closely to the issue of system safety (e.g., the safe operation of an aircraft employing embedded computers to maintain stable flight). In an embedded system, there is usually a much closer connection between the computing machinery and the physical, external environment than in a command and control system or a conventional AIS. The command and control system or conventional AIS often serves to process information for human users to interpret, while the embedded system most often acts in a relatively autonomous sense.

Systems integrity is also related to what is traditionally called the *denial of service* problem. Denial of service covers a broad

category of circumstances in which basic system services are denied to the users. However, systems integrity is less concerned with denial of service than with alteration of the ability of the system to perform in a consistent and reliable manner, given an environment in which system design flaws can be exploited to modify the operation of the system by an attacker.

For example, because an embedded system is usually very closely linked to the environment, one of the fundamental, but less familiar, ways in which such an attack can be accomplished is by distorting the system's view of time. This type of attack is nearly identical to a denial of service attack that interferes with the scheduling of time-related resources provided by the computing system. However, while denial of service is intended to prevent a user from being able to employ a system function for its intended purpose, time-related attacks on an embedded system can be intended to alter, but not stop, the functioning of a system. System examples of such an attack include the disorientation of a satellite in space or the confusing of a satellite's measurement of the location of targets it is tracking by forcing some part of the system outside of its design parameters. Similarly, environmental hazards or the use of sensor countermeasures such as flares, smoke, or reflectors can cause embedded systems employing single sensors such as infrared, laser, or radar to operate in unintended ways.

When sensors are used in combination, algorithms often are used to fuse the sensor inputs and provide control decisions to the employing systems. The degree of dependency on a single sensor, the amount of redundancy provided by multiple sensors, the dominance of sensors within the algorithm, and the discontinuity of agreement between sensors are but a few of the key facets in the design of fusion algorithms in embedded systems. It is the potential design flaws in these systems that we are concerned with when viewing systems from the perspective of systems integrity.

3.3. Information System Protection Goals

Many researchers and practitioners interested in INFOSEC believe that the field is concerned with three overlapping protection goals: *confidentiality*, *integrity*, and *availability*.

From a general review of reference material, we have broadly construed these individual goals as having the following meanings:

- a. Confidentiality denotes the goal of ensuring that information is protected from improper disclosure.
- b. Integrity denotes the goal of ensuring that data has at all times a proper physical representation, is a proper semantic representation of information, and that authorized users and information processing resources perform correct processing operations on it.
- c. Availability denotes the goal of ensuring that information and information processing resources both remain readily accessible to their authorized users.

The above integrity goal (b) is complete only with respect to data integrity. It remains incomplete with respect to systems integrity. We extend it to include ensuring that the services and resources composing the processing system are impenetrable to unauthorized users. This extension provides for a more complete categorization of integrity goals, since there is no other category for the protection of information processing resources from unauthorized use, the *theft of service* problem. It is recognized that this extension represents an overlap of integrity with availability. Embedded systems require one further extension to denote the goal of consistent and correct performance of the system within its external environment.

3.4. Integrity Goals

Using the goal previously denoted for integrity and the extensions we propose, we reinterpret the general integrity goal into the following specific goals in what we believe to be the order of increasing difficulty to achieve. None of these goals can be achieved with absolute certainty; some will respond to mechanisms known to provide some degree of assurance and all may require additional risk reduction techniques.

3.4.1. Preventing Unauthorized Users From Making Modifications

This goal addresses both data and system resources. Unauthorized use includes the improper access to the system, its resources and data. Unauthorized modification includes changes to the system, its resources, and changes to the user or system data originally stored including addition or deletion of such data. With respect to user data, this goal is the opposite of the confidentiality requirement: confidentiality places restrictions on information flow out of the stored data, whereas in this goal, integrity places restrictions on information flow into the stored data.

3.4.2. Maintaining Internal and External Consistency

This goal addresses both data and systems. It addresses self-consistency of interdependent data and consistency of data with the real-world environment that the data represents. Replicated and distributed data in a distributed computing system add new complexity to maintaining internal consistency. Fulfilling a requirement for periodic comparison of the internal data with the real-world environment it represents would help to satisfy both the data and systems aspects of this integrity goal. The accuracy of correspondence may require a tolerance that accounts for data input lags or for real-world lags, but such a tolerance must not allow incremental attacks in smaller segments than the tolerated range. Embedded systems that must rely only on their sensors to gain knowledge of the external environment require additional specifications to enable them to internally interpret the externally sensed data in terms of the correctness of their systems behavior in the external world. It is the addition of overall systems semantics that allows the embedded system to understand the consistency of external data with respect to systems actions.

- a. As an example of internal data consistency, a file containing a monthly summary of transactions must be consistent with the transaction records themselves.
- b. As an example of external data consistency, inventory records in an accounting system must accurately reflect the inventory of merchandise on hand.

This correspondence may require controls on the external items as well as controls on the data representing them (e.g., data entry controls). The accuracy of correspondence may require a tolerance that accounts for data input lags or for inventory in shipment, but not actually received.

- c. As an example of systems integrity and its relationship to external consistency, an increasing temperature at a cooling system sensor may be the result of a fault or an attack on the sensor (result: overcooling of the space) or a failure of a cooling system component such as a freon leak (result: overheating of the space). In both cases, the automated thermostat (embedded system) could be perceived as having an integrity failure unless it could properly interpret the sensed information in the context of the thermostat's interaction with the rest of the system, and either provide an alert of the external attack or failure, or provide a controlling action to counter the attack or overcome the failure. The essential requirement is that in order to have the system maintain a consistency of performance with its external environment, it must be provided with an internal means to interpret and flexibility to adapt to the external environment.

3.4.3. Preventing Authorized Users From Making Improper Modifications

The final goal of integrity is the most abstract, and usually involves risk reduction methods or procedures rather than absolute checks on the part of the system. Preventing improper modifications may involve requirements that ethical principles not be violated; for example, an employee may be authorized to transfer funds to specific company accounts, but should not make fraudulent or arbitrary transfers. It is, in fact, impossible to provide absolute "integrity" in this sense, so various mechanisms are usually provided to minimize the risk of this type of integrity violation occurring.

3.5. Conceptual Constraints Important to Integrity

There are three conceptual constraints that are important to the discussion of integrity. The first conceptual constraint has to do with the active entities of a system. We use the term *agents* to denote users and their surrogates. Here, we relate one of the dictionary definitions [3] of integrity, *adherence to a code of behavior*, to actions of systems and their active agents. The second conceptual constraint has to do with the passive entities or objects of a system. Objects as used here are more general than the storage objects as used in the TCSEC. We relate the states of the system and its objects to a second of Webster's definitions of integrity, *wholeness*. We show that the constraint relationships between active agents and passive entities are interdependent. We contend that the essence of integrity is in the specification of constraints and execution adherence of the active and passive entities to the specification as the active agent transforms the passive entity. Without specifications, one cannot judge the integrity of an active or passive entity. The third system conceptual constraint deals with the treatment of integrity when there can be no absolute assurance of maintaining integrity. We relate integrity to a fundamental aspect of protection, a strategy of *risk reduction*.

3.5.1. Adherence to a Code of Behavior

Adherence to a code of behavior focuses on the constraints of the active agents under examination. It is important to recognize that agents exist at different layers of abstraction (e.g., the user, the processor, the memory management unit). Thus, the focus on the active agents is to ensure that their actions are sanctioned or constrained so that they cannot exceed established bounds. Any action outside of these bounds, if attempted, must be prevented or detected prior to having a corrupting effect. Further, humans, as active agents, are held accountable for their actions and held liable to sanctions should such actions have a corrupting effect. One set of applied constraints are derived from the expected states of the system or data objects involved in the actions. Thus, the expected behaviors of the system's active agents are conditionally constrained by the results expected in the system's or data object's states.

These behavioral constraints may be statically or dynamically conditioned.

For example, consider a processor (an active agent) stepping through an application program (where procedural actions are conditioned or constrained) and arriving at the conditional instruction where the range (a conditional constraint) of a data item is checked. If the program is written with integrity in mind and the data item is "out of range," the forward progress of the processor through the applications program is halted and an error handling program is called to allow the processor to dispatch the error. Further progress in the application program is resumed when the error handling program returns control of the processor back to the application program.

A second set of applied constraints are derived from the temporal domain. These may be thought of as *event constraints*. Here, the active agent must perform an action or set of actions within a specified bound of time. The actions may be sequenced or concurrent, they may be performance constrained by rates (i.e., actions per unit of time), activity time (e.g., start & stop), elapsed time (e.g., start + 2hrs), and discrete time (e.g., complete by 1:05 p.m.)

Without a set of specified constraints, there is no "code of behavior" to which the active agent must adhere and, thus, the resultant states of data acted upon are unpredictable and potentially corrupt.

3.5.2. Wholeness

Wholeness has both the sense of unimpaired condition (i.e., soundness) and being complete and undivided (i.e., completeness) [3]. This aspect of integrity focuses on the incorruptibility of the objects under examination. It is important to recognize that objects exist at different layers of abstraction (e.g., bits, words, segments, packets, messages, programs). Thus, the focus of protection for an object is to ensure that it can only be accessed, operated on, or entered in specified ways and that it otherwise cannot be penetrated and its internals modified or destroyed. The constraints applied are those derived from the expected actions of the system's active agents. There are also constraints derived from the temporal

domain. Thus, the expected states of the system or data objects are constrained by the expected actions of the system's active agents.

For example, consider the updating of a relational database with one logical update transaction concurrently competing with another logical update transaction for a portion of the set of data items in the database. The expected actions for each update are based on the constraining concepts of *atomicity* (i.e., that the actions of a logical transaction shall be complete and that they shall transform each involved individual data item from one unimpaired state to a new unimpaired state, or that they shall have the effect of not carrying out the update at all); *serializability* (i.e., the consecutive ordering of all actions in the logical transaction schedule); and *mutual exclusion* (i.e., exclusive access to a given data item for the purpose of completing the actions of the logical transaction). The use of mechanisms such as dependency ordering, locking, logging, and the two-phase commit protocol enable the actions of the two transactions to complete leaving the database in a complete and consistent state.

3.5.3. Risk Reduction

Integrity is constrained by the inability to ensure absolute protection. The potential results of actions of an adversarial attack, or the results of the integrity failure of a human or system component place the entire system at risk of corrupted behavior. This risk could include complete system failure, corrupted representations of data, or complete loss of data. Therefore, a strategy of protection which includes relatively assured capabilities provided by protection mechanisms plus measures to reduce the exposure of human, system component, and data to loss of integrity should be pursued. Such a risk reduction strategy could include the following:

- a. Containment to construct "firewalls" to minimize exposures and opportunities to both authorized and unauthorized individuals (e.g., minimizing, separating, and rotating data, minimizing privileges of individuals, separating responsibilities, and rotating individuals).
- b. Monitors to actively observe or oversee human and system actions, to control the

progress of the actions, log the actions for later review, and/or alert other authorities of inappropriate action.

- c. Sanctions to apply a higher risk (e.g., fines, loss of job, loss of professional license, prison sentence) to the individual as compared to the potential gain from attempting, conducting, or completing an unauthorized act.
- d. Fault tolerance via redundancy (e.g., databases to preserve data or processors to preserve continued operation in an acknowledged environment of faults). Contingency or backup operational sites are another form of redundancy. Note: layered protection, or protection in depth, is a form of redundancy to reduce dependency on the impenetrability of a single protection perimeter.
- e. Insurance to replace the objects or their value should they be lost or damaged (e.g., fire insurance, theft insurance, and liability insurance).

4. Conclusions & Recommendations

This paper discusses the need for integrity to be promoted and preserved with respect to data and systems. It recognizes that this need exists for military, public, private, and commercial organizations who depend on the integrity of their systems and their data in automated information processing, process control, and embedded computing applications. Further, it shows that this need has been recognized since the early days of computer systems development. This latter point is important in that often the argument is made that we have had no worked examples of integrity and that we need to conduct a significant amount of research before any criteria are written. This paper tries to add some balance to that argument.

The paper discusses the difficulty of trying to provide a single definition for the term integrity as it applies to data and systems. We conclude that a single definition is probably not possible and, indeed, not needed. An operational definition that encompasses various views of the issue seems more appropriate. We

offer such an alternative so that progress beyond definitional aspects can be made. Our framework, or operational definition, provides a means to address both data and systems integrity and to gain an understanding of important principles that underlie integrity. It provides a context for examining integrity preserving mechanisms and for understanding the integrity elements that need to be included in system security policies. However, this study is only a beginning and remains incomplete in terms of fully addressing the topic.

The framework provides foundational material to continue the efforts toward developing criteria for building products which preserve and promote data and systems integrity. For some aspects, we conclude that there is sufficient understanding to write specific criteria, but for other aspects of such criteria, more experience, research, debate, and proofs of concepts will be needed. We believe that this partial knowledge should not delay the writing of criteria. It is the idea of concurrently pursuing both criteria and criteria-enabling research that we believe is key to making the rapid advances necessary to meet the recognized needs for integrity.

We recognize the need to establish a means to make the criteria, and thus the systems, evolvable with respect to integrity protection. Establishing this means may require more participation by systems vendors in the evolutionary development of integrity criteria than there was in the development of confidentiality criteria. The key here is to understand what is involved in designing systems for evolution so that criteria do not unnecessarily stifle new system designs or new concepts for preserving or promoting integrity.

We recommend that a criteria development study be undertaken to extend and apply the framework that has been developed in this paper. The criteria study should be conducted in parallel with protocol and mechanism demonstration/validation studies. This effort should interact with these two areas in receiving and providing direction. One major part of the criteria study should be form, a second part should be scope and specific content, a third part should address the evolution of criteria, and a final part should address the

linkages of product criteria to certification and accreditation of systems by using authorities.

References

- [1] Mayfield, T., J.M. Boone, S.R. Welke. 1991. *Integrity-Oriented Control Objectives: Proposed Revisions to the Trusted Computer Systems Evaluation Criteria (TCSEC)*, DoD 5200.28-STD. Alexandria, VA: Institute for Defense Analyses. IDA Document D-967.
- [2] Department of Defense. 1985. *DoD Trusted Computer System Evaluation Criteria*. DoD 5200.28-STD. Washington, DC: U.S. Government Printing Office.
- [3] Webster's Ninth New Collegiate Dictionary. 1988. Springfield, MA: Merriam-Webster, Inc.
- [4] Bonyun, David A. 1989. On the Adequacy of the Clark-Wilson Definition of Integrity. In *Report of the Invitational Workshop on Data Integrity, January 25-27, 1989, Gaithersburg, Maryland, B.5-B.5-9*. Gaithersburg, MD: National Institute of Standards and Technology.

A FRAMEWORK FOR DEVELOPING ACCREDITABLE MLS AIS

R. K. Bauer, J. Sachs, M. Weidner and W. Wilson

**Arca Systems, Inc.
2841 Junction Avenue, Suite 201
San Jose, CA 95134-1921
(408) 434-6633**

Abstract

Multilevel Security (MLS) is an integral requirement of many of our defense systems. Building a system to meet these requirements while still meeting stringent operational needs is quite challenging if not overwhelming. This paper highlights the tasks associated with certifying and accrediting a system to meet the security and operational needs of the end-user, then proposes a framework for integrating these tasks into the development process.

1. Overview

The ultimate objective of any Automated Information System (AIS) development or integration effort is to be accredited for operational use. To achieve this objective, the system must provide a satisfactory blend of security disciplines while accomplishing the intended mission.

Recent efforts integrating security into the development and acquisition process described in DOD-STD-2167A have focused attention on the TCSEC trust requirements of the TCB [6,7,11]. While this is a necessary condition for secure MLS operation, it is not sufficient. The fundamental premise of this paper is that prior efforts, while taking significant strides toward making trusted systems ubiquitous in all defense systems, have not gone far enough to ensure they will be operationally secure.

Operational security is often described as a chain comprised of links each of which represents a different security discipline (COMPUSEC, COMSEC, personnel security, administrative security, etc). This requires a balanced approach to allocating security requirements to each of the disciplines since the chain is only as strong as the weakest link. This collective set of requirements is the principal concern of the security certification efforts. Certification and MLS AIS development must be closely interrelated in order to achieve an accreditable system meeting its operational requirements. Key objectives of the development process and its products necessary to enforce this interrelationship are the ability to:

- 1) support consideration of mission requirements and security requirements prior to allocating requirements to trusted mechanisms.
- 2) support trade-offs between security disciplines and between overall security versus mission requirements.
- 3) address structure of complex integrated systems using newly developed and COTS components.

This paper presents background on the specific security tasks which must be performed and reviewed in support of certification (assessment of the overall security posture of a system in its intended operational context) and accreditation (the approval for operational use), and proposes a framework for developing Multilevel Secure Automated Information Systems (MLS AISs) meeting these objectives.

2. Certifying and Accrediting AISs

Accreditation is the step which ultimately determines whether an MLS AIS can be used to meet operational needs with acceptable risk. Although this step occurs at the boundary between development and operation, we discuss it first because it defines objectives for the earlier development and certification tasks. Accreditation is the step which determines that a system is

secure, or more accurately, *secure enough* given the fact that no system affords absolute security. The determination of what is secure enough is made in the light of operational mission requirements, sensitivity of data, and residual risk (remaining threats and vulnerabilities) of the system in the operational environment. This decision uses the certifier's assessment of the trustworthiness of the system based on thorough review and analysis of the features and assurance the integrator has provided to make the system trusted. These words go beyond just the requirements in the TCSEC to embrace all security disciplines including those addressing personnel, physical, procedural, communications, and emanations security requirements. The integrator's assertion that the system is trusted and the certifier's assessment of the degree of trustworthiness must cover all aspects of the system's adherence to its System Security Policy.

2.1 Accreditation

The Designated Approving Authority (DAA) is typically the individual responsible for the creation and maintenance of the information resources or the execution of the mission. The DAA determines the acceptable level of risk while balancing the security of the AIS against the operational benefit of meeting the system's mission. Government policies and directives mandate protection features for each of the security disciplines based upon the information types processed and the mission accomplished. An analysis of the adequacy with which these requirements are met provides the evidence that supports the DAA's accreditation decision.

Accreditation considers the relationship between the system's trustworthiness and its operational environment. Important operational and environmental considerations include:

- Range of data processed (e.g., Unclassified through Top Secret)
- User trustworthiness (e.g., clearances)
- Intended mode of operation (e.g., Dedicated, System High, Multilevel)
- Location of the operation (Inside a command center or in a commercial office building?)
- The owner of the information
- What is the mission and the operational concept

The DAA considers both residual risk and operational requirements in determining if the system will be allowed to operate. The DAA decides if the system:

- May operate as planned.
- May operate if specified changes are made verified prior to operation.
- May begin operation as planned on the condition that specified changes are made within some period after initial operation.
- Will not be allowed to operate.

Required changes may affect the system design or implementation, the way the system is operated, or the environment in which the system is operated.

The most intensive DAA involvement occurs at the end of the system development process when the final review is made to determine operational suitability. However, early DAA involvement is important, specifically with respect to the Security Concept of Operations and the intended operational environment. This allows tradeoffs to be made in a manner which adequately minimizes risk while maximizing operational flexibility. The DAA also reviews the system at regular intervals (typically 3-5 years) and after major system changes. Major system changes include altering the underlying security policy, changing the threats the system was designed to counter, modifying or exchanging the components enforcing the policy, or accumulated changes which may impact security enforcement. It is important to provide information to facilitate these reviews so that the DAA can make a sound and expeditious decision. Clear policy and design documentation and rigorous configuration control are needed to support these reviews. Careful analysis of just what each component is trusted to do is essential to the efficient review of the impact of changes to the system as a whole.

2.2 Modes of Operation

Accreditation of an AIS allows it to process data in a specific mode of operation. Modes of operation are defined in DoD 5200.28. The reliance on system enforced security controls varies widely among the various modes of operation. At one extreme is *dedicated mode* in which all users are cleared for all data on the system and have a need-to-know for all data. While accountability may be required in order to determine which users have accessed which data, the system is not counted on to enforce an access control policy restricting which data users can access. Accordingly, the security features required of the system and the degree of assurance required for those features is least in this mode of operation. In TCSEC terms, it is possible that a D system might suffice for dedicated mode although a C2 system would be more appropriate even in this environment because of the accountability it provides.

In *system high* operation all users are cleared for all data but may not possess a need-to-know. The system is not relied on to control access by users to data based on classification, but it does need to provide discretionary controls which can be used to control access to data based on a user's need-to-know. In system high mode a C2 system is usually sufficient. However, the C2 system provides no means for associating classifications with data and this association may be required if output is to be disseminated to anyone not cleared for the data on the system.

In *controlled mode* or *multilevel (MLS) mode*, some users do not possess a sufficient clearance or formal access authorization to access all of the data on the system. The distinction between MLS and controlled mode is the allowed size of the difference between the least cleared user and the classification of the most sensitive data. In either case the system is relied on to control access to data based on user clearances and data classification. This means the system must implement a mandatory access control (MAC) policy. In the TCSEC, MAC enforcement is first required at the B1 level. The driving force for introducing requirements of systems above the B1 level is the need for greater assurance than that provided by a system developed to meet B1 level requirements. Additional security requirements not considered in the TCSEC may be appropriate to meet the operational site's needs in terms of data integrity and availability. (Note: in the intelligence community, *Compartmented Mode* is used where data from multiple compartments is processed on the system and not all users are authorized access for all the compartments).

2.3 Certification

Certification assesses the operational risk of a system. The certification must verify and report on the environmental factors (e.g., physical and personnel security) and determine the trustworthiness of the system. The trustworthiness of a system can be viewed in terms of the security features provided by the system and the degree of assurance that those features are properly designed, implemented and integrated. Since no useful system can provide absolute security, it is necessary to make intelligent tradeoffs between alternative designs and implementations that accurately reflect the security and functionality issues associated with these tradeoffs. This requires the development of documents which clearly and precisely describe the security policy, the system design, and the interaction of the system enforced security features with the operational environment.

It is essential that the relevant documents be stated in a form which is as accessible as possible to developers, certifiers, users and the DAA. Only if all parties understand the issues involved in the tradeoffs between security and functionality is an informed decision possible. While this may seem obvious, experience has shown that considerable care needs to be taken to make sometimes arcane INFOSEC issues understandable to those not familiar with the technology and vocabulary [4,8]. Clarity of policy, requirements, and design documentation is especially crucial when one considers the large number of parties which may participate in this process and their need to share a common understanding and terminology. Interested parties may include security advisors such as MITRE, Aerospace Corporation and NSA and organizations responsible for external interfaces such as DIA, DCA, and NSA.

The certification personnel should be involved in the early stages of system development. Evidence regarding the system's ability to meet the security requirements should be presented to the certifiers in a top-down fashion (system-wide issues followed by subsystem issues followed by component issues) during system development. Thus, feedback regarding the more abstract system design can provide guidance when making more detailed subsystem and component design decisions. Once the system is complete, a bottom-up evaluation of the system should be performed so that the certifiers can use the evidence from lower-level evaluations in their analysis of higher-level subsystems and of the entire system.

The certifiers review evidence provided by the integrator supporting the claim that the system is trusted and evidence produced by independent verification and validation activities. For a trusted system composed of integrated trusted products, certification evidence for the system as a whole will typically depend on an evaluation of certification evidence for the subsystems and components. The NCSC's CSC-STD-003-85, *Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments* provides some help in selecting the appropriate class TCB for a given application environment, but there is no latitude in constraining the operational environment. It assumes worst case operational risk environment. Landwehr and Lubbes developed an approach to use other operational factors [9] to better refine the risk index of the environments guideline by reducing the risk of exploitation in the operational environment, resulting in reducing the trust requirements of the mechanism. Neither approach went far enough in considering the impact of the operational environment, and as such is inadequate to cover the certification and accreditation of large integrated systems consisting of COMSEC and COMPUSEC components with differing levels of trustworthiness in a variety of environments.

2.4 Security Mechanisms

In an MLS AIS there will be many required security mechanisms. These services will be drawn from COMPUSEC, COMSEC, and TEMPEST. The certifiers need a vehicle to determine that the right set of security services have been provided for the operational environment. The Security Policy Statement identifies basic requirements which must be met. However, it is usually possible to meet these requirements with more reliance on environmental controls or more reliance on system enforced controls. The document which relates the system enforced controls to its environment is the Security Concept of Operations. It allocates the security requirements between TCB features and environmental controls and identifies the interrelationships between the TCB and the environmental control measures. This is the first document which explicitly identifies the security features which the system will provide. Information on how these features will work and precisely what controls are enforced is provided by more detailed security documentation such as the Descriptive Top Level Specification.

Potentially mechanisms include the following: confidentiality, accountability, data integrity, and resource availability. Confidentiality covers Mandatory Access Control, Discretionary Access Control, and encryption. Accountability requires identification of individuals, authentication that the individual is who s/he claims to be, and audit of the user's security relevant actions. It is interesting to note that while a product evaluated against the TCSEC or TNI may come with assurance of access control and accountability features by virtue of its evaluation, it is unlikely to have been evaluated for integrity and availability features. Furthermore, integrity and availability of one component may be significant for system wide confidentiality or accountability if the component is used to store audit data or data on which access decisions are based.

2.5 Security Assurance

Trustworthiness cannot be established by emphatic assertion on the part of the developers. The integrator must provide evidence that the system is trusted. In the case of MLS AISs this evidence

is reviewed by the certification team. This review, along with independent testing and analysis, determines the trustworthiness of the system.

Assurance that the system meets its security requirements must be built in as the system is developed. It is more difficult and often impossible to gain the degree of assurance required for a trusted system by after the fact testing and analysis. Testing and a variety of analysis techniques during development and integration are an essential part of gaining the required assurance. The development process must be structured so that designers and implementors are aware of system security requirements and their implications for the design and implementation tasks. The design and implementation must be structured to support analysis of the adherence of the implementation to the system security requirements. This means the design and implementation must be understandable to certifiers. Assurance evidence comes in four forms: structured design, structured development process, testing, and analysis.

Structured design supports the analysis of security requirements adherence. Structured design starts with a carefully conceived security architecture. This architecture, which allocates the system security requirements to the subsystems responsible for the enforcement of the requirements, may be presented as part of the System Security Top Level Specification or as a separate document. An effective security architecture can limit the security responsibility of subsystems and ultimately the components used to implement them. This is an application of the principle of least privilege. Use of least privilege allows certifiers to focus their review on the security critical portions of the design and implementation and to further concentrate their review on the potential abuse of particular privileges. This principle should be followed throughout the design and implementation to the largest extent possible consistent with performance and functionality requirements. Since extensive use of least privilege will certainly impact performance, and is likely to impact usage flexibility, this tradeoff must be made with skill and care.

Structured trusted development has two facets. First, security requirements must be articulated and made available to designers and implementors in a manner which facilitates their use. This requires security analysis and documentation to be closely intertwined with the system development. Security requirements need to be flowed down to more detailed design levels. The implementor of any portion of the system must be able to understand the system security requirements for the task at hand. Second control of what components, software and hardware, are introduced into the final MLS AIS, must be applied throughout the development process. Moreover, configuration control must apply to all design documentation and security documentation as well as hardware and software.

The effectiveness of testing can only be as great as the knowledge of the requirements against which to test. This emphasizes the need for an effective flow down of security requirements to subsystems and components in order to facilitate security testing at these levels. In the case of components which are evaluated products, the requirements for the component need to be reviewed against the evaluated features in order to determine the applicability of evaluation testing. If additional features are counted on to enforce system security it may be necessary to perform additional testing on the product.

With today's operational AISs testing can never be exhaustive. Also security requirements tend to be negative requirements (i.e., the system never allows certain kinds of unacceptable behavior). For these reasons security testing must be supplemented by security analysis techniques to gain assurance in the correct design and/or implementation of the security features. These techniques include: security policy models, security top level specifications, verification, covert channel analysis, security fault analysis (SFA), and penetration testing.

Security Policy Models give a precise statement of the security policy requirements enforced by the Trusted Computing Base (TCB) and the types of operations provided by the TCB. Models may be

developed informally or formally, with the greater precision afforded by formal, mathematically rigorous models required for systems deployed in riskier environments.

Security Top Level Specifications provide insights into how security mechanisms work, although they do not include most implementation details. An informal *Descriptive Top Level Specification* (DTLS) describes not only the functionality provided by the TCB, but also the mechanisms used to make the TCB tamperproof and which guarantee that the TCB controls all accesses by subjects to objects. For more highly trusted systems, a *Formal Top Level Specification* (FTLS) is required. This is required in addition to, not instead of, the DTLS because formal specification languages do not support the specification of some important aspects of TCB behavior such as the interfaces between the TCB software and the hardware described in the DTLS.

Verification compares different descriptions of system behavior to show that the more concrete description satisfies all of the requirements of the more abstract description, for example, one can verify that an FTLS meets the requirements of the Security Policy Model. If both descriptions are formally presented, a formal verification, using mathematical proof, can be done. Formal verification is only practical at the design level (e.g., Model-FTLS verification) because the amount of detail at more concrete design levels and in the implementation quickly make formal verification using current state-of-the-art techniques for systems of even moderate size intractable.

Covert Channel Analysis is a technique for finding information flows contrary to the System Security Policy. Covert channels exist due to the possibility that the modulation of shared resources by one subject (or process) can be detected by another, even if the System Security Policy would normally prohibit communication between the two.

Security Fault Analysis is a technique familiar to the COMSEC community. Whereas the COMPUSEC community has put a large emphasis on software verification, SFA has focussed on analysis of hardware and the effect of faults on the security of the component. This was originally done when the complexity of devices made analysis down to the gate level practical. These techniques are now applied to more complex hardware bases. In addition to the continued need to apply SFA to critical hardware components, the principles of SFA provide lessons to COMPUSEC design and development such as the significance of single points of failure.

Penetration Testing assesses the strength and effectiveness of security features by means of an attempt to circumvent those features. The penetration testers analyze the system design and implementation for potential flaws and then attempt to utilize those flaws to penetrate the system. The results of penetration testing are only meaningful if it is carried out by experienced individuals.

2.6 System and Security Evolution

The discussion above was primarily from the point of view of a newly developed MLS AIS. Actually, most MLS systems are typically based on existing systems. Even when a system is developed from scratch with MLS as an objective, it is likely that the need to promptly address user requirements will necessitate an initial operating capability (IOC) with capabilities which will evolve as new technology becomes available. The system may also have to evolve because of changes in the environment which reduce or increase the reliance on system enforced security. Such development and integration efforts in the past have used a modified waterfall development model to provide some form of iterative development cycle [6,11]. New development models, for example the Spiral Model developed by Boehm [15], are being investigated for their contribution to the development of trusted systems [5].

The evolution of systems needs to be viewed from the point of view of its impact on system functionality, performance, and trustworthiness. New commercial-of-the-shelf (COTS) products

3. MLS AIS Development

The discussion to this point has highlighted the various tasks required to support certification and accreditation. Failure to integrate security requirements and the attendant deliverables into the development process and products has historically resulted in systems that were either operationally deficient, insecure or both [4,8]. Since security cannot be retrofitted, these tasks must be carefully integrated into the system development process. The security requirements must be clearly understood by all parties and appropriate requirements reflected throughout the design and development. It must also support informed tradeoffs between security, performance and functionality for alternative design and implementation approaches.

With these goals in mind we present the framework in Figure 1 as an approach to how particular documents and activities are related to the overall development and certification process. For clarity, the security tasks are called out from the standard development tasks, but the security tasks must be executed in close collaboration with the development tasks or fully integrated with the development process and products. The approach allows for separate security deliverables for COTS trusted products with existing security policy models, top level specifications, etc. The arrows in Figure 1 represent primary inputs. Later tasks will often identify required changes in the results of earlier tasks providing necessary feedback, for example, the development of a Top Level Specification may reveal deficiencies which must be corrected in the system design. Certainly the form these various activities and documents take will vary, especially when the process is used for an evolving system. The figure identifies some items which are optional depending on the complexity of the system and its subsystems. However, it is important that security tasks are performed which allow the tracking of security requirements through all development steps and that these tasks support intelligent and timely tradeoffs between operational flexibility, life cycle costs, performance, and security.

3.1 Requirements

Requirements are driven by mission directives and security directives. Applicable directives and their implications for the particular system under development are captured in the System Security Policy Statement. The System Security Policy Statement specifies the security requirements the system, in conjunction with environmental security controls, must enforce. The System Security Policy must be stated in the context of the mission requirements.

The system security policy must be complementary to the administrative, procedural, physical, and personnel controls present in or anticipated for the operational environment. The document which describes the interaction of system enforced controls and the system environment is the Security Concept of Operations. Both the System Security Policy and the Security Concept of Operations define the requirements for system security features. The Security Concept of Operations is an important document for supporting the intelligent determination of tradeoffs between security controls in the environment and system enforced controls. A Security Concept of Operations can be written to describe phases through which the MLS system may evolve. The Security Concept of Operations must be consistent with the System Concept of Operations. Likewise, the System Concept of Operations must reflect the System Security Policy.

Since the Security Concept of Operations and the System Security Policy define security requirements, they must be used by the developers to integrate security into the functional requirements. The System Security Concept of Operations yields both environmental requirements and system security requirements. This is noted in Figure 1, although the system secure environment description is likely to be a portion of the System Security Concept of Operations rather than a separate document. On the other hand, the System Security Policy Model will typically be a stand-alone document which describes the specific properties which must be enforced by the system TCB and the types of operations supported by the system TCB.

3.2 System Architecture

After the system's functional and security requirements have been established, a system architecture must be developed which defines subsystems and the functional and security requirements on those subsystems. The first step in this process is the development of system functional design. This has to reflect the system functional requirements and the security requirements as described in the Security Policy Model. Depending on the complexity of the system it may be desirable to have a Security Top Level Specification, either a DTLS or a DTLS and FTLS, based on the level of trustworthiness required. However, since security requirements on subsystems are reflected in Subsystem Security Policy Models, it may be possible to incorporate sufficient information about subsystem interaction in the System Security Policy Model and in that way omit an explicit Security Top Level Specification. The subsystem requirements drawn from the system functional design complete the system architecture phase. The Subsystem Security Policy Models must reflect the security requirements allocated to that subsystem and the impact of the functional requirements identified in the subsystem's requirements statement.

3.3 System Design

In this phase subsystem TLSs are developed to describe the security features implemented in the subsystems. Because component Security Policy Models will not always be available, the subsystem TLSs will be relied on to describe how security features work in each subsystem and how components interact to implement those security features. Also, the role of the component Security Model may be replaced by other documents such as the Software Requirements Specification for COMSEC components. The component requirements must reflect both the functional requirements flowed down in the subsystem designs and the security requirements in the subsystem Security Top Level Specification. These component requirements form the basis for selection of COTS products (whether COMSEC or COMPUSEC) or the design and implementation of newly developed components.

3.4 System Implementation

System implementation is accomplished through the design and implementation of the newly developed components which comprise the system and the selection of COTS products in the case of components for which suitable products exist. Security specifications for components detail the security aspects of the component design. Depending on the type of component, the nature of the component security specification may vary significantly. For a complex trusted component, whether newly developed or a COTS product, the specification may take the form of a traditional Security Top Level Specification. For a COMSEC component, the STLS may take the form of a Software Program Specification. In the case of particularly simple components, the component STLS may be omitted.

3.5 Integration and Test

The steps identified in the framework provide the basis for security test and evaluation. Figure 1 shows where testing and analysis techniques can be used to ensure that security design and requirements have been accurately followed. Security tests on subsystems can be performed using test cases developed from the Security Top Level Specifications for the subsystems. For those components where Security Policy Models and/or Security Top Level Specifications have been developed, these documents can be used to generate test cases for component testing. Otherwise, the Subsystem Security Top Level Specification will have to be relied upon to provide sufficient detail on the required security behavior of the component to serve as a starting point for test case generation. Security tests for the integrated system can be performed based on test cases derived from the System Security Top Level Specification, if one has been developed, or directly from the Informal Security Policy Model if it is sufficiently detailed.

3.6 Security Analysis

Section 2.5 described techniques which can be used to perform security analysis at various points in the system development. Verification can be used to demonstrate that the System Security Top Level Specification meets the requirements of the Security Policy Model and that the subsystem Security Policy Models meet the requirements of the System Security Top Level Specification. Alternatively, the Subsystem Security Policy Models can be shown directly to meet the requirements of the System Security Policy Model. For each subsystem, verification can be used to justify that the design reflected in the Subsystem Security Top Level Specification is consistent with the Subsystem Security Policy Model. Where component models and security specifications exist, the verification can be carried down to that level. Otherwise, testing and review of correspondence of the component implementation to the requirements of the Subsystem Security Top Level Specification can be used to show that the component satisfies its specified security requirements.

Penetration testers will use all of the documentation and specifications generated in this process to determine potential faults to exploit. The penetration tester must eventually test these potential faults by attempting to penetrate the integrated system. However, some faults may be dependent solely on the functionality of a particular component or subsystem. These faults can be tested as soon as the component or subsystem is available without waiting for integration. This early feedback can support penetration testing which is extensive enough to provide reasonable assurance, and the need to deploy the system promptly.

3.7 Tradeoffs

The framework described above allows security and functional requirements to be considered at all stages of system development so that intelligent tradeoffs can be made. The first tradeoffs are made in developing a System Security Policy which is sufficiently stringent to meet the requirements of relevant directives and yet flexible enough to allow mission requirements to be met. The next tradeoff is between system enforced and environmentally enforced security. This tradeoff is made as the Security Concept of Operations is developed, reviewed, and updated. In the system architecture phase the system functional design is the vehicle for allocating requirements to subsystems. From the security point of view this allocation should be made in such a way as to minimize and simplify the TCB. However, these considerations need to be considered in the context of their effects on performance and flexibility. In the implementation phase, tradeoffs between least privilege and performance will again need to be made as the subsystem designs are developed. At this stage an important factor in that tradeoff will be the availability of evaluated products which can provide some of the security enforcement.

Finally, it should be noted that this is necessarily an iterative process, for example, as subsystem designs are developed it may become clear that a reallocation of requirements among subsystems would enhance security, performance, functionality or some combination of these. Also, as new COTS products become available, an altered subsystem design or even system functional design may be appropriate. It is important to enforce configuration control on this process so that even with iterations the set of accepted documents, specifications, and implementations are consistent.

3.8 Operational Documentation

One of the important outputs of the MLS system development process is the documentation which tells privileged users and other users how to interact with the system and maintain security. Improperly used security controls can be just as vulnerable as insufficient or improperly implemented controls. Two key documents which tell users how to properly use and maintain the security features are the Trusted Facility Manual (TFM) and Security Features User's Guide (SFUG). The TFM describes the functions available to privileged users such as the system administrator and system security officer as well as what these users must do to properly initialize and maintain the system, and securely recover from system failures. The SFUG explains to

general users what security controls are enforced and what the user's role is in conforming to the security policy. Proper training of all users on their security responsibilities and more extensive training for privileged users on how to keep the system secure are essential for the operational system to be run securely. Clearly written, comprehensive documentation plays a central role in making sure the users understand their security responsibilities.

4.0 Conclusion

This integrated approach to system development and security engineering allows effective tradeoffs between system security controls and operational requirements to minimize the total cost of development, operation, and maintenance. It ensures that the broader set of security requirements, and not just trust requirements, are adequately considered throughout the development process. Finally, it supports the integration of complex systems comprised of trusted and untrusted COTS products, and newly developed components. This MLS AIS development approach provides the basis for successful certification and accreditation of the fielded operational system.

5.0 References

- [1] Department of Defense, Security Requirements for Automated Information Systems DoD 5200.28, March 1988.
- [2] National Computer Security Center, Computer Security Requirements -- Guidance for Applying the DoD TCSEC in Specific Environments, CSC-STD-003-85, 25 June 1985.
- [3] National Computer Security Center, Department of Defense Trusted Computer System Evaluation Criteria, DoD 5200.28-STD, December 1985.
- [4] D. J. Bodeau and M. J. Reece, "A Multilevel-Mode System for Space Applications: Lessons Learned," in *Proceedings of the Sixth Computer Security Applications Conference*, IEEE Society Press, December 1990.
- [5] A. Marmor-Squires, B. Danner, J. McHugh, L. Nagy, D. Sterne, M. Branstad, and P. Rougeau. "A Risk-Driven Process Model for the Development of Trusted Systems," in *Proceedings of the Fifth Computer Security Applications Conference*, IEEE Society Press, December 1989.
- [6] T. C. V. Benzel, "Developing Trusted Systems Using DOD-STD-2167A," in *Proceedings of the Fifth Computer Security Applications Conference*, IEEE Society Press, December 1989.
- [7] S. D. Crocker and E. J. Siarkiewicz, "Software Methodology for Development of a Trusted BMS: Identification of Critical Problems," in *Proceedings of the Fifth Computer Security Applications Conference*, IEEE Society Press, December 1989.
- [8] C. R. Pierce, "Experiences in Acquiring and Developing Secure Communications-Computer Systems," in *Proceedings of the Thirteenth National Computer Security Conference*, National Computer Security Center, October 1990.
- [9] C. E. Landwehr and H. O. Lubbes, *An Approach to Determining Computer Security Requirements for Navy Systems*, Technical Report NRL 8897, Naval Research Laboratory, May 1985.
- [10] A. C. Hoheb, "Integrating Computer Security and Software Safety in the Life Cycle of Air Force Systems," in *Proceedings of the Thirteenth National Computer Security Conference*, National Computer Security Center, October 1990.
- [11] T. C. V. Benzel, "Integrating Security Requirements and Software Development Standards," in *Proceedings of the Twelfth National Computer Security Conference*, National Computer Security Center, October 1989.
- [12] W. Norvell, "Integration of Security into the Acquisition Life Cycle," in *Proceedings of the Twelfth National Computer Security Conference*, National Computer Security Center, October 1989.
- [13] F. Tompkins and R. Rice, "Integrating Security Activities into the Software Development Life Cycle and the Software Quality Assurance Process," *Computers and Security*, Vol. 5, No. 3, September 1986.
- [14] D. E. Bell, "Working Towards A1," in *Proceedings of Seventh DOD/NBS Computer Security Conference*, DOD Computer Security Center, September 1984.
- [15] B. W. Boehm, "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, May 1988.
- [16] M. S. Deutsch and R. R. Willis, *Software Quality Engineering*, Prentice-Hall, 1988.

GENERALIZED FRAMEWORK FOR ACCESS CONTROL: TOWARDS PROTOTYPING THE ORGCON POLICY¹

Marshall Abrams* Jody Heaney* Osborne King* Leonard LaPadula+ Manette Lazear* Ingrid Olson*

The MITRE Corporation *7525 Colshire Dr., McLean, VA 22102

+Burlington Rd., Bedford, MA 01730

1 INTRODUCTION

The Generalized Framework for Access Control (GFAC) was introduced in [1, 4] as a framework for studying and constructing access control policies in Automated Information Systems (AISs). This paper discusses a prototyping effort that uses the GFAC concepts. Further, it describes a security policy and the experience gained through implementing a prototype based on that policy.

GFAC asserts that all access control policies can be expressed as *rules* specified in terms of *attributes* and other information controlled by *authorities*. All policies can be expressed within this framework, including policies conventionally implemented through trusted processes and privilege mechanisms. The GFAC concepts include four factors representing dimensions of choice and constraints to the designer of a trusted AIS:

- Access Control Information (ACI) – Characteristics or properties of subjects and objects. ACI names are used in specifying the rules of the system; their values are used by the access control rules.
- Access Control Context (ACC) – Additional information, such as time of day, used in access control decision making.
- Access Control Authorities (ACA) – Authorized agents who specify ACI, ACC, and rules.
- Access Control Rules (ACR) – The set of formal expressions of policy for adjudicating requests by subjects for access to objects.

1.1 Markings

Within the DOD/intelligence community, numerous dissemination/handling restrictions and markings are applied to the manual handling of classified documents. Examples include NOFORN (Not Releasable to Foreign Nationals), ORCON (Dissemination and Extraction of Information Controlled by Originator), and REL XX (Authorized for Release to (name of country(ies)/international organization), which are defined in DCID 1/7 [2]. Williams and Day [8], and also Graubart [3], provide excellent discussions of the complexities of such markings for classified documents, and the inadequacies of current automated systems in handling them.

Such restrictive control markings are examples of a class of existing access control policies that limit the dissemination of information beyond the traditional Mandatory Access Controls (MAC) and Discretionary Access Controls (DAC) specified in the Trusted Computer System Evaluation Criteria (TCSEC) [5]. MAC and DAC have almost become synonymous with access control in automated systems, when in practice there are many other policies in existence in the paper world that are reasonable candidates for automation. Although MAC in particular, and DAC to some degree, are useful and reasonable policies for some environments, support for additional policies in automated systems is needed. The GFAC effort is attempting to demonstrate that a more general, useful model of access control is feasible and necessary to support the many access control policies.

In the DOD/intelligence community, other policies must be satisfied in addition to MAC (i.e., in addition to having the appropriate security clearance, the user must also satisfy the access rules of the additional policy). In the unclassified world, such policies may be implemented through non-disclosure agreements and contractual limitations on information disclosure. MAC *may* not be a requirement in conjunction with other non-DOD policies. For example, the Bureau of Labor Statistics uses RELEASABLE AT <time,date> to safeguard unemployment figures. This information is highly protected until <time,date> when it is widely distributed.

Using GFAC, appropriate markings and other supporting information needed to make access control decisions to implement such restrictive control markings can easily be included as subject/object ACI or additional ACC

¹ This work was supported in part by the MITRE Corporation as MITRE-Sponsored Research and in part by the U.S. Army as Mission-Oriented Investigation and Experimentation under contract DAAB07-91-C-N751. Technical direction for the research was provided by the National Security Agency.

information. Development of the necessary access controls is theoretically straightforward using GFAC. Note that the strength or universal applicability of access control rules is independent of the information on which the rules base their decisions. Thus, the implementation of a marking policy can be just as strong and pervasive in a trusted system as the implementation of a traditional MAC policy.

1.2 Prototyping based on the GFAC Concepts

To provide a tangible proof-of-concept, we are developing a prototype for one of the additional policies noted above that was, in turn, expressed using the GFAC concepts. ORCON is the most restrictive policy defined in DCID 1/7 and, therefore, was selected as the basis for an automation policy. Development of an ORCON-like policy has been instructive; this paper is intended to help share some of the experiences. The following sections discuss the ORGCON policy, the ORCON-like policy that forms the basis of the prototype, and numerous prototype issues, design decisions, results, and lessons learned. There are some characteristics of ORCON, such as special instructions relating to incorporation or retention period, that were not included in the ORGCON policy. The term ORGCON, instead of ORCON, is used so that a precise AIS policy can be implemented without usurping the Government's definition(s) of ORCON.

2 ORCON POLICY

The Organization Controlled (ORGCON) policy (described in Section 3) was developed as a practical example of using the GFAC concepts. The ORGCON policy is a policy for AISs that builds upon the ORCON ("Dissemination and extraction of information controlled by originator") dissemination control on paper documents. In choosing to develop ORGCON and prototype based on this policy, we are attempting to transfer a well-established policy from the control of paper documents to the control of information in an AIS. For completeness, this section provides a high-level description of the ORCON policy that the ORGCON policy is based on.

2.1 The ORCON Dissemination Control

ORCON is only one of a number of restrictive control markings defined in DCID 1/7 applied in the dissemination and use of intelligence information and related materials. These markings represent handling policies that limit the authority of recipients of the information to use or transmit it. ORCON requires the permission of the originator to distribute information beyond the original receivers designated by the originator. For the purposes of this paper, the following extract from DCID 1/7 defines the ORCON marking:

This marking is used, with a security classification, to enable a continuing knowledge and supervision by the originator of the use made of the information involved... Information bearing this marking may not be disseminated beyond the headquarters elements of the recipient organizations and may not be incorporated in whole or in part into other reports or briefings without the advance permission of and under conditions specified by the originator.

2.2 The ORCON Originator and Recipient

The originator has not only the right, but the responsibility to identify and mark information as ORCON information. The originator also has the responsibility to explicitly identify what organizations will be indicated on the distribution list for the specific information.

"Originator" is not defined in DCID 1/7. We believe that the authority to control dissemination of the information rests within an office or organization. An individual may only have dissemination authority by virtue of acting on behalf of that office or organization. This implies that the originator of ORCON information is never an individual user. In fact, the originator is always an office or organization code or some analog thereto. An individual does not own such information any more than an Air Force pilot owns an F-15. Similarly, ORCON material is never addressed and distributed to an individual. Some information may, however, be addressed to a commander only. Even this information is likely to be handled by a limited number of people in addition to the designated recipient (e.g., executive officer). The term designated recipient, in the preceding sentence, does not connote an individual, but rather the role filled by an individual (e.g., Commander-in-Chief (CINC)).

2.3 ORCON Information Dissemination

ORCON information may be received and processed in a number of different ways. Message traffic is the most common. Messages may be read from a terminal, posted on a read-board, or routed as paper-copy. Access to the data may be via remote terminal access from a terminal to a database at a central location. Information may also be received via a dedicated or special purpose system.

The internal access to and distribution of ORCON marked information depends on its form and content, as well as the number of staff with assigned responsibilities in the area related to the information. Access and distribution are also dependent on the tools available to process ORCON information.

The originating organization for ORCON information is either explicit (a message has a "from" address indicating the originating organization) or implicit (remote access to a database implies that the organization hosting the database is the originator). ORCON information is transmitted by some method (e.g., photo-copy, electronic transmission) that effectively creates a new copy of the information at the destination. Handling, retention, and destruction of ORCON information, by both the originator and recipient organizations, varies. Handling, retention, and destruction depend, in part, on other security markings on the information. Old copies of information may be destroyed after database updates. Reports containing ORCON information are more likely kept on file for a specified period of time.

3 THE ORGCON POLICY

The ORGCON policy uses the ORCON policy concepts applied to paper documents, but was developed as an AIS policy to control the dissemination of information. ORGCON is a policy for non-discretionary group-based access control. Groups are discussed further in Section 4. The primary elements of the ORGCON policy are as follows:

- ORGCON information is owned by an originating organization and ownership is not alterable.
- ORGCON information is distributed only to an identified list of recipient organizations.
- The list of authorized individuals of each recipient organization is maintained by a recipient representative.

3.1 Originator and Recipient Representative Roles

The ORGCON policy controls the ownership and dissemination of ORGCON information (information marked ORGCON). ORGCON information is owned by its originating organization. The originating organization is represented by one or more individuals acting in the role "originator representative" (ORGREP). Any individual may generate information that may eventually be designated with the ORGCON marking, but only an ORGREP can mark the information ORGCON and specify a distribution list of recipients.

Individuals acting in the role of "recipient representative" (RECREP) specify the individuals who are authorized to receive ORGCON information at the recipient organization.² Example recipient organizations include the headquarters staff and CINC. Note that ORGCON differs from ORCON by the introduction of the RECREP, which is believed to be a necessary and practical step. The ORGREP cannot be expected to be aware of personnel changes in the recipient organization, nor will (s)he be likely to have the privileges to redefine the membership of the recipient organizations. The originator and RECREPs are authority agents (authority component of GFAC), perhaps the Information System Security Officer (ISSO) or Security Administrator.

3.2 ORGCON and ORGCON-C Markings

Two markings are defined for the ORGCON policy. The ORGCON marking identifies an object as being under ORGCON policy control. ORGCON-C is a special marking that identifies an object as a candidate for handling under the ORGCON policy. The ORGCON-C marking identifies the object as being write-accessible only to the individual user who created it. By convention, this user is referred to as the owner of the ORGCON-C object. The owner may read, write, or delete the object. The ORGREP may read the object and is privileged to change the marking; the only authorized changes are from ORGCON-C to ORGCON. A normal progression would be for the individual owner to pass an ORGCON-C object to the ORGREP for marking as ORGCON and distribution.

3.3 Reading ORGCON Objects

An individual can only obtain read access to ORGCON information if the individual is a member of a recipient organization that is on the distribution list. This condition for read access holds for the creator of the ORGCON information and all representatives of organizations. That is, once the information is marked ORGCON, there are no exceptions to the conditions for read access. In order for an ORGREP or RECREP to be able to read an ORGCON object, they must be members of a group named on the distribution list. As a practical matter, the ORGREP role will

² Note that the ORCON policy identifies the headquarters element of an organization as the recipient. The ORGCON policy has been generalized and does not imply the headquarters element as the recipient.

probably be placed on the distribution list and the RECREPs will be part of each respective recipient organization. Other policies can be envisioned under varying circumstances.

3.4 Copying ORGCON Objects

In the process of distributing ORGCON information, multiple copies of the information may be generated. Many different mechanisms could be employed for distributing ORGCON objects within a recipient organization, depending on the AIS architecture employed. For the purpose of this paper, we discuss two possible architectures. The first architecture has only authorized users accessing a shared file system (e.g., a single multi-user system, a shared file server). Given this architecture, only one copy of an ORGCON object is required. The second architecture has users without access to shared file systems (e.g., separate single or multi-user systems, non-client-server workstations). These users will require individual copies. Therefore, the ORGCON policy must control the copying of ORGCON information, as well as its final disposition. The original of any ORGCON information logically resides with the originating organization.

The major points of the ORGCON copy policy are identified below and discussed in the following paragraphs.

- Only RECREPs or a daemon performing privileged system operations can copy ORGCON information for distribution to those individuals defined as recipients.
- Any recipient of an individual copy of ORGCON information can view and dispose of his/her own copy of the information.
- No individual recipient of ORGCON information can copy that information.

Many schemes for marking ORGCON objects are possible. In one scheme, an object marking has two fields, an ORGCON field and an ORGCON-copy-control field. When an object enters the ORGCON system it is marked ORGCON by the ORGREP. At this point the ORGCON-copy-control field defaults to Null. This configuration of marking automatically identifies the object containing the information as the original version. When a copy is made the ORGCON-copy-control field is filled-in. This could be done in several ways. The field could contain a copy number or some designation which identifies the recipient of the copy. This is summarized in the table 1.

Table 1. Possible Implementation of ORGCON Control Fields

STATUS	FIELD	
	ORGCON	ORGCON-Copy-Control
ORGCON original	ORGCON	Null
ORGCON copy	ORGCON	Recipient ID

The RECREP (or daemon) is privileged to copy ORGCON objects for distribution to those users defined as belonging to the recipient organization. This distribution may be performed manually, but is performed by a process (e.g., a daemon) with the privilege to make and distribute the copy, running on behalf of the RECREP. Each copy of ORGCON information created carries the distribution list for that information and an identifier for the originating organization.

Individuals in the recipient organization who are not RECREPs may not copy ORGCON information. The access of these individuals to ORGCON information is limited to reading and disposal. Each individual recipient is responsible for proper disposal of their individual copy of ORGCON information. The RECREP may delete the recipient organization copy of the ORGCON object. ORGREPs are responsible for proper disposal of the original ORGCON information.

3.5 Handling an ORGCON Object

There are several different roles associated with the ORGCON policy, and each role has different responsibilities and privileges associated with it. In the prototype, two roles are implemented: the ORGREP and the RECREP.

When an object is marked ORGCON, the ownership of the object is changed to the ORGREP, the designated authority for marking and extending access to ORGCON objects. The ORGCON policy rules specify the authority of the originator representative role and the recipient representative role relative to granting read access to ORGCON objects.

An important feature of the ORGCON policy is that the distribution list for ORGCON information is part of the object. The ORGREP is the only role responsible for creating the distribution list (DL) for an ORGCON object. The policy decision was made that once an ORGCON object is created and the distribution list attached, no changes can be made to the list of recipients. At the time of distribution, the ORGCON object should be thought of as including the DL. Consider that the aggregation of object and DL could change the hierarchical level classification. We have chosen not to implement this in the current prototype, but it is one example of why the ORGCON policy forbids changes once the DL is attached.

3.6 ORGCON Control of Access

The access control rules of the ORGCON policy are summarized in table 2.

Table 2. ORGCON Control of Access

WHEN THE REQUESTED ACTION IS:	THE FOLLOWING CONDITIONS MUST BE MET:
Mark as ORGCON-C	User is owner
Change ORGCON-C to ORGCON	User is ORGREP
Read ORGCON-C object	User is owner or ORGREP
Read ORGCON object	User belongs to a recipient organization, or daemon
Delete ORGCON object copy	User received an individual copy of the ORGCON object, or copy belongs to recipient organization and user is RECREP
Delete ORGCON object original	User is ORGREP
Copy ORGCON object	User is RECREP, or daemon
Write ORGCON-C object	null

4 ROLES/GROUPS AND DAC

4.1 Roles and Groups

To develop a prototype for the ORGCON policy, identification of several roles (i.e., equivalence classes of users) is required. Each of these equivalence classes is identified by name. The TCSEC implicitly defines groups as part of the specification of DAC as follows:

The enforcement mechanism ... shall allow users to specify and control sharing of those objects by named individuals or defined groups of individuals, or both...

The Trusted Network Interpretation (TNI) distinguishes between users and roles:

Note that "users" does not include "operators," "system programmers," "technical control officers," "system security officers," and other system support personnel. They are distinct from users and are subject to the Trusted Facility Manual and the System Architecture requirements. Such individuals may change the system parameters of the network system, for example, by defining membership of a group. These individuals may also have the separate role of users.

The concept of named equivalence classes of users, however, is too important a concept to be used only with DAC. The usage has, therefore, been extended by prepending the policy name as an adjective when necessary for clarity (e.g., DAC-group, ORGCON-group). The meaning is clear: the members of this identified set of users are to be treated identically with respect to the specified policy. There may be multiple groups, each having different privileges relative to the specified policy.

Informally, a group is a collection of users that share a set of access control attributes. An individual member of the group may act with any of the access privileges authorized for the group. The composition of a group is determined by an appropriate authority, and a primary purpose of creating groups is essentially administrative convenience. However, it is important to note the support for separation of function provided by groups. A role may be viewed as a particular

kind of group. The distinguishing feature of a role is the identification of unique privileges with respect to the stated policy. When a user takes on a role (usually explicitly), the user relinquishes the privileges associated with their previous role. A role is not associated with an individual user, but with a set of users (i.e., a group) authorized for the specific role. ORGCON-roles defined in this paper are summarized in Table 3.

Table 3. ORGCON-Roles

ROLE	FUNCTION
Originator representative	Marks an ORGCON-C object as ORGCON and affixes the distribution list (list of recipient organizations)
Recipient representative	Controls membership of recipient organization; may copy ORGCON object for distribution to recipients

4.2 Traditional DAC Policy

This effort has caused us to explore the nature of DAC and how it fits in the GFAC view of access control policies and their implementations. Primarily, DoD Directive 5200.28, the TCSEC, and the DAC Guide [6] have been consulted. It appears that the term DAC is used interchangeably to refer to both a set of mechanisms and a policy. The DAC policy defined by the TCSEC is referred to here as traditional DAC. Traditional DAC allows an authorized user to determine who is authorized what mode of access to an object. Nothing is stated about how the user receives authorization for specific modes. In the literature, the initial authorized user is often identified as the owner of an object, but this is not necessarily the case. For that matter, the concept of ownership is not universally defined. The DAC policy is really a special case of the principle of least privilege; that special case is need-to-know.

There are also numerous supporting policies that are NOT associated with DAC. With hindsight and the benefit of the GFAC perspective, we note that many, perhaps all, of the weaknesses attributed to traditional DAC actually identify the absence of supporting policies. GFAC provides an opportunity to experiment with the design of other identity-based policies to overcome DAC deficiencies and to meet other policy objectives. Nothing in the TCSEC prohibits the addition of these or other supporting policies to DAC. However, it is not clear if anyone has ever done so. A precedence has thereby developed defining traditional DAC.

The two major shortcomings of traditional DAC are the lack of an inheritance policy and the lack of accountability. The lack of an inheritance policy means that the mechanism only protects the container, not the information. Once the information is read from the container, there are generally no controls on what can be done with the information; there is no ability to control copies. The lack of accountability means that the DAC mechanisms are vulnerable to Trojan Horses, since programs executing on behalf of a user generally assume the privileges of the user.

5 THE PROTOTYPE

In this section, the goals of this prototyping effort are described and an overview of the System V/MLS prototyping environment is provided. Some advantages, difficulties and limitations of adding an additional policy to an existing secure system [9] are also discussed.

5.1 Prototype Goals

There are two main goals to this initial GFAC prototype effort:

1. To demonstrate a prototype based on the GFAC concepts.
2. To implement an access control policy, namely ORGCON, in addition to MAC and DAC.

To demonstrate the GFAC concepts, the prototype must satisfy the following three goals. Note that accomplishment of these goals makes it possible to implement any access control policy.

1. The prototype must provide for the creation, maintenance, and change of those ACI relevant to the particular access control policy being implemented.
2. The prototype must provide the appropriate set of rules necessary to implement the given policy.
3. The prototype must embody an explicit definition of authority with respect to the given policy, either through well-defined roles, through the rules, or in the ACI.

The second goal of the GFAC effort, to implement an additional policy, is important in order to demonstrate the feasibility of implementing policies other than MAC and DAC.

For the sake of expediency, a system that already provides a B-level MAC policy was used as the prototype base. That is, an existing TCB was modified to execute additional policies. We expected that many of the mechanisms used to implement MAC sensitivity labels might carry over to the handling of the ACI for additional policies. Portions of the TCB outside the kernel (i.e., the reference monitor implementation) were expected to be directly useful. AT&T System V/MLS [7] was selected as the host base for development of the prototype.

5.2 Prototype Environment

System V/MLS supports two types of access controls: DAC and MAC. The discretionary controls provided are identical to those controls provided by standard UNIX System V. DAC permits owner control of access to resources by other users, and is implemented via the user/group/other mechanism. Permission to read, write, and/or execute (for files) and search (for directories) may be set for each class of users (owner of the object, group associated with the object, and for others (all system users)).

In addition, System V/MLS provides mandatory controls, defining access to resources based on labels. System V/MLS controls access to resources using the current operating privilege of the user and the privilege requirements associated with a resource. Privilege is the term used to refer to the DAC group and the MAC label associated with a user or a resource. The label is the combination of a hierarchical level and zero or more categories. A privilege can be thought of as an instance of a group at different levels and categories. While an object has only one privilege associated with it, users may change their current operating privilege (i.e., the label and group associated with them). The range of labels over which a user may operate is referred to as the user's clearance. A user, however, may not necessarily be a member of all privileges defined in that range.

Files or directories may only be created in a directory that has a label identical to the user's current operating label. Once created, however, the files/directories' labels may be upgraded. Within DAC and MAC, files and directories are accessed based on the user's current operating label (i.e., the label part of the privilege). This label must dominate (for read access) or be identical to (for write access) the label of the file/directory the user is trying to access. Formal models of System V/MLS and of the ORGCON policy are in preparation.

5.3 Difficulties/Tradeoffs

While developing the prototype on an existing security system has its advantages, there are also numerous difficulties and tradeoffs in retrofitting an existing system. A major dilemma was deciding which of the following two objectives took precedence in the prototype:

1. Strict adherence to the GFAC concepts and structure which could require extensive changes or additions to the existing system base.
2. Implementation of the additional policy using capabilities of the existing system without necessarily demonstrating the GFAC concepts.

Occasionally, the effort was limited by existing structures within System V/MLS that were not alterable. Therefore, 'workarounds' were devised to implement the controls of the ORGCON policy. For example, the privilege concept in System V/MLS, the coupling of the label (i.e., hierarchical level plus any categories) and the DAC group, and the mechanism for the user's current operating privilege, restrict how a subject can access an object. These controls are strict and useful with respect to MAC and DAC. ORGCON, however, has additional controls and a different set of groups (ORGCON-groups vs. DAC-groups) that presented difficulties during incorporation into the existing structure.

Part of the difficulty was related to our attempt to strictly adhere to the GFAC concepts. It was desirable to implement ORGCON using data structures that clearly mirrored the GFAC concepts of ACI and ACC. The structures finally chosen were rationalized as practical compromises that do not violate the GFAC concepts nor the System V/MLS mechanisms.

In some cases, working on an existing secure system resulted in a less than ideal balance in terms of achieving the stated goals. GFAC provides a high-level informal model of access control in AISs (i.e., an abstraction). The restrictions of an actual system forced the sacrifice of implementation of the prototype strictly according to the GFAC concepts.

5.4 Implementing Based on the ORGCON Policy

This section discusses the actual implementation effort on the prototype to date (June 1991). Primarily, the discussion focuses on our current thinking with regard to best approaches for implementation of the controls to support the ORGCON policy. Some details and issues remain to be resolved.

5.4.1 Observations on Implementing Identity-Based Non-Discretionary Access Control

When formulating ideas for how to implement ORGCON, the discussion of the suitability of the O/G/W bit mask or similar mechanism arose repeatedly. Initially, there was reluctance to use these mechanisms because of the well-known weaknesses of traditional DAC mechanisms. The DAC access control list (ACL) and O/G/W mechanisms are useful, well known, and widely implemented. There is no apparent reason not to use them in implementing other policies. Confusion has sometimes resulted from the common identification of DAC mechanisms with DAC policy. The following discussion should aid in the clarification of the issue.

The particular category of controls we are interested in is the class of identity-based non-discretionary access controls, as exemplified by ORGCON. Traditional DAC mechanisms provide a weak form of need-to-know; the ORGCON policy requires a much stronger form of need-to-know. After considerable debate, we decided that the O/G/W mechanism can be an effective mechanism for identity-based access control, IF we also implemented supporting policy(ies) that closed the DAC weaknesses. Put another way, we designed the mechanisms to implement the ORGCON policy using the traditional DAC mechanisms in conjunction with other mechanisms. The uncontrolled copy and the lack of accountability weaknesses of DAC are limited by restricting user access to ORGCON objects to a limited set of functions.

A major difference between the requirements of the ORGCON policy and DAC mechanisms is delegation of authority. Under traditional DAC, authority to determine read and execute access to information is effectively given to anyone having read access to an object containing the information. Write access is somewhat more restricted. Under the ORGCON policy, the authority to grant read access is shared by two roles to whom authority is delegated. One role (the ORGREP) has the authority to change ACI associated with the object (i.e., the ACL). The other role (the RECREP) has the authority to change ACI which is part of the context (i.e., subject's group/role membership). This can be compared to mandatory controls wherein a single role (e.g., ISSO) or some agent such as the classification/clearance officer, changes ACI associated with the subject (clearance) and ACI associated with the object (classification).

The prototype implements two roles: the ORGREP and the RECREP. The prototype includes a "role" command, that allows users to assume a given role and limits their actions within that role. For example, a user wishing to act as the ORGREP would explicitly change role to that of ORGREP. Appropriate TCB checks are made to ensure that the user is authorized to act in the ORGREP role, and if so, the user, acting as ORGREP, is put in a restricted shell that limits the available commands to those necessary to perform the appropriate ORGREP functions (e.g., read ORGCON objects, add the distribution list, store the object, print the object). A similar role command is provided for the RECREP.

Since System V/MLS does not implement ACLs it was necessary to develop a strategy for providing an ACL. Initially we anticipated using available space in the label structure to implement an ACL by creating a pointer field to a linked list containing the list of recipient organization roles. However, this proved infeasible due to the implementation of labels in System V/MLS. A further issue was how to notify recipients of a new ORGCON object and how to deal with recipients on remote AISs. Both of these were solved by exploiting the multi-level secure mail facility provided by System V/MLS. The ACL was incorporated in the header of the message and mail mechanisms are used to distribute ORGCON objects and notify recipients.

The credibility, reliability, and trustworthiness of the DAC authority is rather low. Lack of accountability undoubtedly contributes to this low esteem. While DAC is supposed to be used to implement need-to-know policy, the DAC Guideline [6] points out that access could be granted based on "whom do I like." Under the ORGCON policy and MAC, access is controlled by a designated authority who is held accountable for his/her action. This authority is responsible for changing the appropriate ACI based on information, such as a person's clearance or an organization's roster, supplied by equally authorized and audited officials.

5.4.2 Additional ORGCON ACI

To support the ORGCON policy, several attributes were added to the object's ACI. The ORGCON marking was previously discussed. The attribute "ORGCON- distribution" is also part of an object's ACI. The distribution list is a set of recipient organizations (e.g., CINCPAC, Division X), serving as an access control list within the computer system(s) and a distribution list when hardcopy is obtained. The definition of these organizations and roles is part of the ACC (i.e., the context on the destination AIS). The attribute originator identification (orig-ID) is also maintained in the ACI. Only the ORGREP can populate the ORGCON distribution list and provide the orig-ID for an ORGCON object. In the prototype, all the ORGCON-related ACI are associated with the object or are ACC (i.e., there is no additional ACI associated with the subject).

5.4.3 Designating an Object ORGCON-C

One of the components of the implementation of the ORGCON policy is the *ORGCON DESIGNATE program*. This program achieves the first steps in limiting the dissemination and use of ORGCON information.

Any user can create a potential ORGCON object. However, once an object is designated ORGCON, the creator of the object no longer has the authority over that object. The originating representative is responsible for attaching the distribution list to the object (although the creator may provide a suggested distribution list) and distributing the object. The designate program handles this "passing" of authority from the creator to the originating representative.

The program provides a convenient interface to the user for "passing" an object to an ORGREP, and handles the details associated with the changes in authority, labeling the object, and the restricted access requirements of an ORGCON-C object. The designate program takes the specified file and performs the following actions:

- Marks the file "ORGCON-C" (ORGCON candidate)
- Changes the user (creator) permissions to <read>.
- Changes the groups permissions to <read>.
- Changes the other permissions to <null>.
- Changes the group to the ORGREPS group.
- Renames the file uniquely to prevent accidental overwrite.
- Moves the file to /usr/users/orgreps.

In the process, the creator retains read permission on the file so (s)he may still review it, and the originating representatives may read the file (as long as the user is operating at the same MAC classification level as that of the object). By changing the group and moving the file to /usr/users/orgreps, access is restricted to the owner and the orgreps; no one in the group that the creator belongs to still has access to the file. This begins the process of changing the markings and putting the additional controls on the object. At this point, the owner still has limited authority over the ORGCON-C object.

5.4.4 Designating an Object ORGCON

The next step, then, is for the ORGREP to change the designation of an object from ORGCON-C to ORGCON. Once an object is designated ORGCON, the creator of the object no longer has the authority over that object. The program again provides a convenient interface to the object, and handles the details associated with the changes in ownership, labeling, and the restricted access requirements of an ORGCON object. At this point the designate program takes the specified file and performs the following actions:

- Marks the file "ORGCON"
- Changes the ownership to "ORGREP".
- Changes the owner permissions to <read>.
- Changes the groups permissions to <read>.
- Changes the other permissions to <null>.

The ORGREP then initiates the DISTRIBUTE program. This program prompts the ORGREP for a distribution list and handles the actual distribution of the ORGCON object. It checks to verify that the organizations specified as recipients are valid organizations and handles the dissemination of the object to local and remote systems as indicated in configuration files. (Maintaining this authorized list of valid organizations is outside the scope of the prototype). This program likewise provides a convenient interface for the ORGREP to distribute an ORGCON object.

6 CONCLUSIONS

6.1 Concerning Security Policies

In this paper, a policy named ORGCON (Organization Controlled) that is based on the ORCON policy has been defined. Though a policy for manual control of paper documents can be workable even though vague or lacking detail, the policy must be extended and the detail must be specified to make it suitable for an AIS. By creating supporting policies and hypothesizing procedures, the ORCON policy was extended and details added to create the ORGCON policy. The ORGCON policy created permits copying for distribution but not for incorporation of information in derivative objects.

Since the prototype described in this paper was a proof-of-concept, conformance to real world constraints was not the highest priority though we understand that a real implementation would indeed have many such constraints. Experience suggests that most organizations do not understand their information flows, and that security restrictions exacerbate the

concerns. In the extreme, some organizations may decide not to automate certain security policies because the AIS will not have the ability to discern when the letter of the law may be ignored with impunity. However, we believe that it is both possible and desirable to implement additional security policies in an AIS and plan to implement other existing information dissemination/control policies.

6.2 Concerning Technology

This effort has demonstrated that it is possible to implement additional security policies by extending an existing TCB. Such an effort requires a well formulated approach such as the Generalized Framework for Access Control. Part of our approach has involved formal modeling, which proved invaluable in aiding our understanding of the policies. A new understanding of the increased level of detail required for modeling GFAC concepts will be reported in a subsequent paper.

As with all research, additional questions surfaced while several others were answered. In particular, the potential growth in size and complexity of the TCB if the mechanisms for implementing all of the security policies are placed in the same TCB remains an issue. Exploration of the relationships among TCB mechanisms supporting separate policies is required. For example, the TCB code that implements the ORGCON controls has no relationship to MAC or DAC policy. Part of the problem is determining appropriate terminology to express the concepts. What words should be used to refer to mechanisms that implement different policies? Can and should the TCB concept be expanded to embrace additional policies? What should be the relationships among TCBs for different policies? It is anticipated that answers to at least some of these questions will be discovered in the coming year.

LIST OF REFERENCES

1. Abrams, M. D., K. W. Eggers, L. J. LaPadula, I. M. Olson, "A Generalized Framework for Access Control: An Informal Description," *Proceedings of the 13th National Computer Security Conference*, October 1990.
2. Director of Central Intelligence Directive No. 1/7, *Control of Dissemination of Intelligence Information*, 4 May 1981.
3. Graubart, T. D., "On the Need for a Third Form of Access Control," *Proceedings of the 12th National Computer Security Conference*, Baltimore, MD, October 1989.
4. LaPadula, L. J., "Formal Modeling in a Generalized Framework for Access Control," *Proceedings of the Computer Security Foundation Workshop III*, 12 June 1990.
5. National Computer Security Center, *Department of Defense Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD, December 1985.
6. National Computer Security Center, *A Guide to Understanding Discretionary Access Control in Trusted Systems*, NCSC-TG-003, Version-1, 30 September 1987.
7. National Computer Security Center, *Final Evaluation Report of American Telephone and Telegraph System V/MLS Release 1.1.2 Running on UNIX System V Release 3.1.1*, CSC-EPL-89/003, 18 October 1989.
8. Williams, J. C. and M. L. Day, "Sensitivity Labels and Security Profiles," *Proceedings of the 11th National Computer Security Conference*, Baltimore, MD, October 1988.
9. AT&T, "System V/MLS Users' Guide and Reference Manual," 27 March 1990.

HONEST DATABASES THAT CAN KEEP SECRETS

*Ravi S. Sandhu and Sushil Jajodia**

Center for Secure Information Systems
and

Department of Information and Software Systems Engineering
George Mason University, Fairfax, VA 22030-4444

ABSTRACT Polyinstantiation has generated a great deal of controversy lately. Some have argued that polyinstantiation and integrity are fundamentally incompatible, and have proposed alternatives to polyinstantiation. Others have argued about the correct definition of polyinstantiation and its operational semantics. In this paper we provide a fresh analysis of the basic problem that we are trying to solve, i.e., how can a honest database keep secrets? Our analysis leads us to the concept of restricted polyinstantiation wherein we show how to solve this problem without compromising on any of the following requirements: secrecy, integrity, availability-of-service, element-level labeling and high assurance. This is the first solution to meet all these requirements simultaneously.

1 INTRODUCTION

What distinguishes a multilevel database from ordinary single level ones? In a multilevel world as we raise a user's clearance new facts emerge; conversely as we lower a user's clearance some facts get hidden. Therefore users with different clearances see different versions of reality. Moreover, these different versions must be kept coherent and consistent—both individually and relative to each other—without introducing any downward signaling channels.[†]

The caveat of “no downward signaling channels” poses a major new problem in building multilevel secure database management systems (DBMSs) as compared to ordinary single-level DBMSs. This caveat is inescapable and absolute. We must reject outright “solutions” which tolerate downward signaling channels. Solutions with such channels, e.g., as proposed in [1, 9], may well be acceptable as an engineering compromise in particular situations. But they are clearly not acceptable as general-purpose solutions. This point needs to be emphasized because security is usually the one to take the first hit in engineering trade-offs. It behooves us as security researchers to present solutions which avoid taking this hit while at the same time providing

- no downward signaling channels,
- consistency and integrity of the database both within and across levels,
- flexibility for application semantics,
- fine-grained classification of data (i.e. element-level labeling), and
- high assurance with minimal trusted code.

*The work of both authors was partially supported by the U.S. Air Force, Rome Air Development Center through subcontract #C/UB-49;D.O.No.0042 of prime contract #F-30602-88-D-0026, Task B-O-3610 with CALSPAN-UB Research Center.

[†]We deliberately use the term downward signaling channel rather than covert channel. A downward signaling channel is a means of downward information flow which is inherent in the data model and will therefore occur in every implementation of the model. A covert channel on the other hand is a property of a specific implementation and not a property of the data model. In other words, even if the data model is free of downward signaling channels, a specific implementation may well contain covert channels due to implementation quirks.

© Ravi S. Sandhu and Sushil Jajodia, 1991

The central point of this paper is to demonstrate how these diverse goals can be met in a multilevel relational DBMS without compromising security as part of the bargain. Our solution is simple in concept and almost obvious in retrospect. For the most part it uses standard concepts from the database arena. A key new idea is to introduce a special value called "restricted" distinct from the normal data values of an attribute (or column) as well as distinct from "null." The value "restricted" denotes that the particular field cannot be updated at the specified level. So long as the value of a field is not "restricted" our multilevel relations behave much as ordinary single-level relations do. Particular attention is required when a field is changed from unrestricted to restricted and vice versa. A notable property of our solution is that it can be implemented entirely by untrusted subjects, i.e., subjects which are not exempted from the simple security or \star -properties.¹

The rest of this paper is organised as follows. Section 2 reviews the concept of polyinstantiation from an intuitive point of view, with the objective of identifying the sources of polyinstantiation and alternatives to it. Section 3 informally introduces our solution of restricted polyinstantiation and illustrates it by examples. Section 4 formalises and precisely defines our solution. It also provides additional examples. Section 5 discusses how our solution can provide the highest degree of assurance. Section 6 concludes the paper.

2 POLYINSTANTIATION

The concept of polyinstantiation was explicitly introduced by Denning et al [3] in connection with the SeaView project. Since then much has been written about this topic [1, 3, 4, 5, 6, 7, 9, for instance]. In this paper we will set aside all this previous theory, formalism and debate. Instead we go back to first principles and consider by means of examples how polyinstantiation arises and therefore how it might be controlled. We assume the reader is familiar with basic relational notions and terminology.

2.1 The Source of Polyinstantiation

Polyinstantiation can occur in basically two different ways which we call *polyhigh* and *polylow* respectively for mnemonic convenience.

1. Polyhigh occurs when a high user² attempts to insert data in a field which already contains low data. Overwriting the low data in place will result in a downward signaling channel. Therefore the high data can be inserted only by creating a new instance of the field to store the high data. We also have the option of rejecting the update altogether with the attendant possibility of denial-of-service to the high user.
2. Polylow occurs in the opposite situation where a low user attempts to insert data in a field which already contains high data. In this case rejecting the update is not a viable option because it establishes a downward signaling channel. That leaves us two alternatives. We can overwrite the high data in place which violates the integrity of the high data. Or we can create a new instance of the field to store the low data.

In both cases note that we have identified "secure" alternatives to polyinstantiation. These alternatives are secure in the sense of secrecy and information flow. Unfortunately the alternatives have denial-of-service and integrity problems reiterated below.

¹The protocols of section 4 can be simplified if trusted subjects which are exempted from these properties are allowed in selected situations.

²Strictly speaking we should be saying subject rather than user. For the most part we will loosely use these terms interchangeably. Where the distinction is important we will be appropriately precise.

1. The alternative to polyhigh entails denial-of-service to high users by low users (i.e., once a low value has been entered in a field a high value cannot be entered until the low value has been nullified by a low subject[†]).
2. The alternative to polylow entails destruction of high data by low users which presents a serious integrity problem (i.e., the high data is overwritten in place by low data.)

A naive implementation of these alternatives will create more real security problems than it solves. Our main contribution in this paper is to show how these alternatives to polyhigh and polylow can be employed in a careful, disciplined manner to achieve secrecy, availability-of-service and integrity with high assurance.

It should be noted that there is an important difference between polyhigh and polylow. Polyhigh can be completely prevented by reactive mechanisms at the cost of denial-of-service to entry of high data. This is likely to be a tolerable cost in many applications. On the other hand polylow cannot be completely prevented by reactive mechanisms. At the moment of enforcement a reactive mechanism has only the alternative of overwriting high data by low data. This is likely to be intolerable in most applications. Therefore polylow must—for all practical purposes—be prevented by a proactive mechanism, i.e., steps must be taken in advance of the problem's occurrence to ensure that it cannot occur.

2.2 Polyhigh Example

Let us now consider a concrete example to make polyhigh and polylow clearer. Consider the following relation SOD where Starship is the apparent primary key.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	null U	U

Here, as in all our examples, each attribute in a tuple not only has a value but also a classification. In addition there is a tuple-class or TC attribute. This attribute is computed to be the least upper bound of the classifications of the individual data elements in the tuple.

Now consider the following scenario.

1. A U user updates the destination of the Enterprise to be Talos. The relation is therefore modified as follows.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Talos U	U

2. Next a S user attempts to modify the destination of the Enterprise to be Rigel. We cannot overwrite the destination in place because that would create a downward signaling channel. We can reject the update at the risk of denying entry of legitimate secret data. Or we can polyinstantiate and modify the relation to appear as follows, respectively for U and S users. Note that U users see no change.

[†]This protocol—of nullifying low data prior to entry of high data—does not guarantee protection against denial-of-service. If a low value is nullified to enable entry of a high value there remains the risk that a low Trojan Horse can enter another low data value before the high subject has the opportunity to enter its high value. The solution described in this paper (see Section 3) eliminates this vulnerability.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Talos U	U

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Talos U	U
Enterprise U	Exploration U	Rigel S	S

What are we to make of this last relation given above. There are at least two reasonable interpretations.

- *Cover Story.* The destination of Talos may be a cover story for the real destination of Rigel. In this case the database is accurately mimicking the duplicity of the real world. There are, however, other ways of incorporating cover stories besides polyinstantiation. For example we may have two attributes, one for cover-story destination and one for the real destination. Debate on the relative merits and demerits of these techniques is outside the scope of this paper. *For purpose of this paper we assume that polyinstantiation is not to be used for cover stories. We therefore reject this alternative as a valid interpretation.*
- *Temporary Inconsistency.* We have a temporary inconsistency in the database which needs to be resolved. For instance the inconsistency may be resolved as follows: the S user who inserted the Rigel destination latter logs in at the U level and nullifies the Talos value, so thereafter the relation appears respectively as follows to U and S users.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	null U	U

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Rigel S	S

It is most important to understand that this scheme does not create a downward signaling channel from one subject to another. The nullification of the destination at the U level is being done by a U subject. One might argue that there is a downward signaling channel with a human in the loop. The human is however trusted not to let the channel be exercised without good cause. Finally note that the U user who executed step 1 of the scenario may again try to enter Talos as the destination, which brings us within the scope of polylow.

2.3 Polylow Example

Our example for polylow is similar to the polyhigh example with the difference that the two update operations occur in the opposite order. So again consider the following relation SOD where Starship is the apparent primary key.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	null U	U

This time consider the following scenario.

1. A S user modifies the destination of the Enterprise to be Rigel. The relation is modified to appear respectively as follows to U and S users. Note that U users see no change in the relation.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	null U	U

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Rigel S	S

2. A U user updates the destination of the Enterprise to be Talos. We cannot reject this update on the grounds that a secret destination for the Enterprise already exists, because that amounts to establishing a downward signaling channel. We can overwrite the destination field in place at the cost of destroying secret data. This would give us the following relation for both U and S users.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Talos U	U

For obvious reasons this alternative has not been seriously considered by most researchers. That leaves us the option of polyinstantiation which will modify the relation at the end of step 1 to the following for U and S users respectively.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Talos U	U

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Talos U	U
Enterprise U	Exploration U	Rigel S	S

This is exactly the same relation as obtained at the end of step 2 in our polyhigh example. The possible interpretations are therefore similar, i.e., we either have a temporary inconsistency or a cover story (the latter alternative has already been rejected for our database). The temporary inconsistency can be corrected by having a U subject (possibly created by a S user logged in at the U level) nullify the Talos destination. But the inconsistency may recur again and again.

3 RESTRICTED POLYINSTANTIATION

In the previous section we have examined the source of polyinstantiation and identified polyhigh and polylow as the two different ways in which polyinstantiation arises. In this section we consider applications which have the following requirements.

1. Downward signaling channels cannot be tolerated.
2. The simple security and \star -properties must be enforced for all subjects, i.e., no trusted code can be used.
3. Temporary inconsistencies cannot be tolerated.
4. Denial of data entry service to high users cannot be tolerated.

Moreover each of these requirements has equal importance and one cannot be sacrificed for another. The scenarios of the polyhigh and polylow examples of the previous section show that polyinstantiation by itself cannot meet these requirements simultaneously. One requirement or the other must give in some way.

In this section we show how all four requirements identified above can be simultaneously met. We describe our solution as *restricted polyinstantiation*. The basic idea is to introduce a special symbol denoted by "restricted" as the possible value of a data element. The value "restricted" is distinct from any other value for that element and is also different from "null." In other words the domain of a data element is its natural domain extended with "restricted" and "null." We define the semantics of "restricted" in such a way that we are able to eliminate both polyhigh and polylow. "Null" has exactly the same semantics as any other data value and needs no special treatment.

Let us now play out the polyhigh and polylow scenarios of the previous section to intuitively motivate our solution. A formal description of the update protocols is given in the next section.

3.1 Polyhigh Example Revisited

Consider again the following relation SOD where Starship is the apparent primary key.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	null U	U

Now consider the following scenario.

1. A U user updates the destination of the Enterprise to be Talos. The relation is therefore modified as follows.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Talos U	U

2. Next a S user attempts to modify the destination of the Enterprise to be Rigel. We cannot polyinstantiate even temporarily, so we must reject this update. Do we have denial-of-service to the S user? No, because the S user can obtain service as follows.

Step 2a. The S user first logs in as a U-subject and marks the destination of the Enterprise as restricted giving us the following relation.[‡]

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted U	U

The meaning of restricted is that this field can no longer be updated by a U user. U users can therefore infer that the true value of Enterprise's destination is classified at some level not dominated by U.

Step 2b. The S user then logs in as a S-subject and enters the destination of the Enterprise as Rigel giving us the following relations at the U and S levels respectively.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Rigel S	S

[‡] Alternately the S user logs in at the U-level and requests some properly authorised U user to carry out this step. Communication of this request from the S user to the U user may also occur outside of the computer system, by say direct personal communication or a secure telephone call.

How does this differ from the scenario of section 2.2 (where the end result after cleaning up the temporary inconsistency was as above except that we have null instead of restricted)? The main difference is that, after step 2a, U users are no longer able to update the destination of the Enterprise. In particular, attempts by U users to reenter Talos as the destination of Enterprise will be rejected on the grounds that the field is restricted. Therefore the relation is guaranteed to be consistent till such time as the restricted value is eliminated. Consideration of who should be allowed to enter and remove the restricted value is deferred for now.

Does step 2a introduce a signaling channel? Yes, but this signaling channel is very similar to the one resulting from the nullification of Talos at the U-level in the example of section 2.2. Both involve a trusted S user in the loop who presumably will ensure that the channel is not exercised wantonly, but rather that this inference is permitted only when the real world situation is actually so. Such a channel with trusted humans in the loop can be exercised only by Trojan Horses that are capable of manipulating the real world. This entails the manipulation of real trusted people making real decisions and not merely the manipulation of bits in a database.

3.2 Polyflow Example Revisited

Now consider the two update operations in the opposite order. So again we begin with the following relation SOD where Starship is the apparent primary key.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	null U	U

This time consider the following scenario.

1. A S user modifies the destination of the Enterprise to be Rigel. This update is rejected! Instead the S user is asked to go through steps 2a and 2b of section 3.1 giving us the following relations at the U and S levels respectively.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Rigel S	S

2. A U user updates the destination of the Enterprise to be Talos. The update is rejected on the grounds that the field is restricted.

Note that there is no denial-of-service to the S user. What is happening is a denial of improper service, i.e., there is a protocol for entering high data which all S users are required to follow. Failure to follow the protocol results in denial-of-service but this can hardly be considered a security breach. The denial-of-service to the U user is, of course, only appropriate in this situation.

There is a crucial difference between this protocol and the one discussed in section 2.1. In both cases entry of high data is enabled by an action of a low subject. Our protocol requires the low subject to enter the "restricted" value in the data element. In section 2.1 the suggestion was for the low subject to enter a "null" value. The key difference in the two cases is that a null value can be made non-null by a low Trojan Horse, whereas the restricted value cannot be made unrestricted by a low Trojan Horse. The latter operation requires a special privilege whose distribution is carefully controlled by non-discretionary means. This privilege is available only to selected low subjects who are trusted to exercise its use properly.

4 THE PREVENT PROTOCOLS

In this section we precisely define the collection of update protocols illustrated by example in the previous section. We collectively call this collection the *prevent protocols* because they prevent polyinstantiation due to either polyhigh or polylow from occurring. These protocols can be implemented entirely by untrusted subjects, i.e., subjects which are not exempted from the simple security or \star -properties.

4.1 Multilevel Relations

We begin by reviewing some basic concepts and notation for multilevel relations. Let $A_1, C_1, A_2, C_2, \dots, A_n, C_n$ denote the attributes (columns) of a multilevel relation R with element level labeling. Each A_i is a *data attribute* and each C_i is the *classification attribute* for A_i . A data attribute can take on values from its natural domain D_i extended with two special values, "null" and "restricted," whose meaning will be defined shortly. We assume that each C_i can take on any value c in the security lattice.** We require that C_i cannot be null. Finally R has a collection of *relation instances* R_c one for each access class c in the given lattice.

Assume there is a user-specified primary key AK consisting of a subset of the data attributes A_i . We call AK the *apparent primary key* of the multilevel relation scheme. In general AK will consist of multiple attributes. We have the following requirement in analogy to entity integrity in the standard relation model. (The notation $t[A_i]$ denotes the value of the A_i attribute in tuple t , and similarly for $t[C_i]$.)

Property 1 [Entity Integrity] Instance R_c of R satisfies entity integrity iff for all $t \in R_c$: (i) AK is uniformly classified in each tuple, i.e., $A_i, A_j \in AK \Rightarrow t[C_i] = t[C_j]$, and (ii) the classification of each non-key data attribute dominates the classification of the apparent key, i.e., $A_i \notin AK \Rightarrow t[C_i] \geq t[C_{AK}]$ where C_{AK} is the classification of AK . \square

The notions introduced thus far are standard ones first introduced in the SeaView model [7]. Our next requirement severely limits polyinstantiation and distinguishes the approach of this paper from previous work on element-level labeling (such as [3, 4, 5, 6, 7]).

Property 2 [Key Integrity] R satisfies key integrity iff for every R_c we have for all $i: AK, C_{AK} \rightarrow A_i, C_i$. \square

This property stipulates that the user-specified apparent key AK , in conjunction with key-classification C_{AK} , functionally determines all other attributes. In other words R_c cannot have more than one tuple for a given combination of values for AK and C_{AK} . That is, the real primary key of the relation is AK, C_{AK} . The effect of key integrity is to rule out instances such as the following.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U
Enterprise	U	Exploration	U	Rigel	S	S

The reason for rejecting this instance is its inconsistency in specifying two different destinations—one secret and one unclassified—for the Enterprise. Recall our assumption that cover stories are not to be incorporated by polyinstantiation, so interpretations such as discussed in [5] do not apply in this situation. Key integrity does allow instances such as the following where there is polyinstantiation of the key.

**In practice of course it is desirable to place appropriate upper and lower bounds on each C_i . This will only require minor changes to the following discussion.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U
Enterprise	S	Spying	S	Rigel	S	S

In this case we interpret the two tuples as describing two distinct Starships which happen to have the same name.

The next property is concerned with consistency between relation instances at different access classes. Here again we depart from the analogous property defined in [5, 6, 7].^{††}

Property 3 [Inter-Instance Integrity] R satisfies inter-instance integrity iff for all $c' \leq c$ we have $R_{c'} = \sigma(R_c, c')$ where the *filter function* σ produces the c' -instance $R_{c'}$ from R_c as follows:

1. For every tuple $t \in R_c$ such that $t[C_{AK}] \leq c'$ there is a tuple $t' \in R_{c'}$ with $t'[AK, C_{AK}] = t[AK, C_{AK}]$ and for $A_i \notin AK$

$$t'[A_i, C_i] = \begin{cases} t[A_i, C_i] & \text{if } t[C_i] \leq c' \\ \langle \text{restricted}, c' \rangle & \text{otherwise} \end{cases}$$

2. There are no tuples in $R_{c'}$ other than those derived by the above rule. □

The filter function maps a multilevel relation to different instances, one for each descending access class in the security lattice. Filtering limits each user to that portion of the multilevel relation for which he or she is cleared. For instance filtering the following S-instance of SOD

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Rigel	S	S

gives us the following U-instance

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	U	U

4.2 Update Protocols

In section 4.1 we have identified integrity properties for multilevel relations considered at some instant in time as static objects. We now consider the dynamic behavior of these relations by considering their update semantics. We emphasize that our protocols do not require any exception from the simple security or \star -properties.^{††} There are three subcases to consider as follows.

4.2.1 Data Value Update

By the term *data value* we mean any value other than "restricted." Our first protocol addresses the case where the value of attribute $t[A_i]$ is changed from its previous data value to a new data value, i.e., neither the previous value nor the new one can be "restricted." "Null" does not need any special treatment in our protocols and is viewed as just another data value. We have the following update protocol.

^{††}The definition of the filter function given in [5, 6, 7] differs from the one given here in that $\langle \text{restricted}, c' \rangle$ is replaced by $\langle \text{null}, t[C_{AK}] \rangle$.

^{††}Note that the protocols can be simplified if trusted subjects which are exempted from these properties are allowed in selected situations. In particular the protocol to change a restricted value to unrestricted (see section 4.2.3) would be considerably simplified by using a trusted subject which is exempted from the \star -property.

Protocol 1 $t[A_i]$ can be changed from its previous data value to a new data value by a c -user only if $t[C_i] = c$.

The effect of this update operation is defined as follows.

1. The value of $t[A_i]$ is changed to its new value in all relation instances $R_{c'}$, $c' \geq c$. The value of $t[C_i]$ remains unchanged as c in all $R_{c'}$, $c' \geq c$.
2. All other instances of R remain unchanged. □

Note that the precondition for this protocol is stated as a necessary condition ("only if"). It is thus a mandatory requirement. In addition to this mandatory pre-condition we may as usual impose further mandatory and/or discretionary controls.

To illustrate the protocol consider the following U and S instances of SOD respectively.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Rigel S	S

An update by a U user to change the Objective from "Exploration" to "Mining" has the following effect.

Starship	Objective	Destination	TC
Enterprise U	Mining U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Mining U	Rigel S	S

That is the update takes effect at both the U and S levels. An attempt by a S user to change the Objective attribute would be rejected. So would an attempt by a U user to change the Destination attribute. A S user may change the Destination attribute to say "Talos" giving us the following U and S instances of SOD respectively.

Starship	Objective	Destination	TC
Enterprise U	Mining U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Mining U	Talos S	S

To appreciate how "null" is treated just like any other data value consider what happens if a S user nullifies the Destination attribute. We get the following U and S instances of SOD respectively.

Starship	Objective	Destination	TC
Enterprise U	Mining U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Mining U	null S	S

The Destination attribute remains restricted for U users and the null value is shown only to S users. The classification of the null at S signifies that data in this field can only be entered by S users. If

the Destination attribute has a null value at the U level then both U and S instances of SOD must be as follows. -

Starship	Objective	Destination	TC
Enterprise U	Mining U	null U	U

In this case U users are allowed to enter data for the Destination attribute whereas S users are not permitted to do so. In order to enable S users to change the Destination of the Enterprise we must first restrict this field at the U level. This brings us to our next protocol.

4.2.2 Update from Unrestricted to Restricted

Let us first consider the case where the security lattice is totally ordered (i.e., there are no compartments). An update of attribute A_i in tuple t from some existing data value to "restricted" is performed as follows.

Protocol 2 $t[A_i]$ can be changed from its previous data value to "restricted" by a c -user only if $t[C_i] = c$.

The effect of this update operation is defined as follows.

1. The value of $t[A_i, C_i]$ is changed to $\langle \text{restricted}, c \rangle$ in the instance R_c .
2. Let $\pi(c)$ be the immediate predecessor of c (i.e., $\pi(c) > c$ and there is no c' such that $\pi(c) > c' > c$). The value of $t[A_i, C_i]$ is changed to $\langle \text{null}, \pi(c) \rangle$ in all instances $R_{c'}, c' > c$.
3. All other instances of R remain unchanged. □

It suffices to have the pre-condition $t[C_i] = c$ for this operation because, in conjunction with the inter-instance integrity property, $t[C_i] = c$ implies

$$(\forall c' : t[C_{AK}] \leq c' < c) \ t[A_i, C_i] = \langle \text{restricted}, c' \rangle \text{ in } R_{c'}$$

In other words a data element can be made restricted at level c only if its data value is currently classified at level c , which in turn implies that the data element is restricted at all relevant levels below c .

To illustrate the effect of such updates consider the following U instance of SOD (which is identical to the S instance).

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Rigel U	U

A U user can change the destination of the Enterprise to be "restricted" giving us the following U and S instances.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Exploration U	null S	S

Now let us consider the general case of a partially ordered security lattice. The problem with partially ordered labels lies in step 2 in defining the effect of protocol 2. In a partial ordering there

may be multiple immediate predecessors of c so $\pi(c)$ is no longer uniquely defined. As part of the update operation we have to designate one of c 's immediate predecessors as the distinguished one which will remain unrestricted. All other immediate predecessors become restricted. Let $\pi(c)$ denote the distinguished immediate predecessor. Step 2 of protocol 2 needs to be restated as follows.

2'. The value of $t[A_i, C_i]$ is changed as follows for all instances $R_{c'}, c' > c$.

$$t[A_i, C_i] = \begin{cases} \langle \text{null}, \pi(c) \rangle & \text{if } c' \geq \pi(c) \\ \langle \text{restricted}, c' \rangle & \text{if } c' \not\geq \pi(c) \end{cases}$$

As an example consider a lattice with four labels, S, U, M_1 and M_2 ; where M_1 and M_2 are both dominated by S and both dominate U, but M_1 and M_2 are themselves incomparable. Suppose we have the following instance of SOD at all four levels.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Rigel U	U

Let a U user make the Destination field of the Enterprise "restricted" while designating M_1 to be $\pi(U)$ for this update. The U, M_1 , M_2 and S instances of SOD will respectively become as follows.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Exploration U	null M_1	M_1

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted M_2	M_2

Starship	Objective	Destination	TC
Enterprise U	Exploration U	null M_1	M_1

4.2.3 Update from Restricted to Unrestricted

Again for simplicity let us first consider the case where the lattice is totally ordered. We have the following protocol for making a field unrestricted.

Protocol 3 $t[A_i]$ can be changed from its current value of "restricted" to a data value dv only by a c -user.

The effect of this update operation is defined as follows.

1. The value of $t[A_i, C_i]$ is changed to $\langle dv, c \rangle$ in all instances $R_{c'}, c' \geq c$.
2. All other instances of R remain unchanged. □

The pre-condition for this update, that $t[A_i, C_i] = \langle \text{restricted}, c \rangle$ in R_c , is sufficient to ensure that $t[A_i, C_i] = \langle \text{restricted}, c' \rangle$ in all $R_{c'}, c' \leq c$ (due to inter-instance integrity).

The protocol will overwrite any existing data value for $t[A_i]$ in instances $R_{c'}, c' > c$. This operation therefore has the potential for creating integrity problems by overwriting existing higher level data. We have rejected this approach as a general solution in section 2. Here we are proposing

to employ it for the specific purpose of converting a field from restricted to unrestricted. We require that this be a specially privileged operation so that we can be sure it is executed only when the real world conditions warrant it. We will return to this point in the next section.

To illustrate this operation consider the following U and S instances of SOD.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Exploration U	null S	S

A suitably privileged U user can change the value of the Destination attribute in this tuple to be say "Talos" giving us the following (identical) U and S instances of SOD.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Talos U	U

Next let us consider the case of a partially ordered security lattice. The pre-condition of protocol 3 is no longer sufficient. Before a c user is allowed to change a restricted field to non-restricted we must ensure that field is restricted at all levels which do not dominate c . This includes levels which are dominated by c as well as levels incomparable with c . The latter requirement cannot be checked by a c user without violating simple-security. We circumvent this problem by requiring the update of protocol 3 to occur in two phases as follows.

1. *Preparatory Phase.* Login at level $t[C_{AK}]$ and set

$$t[A_i, C_i] = \langle \text{restricted}, c' \rangle \text{ in all instances } R_{c'}, c' \geq t[C_{AK}]$$

i.e., set $t[A_i]$ to "restricted" at all levels where tuple t is visible.

2. *Update Phase.* Login at level c and set $t[A_i, C_i] = \langle dv, c \rangle$.

The net effect of this modified protocol is to set

$$t[A_i, C_i] = \begin{cases} \langle dv, c \rangle & \text{in all instances } R_{c'}, c' \geq c \\ \langle \text{restricted}, c' \rangle & \text{in all instances } R_{c'}, c' \not\geq c \end{cases}$$

For example consider the following U, M_1 , M_2 and S instances of SOD respectively taken from the end of section 4.2.2.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Exploration U	null M_1	M_1

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted M_2	M_2

Starship	Objective	Destination	TC
Enterprise U	Exploration U	null M_1	M_1

The preparatory phase will give us the following U, M₁, M₂ and S instances of SOD respectively.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted M ₁	M ₁

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted M ₂	M ₂

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted M ₂	M ₂

In other words the preparatory phase restricts the Destination attribute of this tuple at all levels above U (which is the key class of the tuple). Subsequently, the update phase results in (say) the following U, M₁, M₂ and S instances of SOD respectively.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted M ₁	M ₁

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Rigel M ₂	M ₂

Starship	Objective	Destination	TC
Enterprise U	Exploration U	Rigel M ₂	M ₂

5 ASSURANCE

In this section we briefly consider how the prevent protocols can be enforced.

Our first observation is that all our protocols adhere to both simple security and the \star -property. They can therefore be enforced by a DBMS trusted computing base (TCB) to the highest assurance standards without the use of subjects which are exempt from simple-security or the \star -property.

Secondly, our protocols are designed to achieve integrity and availability-of-service in addition to secrecy. The secrecy objective can be enforced to A1 standards by strict enforcement of simple security and the \star -properties. In order to achieve the integrity and availability of service requirements we need controls beyond the traditional simple security and \star -property. Let us consider each of the following three cases in turn.

5.1 Data Value Update

This is the simplest case where our multilevel relations behave much as conventional single-level relations do. It is obvious that in a high integrity system updates must be carefully controlled even within a single security level. Conventional databases use mechanisms such as well-formed transactions and least privilege for this purpose [2, 8]. The DBMS TCB must provide high assurance

support for such mechanisms. We do not need any additional mechanisms for multilevel DBMSs. The required mechanisms should anyway be available in high-quality single-level DBMSs as discussed in [8].

5.2 Update from Unrestricted to Restricted

Assigning a restricted value to a field with classification c requires a check that this field is already restricted at levels below c . This is feasible within the scope of simple security. In high assurance systems this application-independent pre-condition should be checked by the DBMS TCB. At lower levels of assurance the pre-condition may be tested by individual transactions rather than the DBMS.

The effect of restricting a field at the c level is dangerous in that it can cause denial-of-service to c users. So when the destinations of all our flights are made restricted, when they should not be, we might end up grounding the entire fleet! Therefore the ability to mark a field as restricted should be a carefully controlled privilege. This privilege should be assigned to a few subjects who need to do this operation. We can ensure that this privilege cannot be acquired except by some very special non-discretionary means such as involving intervention by a security officer.

The general problem of incorrect data essentially exists whether or not we recognize restricted as a special value. For suppose a malicious program running at the U level, and obeying simple security and \star -property, sets the destination of all flights to be Dayton, Ohio. Does the entire fleet converge on Wright Patterson Air Force Base? Presumably a high integrity system has corrective measures to detect and recover from such errors. In principle, incorrectly restricted fields present a similar problem except that recovery may be slightly more cumbersome.

5.3 Update from Restricted to Unrestricted

An update from restricted to unrestricted is different from the previous two cases because we cannot test the pre-conditions for this action within the confines of simple security. If we wish to prevent overwriting of high data by this operation we have to check that no high data exists (i.e., no non-null high data exists). In view of simple security this is not feasible. Therefore we define the operation as potentially overwriting high data. It follows that we must strictly control the ability to make a restricted value unrestricted. The control in this case should be even stricter than in the case of update from unrestricted to restricted. Alternately, we can use a trusted subject for this operation.

6 CONCLUSION

In this paper we have shown how both the polyhigh and polylow variations of polyinstantiation can be eliminated by our solution of restricted polyinstantiation. This allows us to avoid downward signaling channels, inconsistencies, denial of data entry to high users and the overwriting of high data by low subjects while providing element-level labeling. This is the first solution to meet all these requirements simultaneously.

In conclusion we wish to note that restricted polyinstantiation makes a particular trade-off among conflicting objectives. It may be eminently suitable to most applications. Yet we would advise against having this as the only option. Databases are long lived and develop a great deal of inertia over their life. Moreover different applications may call for different trade-offs. For example temporary inconsistencies may be preferred to inconvenience in data entry. General-purpose multilevel secure DBMSs must cater to such applications too. Therefore our recommendation is that restricted polyinstantiation be available as one of several options that a multilevel secure DBMS supports.

Acknowledgment

We are indebted to John Campbell, Joe Giordano, and Howard Stainer for their support and encouragement, making this work possible. The opinions expressed in this paper are of course our own and should not be taken to represent the views of these individuals.

References

- [1] Burns, R.K. "Referential Secrecy." *IEEE Symposium on Security and Privacy*, Oakland, California, May 1990, 133-142.
- [2] Clark, D.D. and Wilson, D.R. "A Comparison of Commercial and Military Computer Security Policies." *IEEE Symposium on Security and Privacy*, 184-194 (1987).
- [3] Denning, D.E., Lunt, T.F., Schell, R.R., Heckman, M., and Shockley, W.R. "A Multilevel Relational Data Model." *IEEE Symposium on Security and Privacy*, 220-234 (1987).
- [4] Denning D.E. "Lessons Learned from Modeling a Secure Multilevel Relational Database System." In *Database Security: Status and Prospects*, (Landwehr, C.E., editor), North-Holland, 35-43 (1988).
- [5] Jajodia, S. and Sandhu, R.S. "Polyinstantiation Integrity in Multilevel Relations." *IEEE Symposium on Security and Privacy*, Oakland, California, May 1990, 104-115.
- [6] Jajodia, S., Sandhu, R.S. and Sibley, E. "Update Semantics for Multilevel Relations." *Sixth Annual Computer Security Applications Conference*, Tucson, Arizona, December 1990, pages 103-112.
- [7] Lunt, T.F., Denning, D.E., Schell, R.R., Heckman, M. and Shockley, W.R. "The SeaView Security Model." *IEEE Transactions on Software Engineering*, 16(6):593-607 (1990).
- [8] Sandhu, R.S. and Jajodia, S. "Integrity Mechanisms in Database Management Systems." *13th NIST-NCSC National Computer Security Conference*, Washington, D.C., October 1990, 526-540.
- [9] Wiseman, S.R. "On the Problem of Security in Data Bases." In *Database Security III: Status and Prospects*, (Spooner, D.L. and Landwehr, C.E., editors), North-Holland, pages 143-150 (1990). Also available as Royal Signal and Radar Establishment, U.K., Memo 4263.

IDENTIFYING AND CONTROLLING UNDESIRABLE PROGRAM BEHAVIORS

Maria M. King*
MKing@Dockmaster.ncsc.mil

Abstract

This paper describes a new mechanism for comparing selected program properties against a policy, or set of rules, that states allowable program behavior[2, 10]. The motivation for this work is the increased need to control undesirable behaviors of programs, such as those inherent in Trojan horses and computer viruses. This mechanism, called an Automatic Policy Checker (APC), is currently implemented under SunOS¹. This paper will discuss the design and implementation of the APC and the application of the APC to the virus problem. Conclusions concerning anti-viral policy in light of the test results will also be presented.

Introduction

The motivation for this work is the increased need for computer security mechanisms to control undesirable activity of programs, such as those caused by computer viruses[1], Trojan horses and other types of malicious logic.

The major contribution of this work is an automatic tool, called an Automatic Policy Checker (APC), for comparing certain types of program behaviors against a policy that states allowable program behaviors. An important feature of the APC is that it does not implement any specific policy, clearly separating the policy from the mechanism which enforces the policy[8]. Existing mechanisms either rely on the user to specify their own policy[7] or embed an ad hoc policy in the mechanism[5]. The APC allows experiments with policies intended to prohibit a variety of undesirable program behaviors. The APC does not rely on any new architectural support, has minimal effect on performance, and does not require user knowledge of threat. Furthermore, if the APC is used in conjunction with a filter mechanism as described in [2, 6], reliance on some number of humans to act in a trustworthy manner, which is often required in many computer security mechanisms, is no longer needed.

This paper first describes a formal language based on regular expressions that was developed for stating policies and certain types of program behaviors. A high-level overview of the design of the APC is described here while [10] provides a more detailed discussion. The APC has been applied to the computer virus problem. A study of anti-viral policies based on the viral property of *file modification* was conducted and is described in the section on policies. Experiments were run and the empirical data is discussed and results presented.

High-Level Overview

The idea is to explicitly state a system's policy regarding allowable program activity. Subsequently, the APC is used to compare a selected program property against the policy, prior to installation. The APC determines whether a program's specified actions fall within the perimeter of a particular policy.

Definition 1 A policy is a set of rules that formally states allowable program behavior, in a particular system.

*Formerly Maria M. Pozzo.

¹SunOS is a trademark of Sun Microsystems, Incorporated.

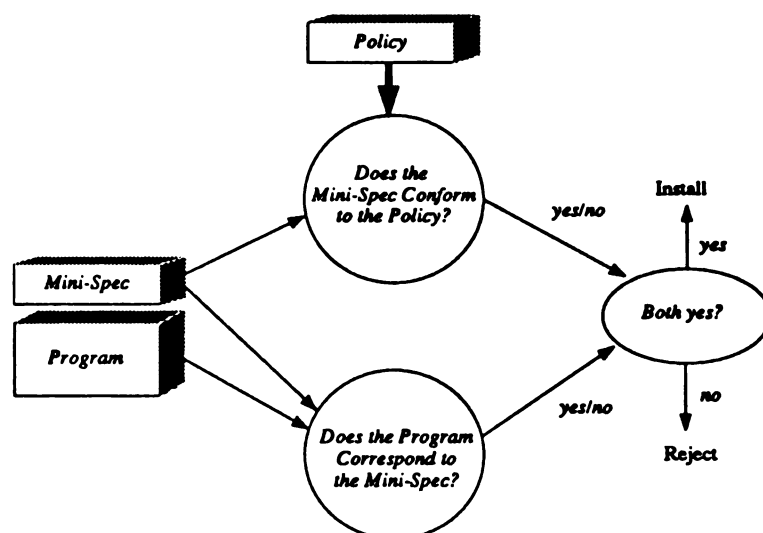


Figure 1: High-Level Overview

The term *specification* when applied to programs is usually taken to mean a general statement of *all* of the functional and/or other relevant properties of a program. To distinguish this form of specification from the more general use, the term *mini-spec* is used.

Definition 2 A *mini-spec* formally states a selected subset of the functional properties of a program's behavior.

This paper discusses the question: "Does the mini-spec conform to the policy?" Of equal concern is the correspondence between the mini-spec and the program it specifies. The scheme described in [2] proposes the use of a *filter* that will analyze a binary program and ensure that it conforms to what is stated in the mini-spec (see Figure 1). Traditionally, such an analysis has proven to be difficult. However, the assumption in [2] is that such programs should take full advantage of good software engineering techniques and need not contain the types of actions that are difficult to analyze, such as dynamic code generation, complicated computations for generating object names, and operating system manipulations. The basic premise is that reasonably engineered programs will be analyzable[2]. A reasonably engineered program is one that at least uses a structured methodology, is modular, and is written in a higher-level language. Current research described in [6] has implemented a filter program such as the one proposed in [2]. The filter approach appears promising.

An alternative method for verifying that the program conforms to the mini-spec is source code to specification correlation. The code-to-spec correlation process would have to be altered slightly since it is a one-to-one mapping between each line of code and each line of the specification. The mini-spec only states a *subset* of the program's behavior and such a mapping does not exist. However, verifying the source code against the mini-spec, as opposed to the binary, requires the existence of a trusted means for generating the binary from the source code. Without a trusted means, it would be possible to change the binary during the compilation stage.

The scope of this work is the specification of the mini-spec, development of policy, and the conformance of the mini-spec to a policy. It is assumed that mechanisms exist for verifying a program against its mini-spec, as described above. It is further assumed that once a program is verified against its mini-spec, whether by a filter program or some other means, the program and

the associated mini-spec must be sealed or encapsulated in some way to prevent tampering. These issues are well understood and will not be addressed here. The APC accepts a program/mini-spec pair that has been verified and properly sealed. The next section discusses the language used for stating mini-specs and policies.

A Regular Expression Based Specification Language

This section discusses the formal language that was developed for writing mini-specs and policies. The language is based on regular expression notation. The reasons for choosing regular expressions are presented in the next section. The syntax and use of the language is provided in Section , and the limitations of the language are discussed in Section .

Why Regular Expressions?

At the level of an applications program, a system resource might correspond to a file, device, block of memory, an so on. An applications program requests system services through system calls in which a system resource is referenced by a human-readable name. A name translation mechanism converts the human-readable name to the actual page(s) on disk, memory location, etc. The name translation mechanism assumes that the supplier of the name being translated has appropriate access, leaving all access decisions to the access control mechanism, if one exists. The problem is that conventional access control mechanisms are concerned with the access between users and resources, no check is made concerning the access between programs and resources. The example provided in [5] shows how the Fortran compiler only needs access to xyz.for and xyz.obj but can easily gain access to login.com if allowed by the access control mechanism.

The APC controls the access between programs and system resources. The policy is a set of rules which states allowable program behavior. There is one rule for each type of operation under control. Each rule is a set of human-readable names of system resources accessible to that operation. For example, the "modification rule" might be a set of names of directories where modification is permitted on the system. A mini-spec is also a set of rules, one for each type of operation that must be controlled in the particular system. Thus, a program's "modification rule" would be the set of human-readable names of system resources that the program might attempt to modify.

The notion of regular expressions has long been used in the design of lexical analyzers for grouping variable names and other tokens[4]. Other uses for regular expressions include text editors, pattern matching programs, and various file-searching programs. Regular expressions are well-suited for representing a set of strings such as the set of resource names, attribute names, or system call names that can be manipulated by a program.

For ease of discussion, the remainder of this paper will discuss policies and mini-specs that have only one rule, i.e., control a single operation. It is a simple matter to extend these ideas to multiple rules.

Discussion

An *alphabet*, Σ , is a finite set of symbols. A *(formal) language*, denoted L , is a set of strings of symbols from a particular alphabet. The language Σ^* is the set of all strings over a particular alphabet Σ ; thus all languages L over Σ are a subset of Σ^* . A regular expression, r , is a way of describing these languages. The notation $L(r)$ denotes the language described by r .

Let r_i be the regular expression that denotes the mini-spec for a particular operation of program i . The set of strings denoted by r_i is a finite-state language over some alphabet Σ . The language specified by r_i is denoted as

$$L(r_i) \tag{1}$$

Let p be the regular expression that denotes the policy, and $L(p)$ is the language denoted by p . Determining if the mini-spec for a given program is acceptable according to the policy of a specific

system then becomes a matter of determining if the language represented by the program's mini-spec is a subset of the language denoted by the policy, for each individual rule. More formally, if

$$L(r_i) \subseteq L(p) \quad (2)$$

for each corresponding rule in the policy, then the mini-spec is acceptable according to the system's policy.

Theoretically, the answer to equation 2 is straightforward. Ultimately, we want to be able to compare the two regular expressions without having to elucidate each element in the languages denoted by the expressions. To show that this can be done, consider the following properties of regular expressions.

1. First, the languages denoted by regular expressions are precisely those languages accepted by finite automata; so $L(r_i)$ and $L(p)$ are accepted by deterministic finite automata $M(r_i)$ and $M(p)$, respectively[4, 9]. The class of languages denoted by regular expressions is closed under complementation, i.e., the complement of a language denoted by a regular expression is also a language that can be denoted by a regular expression. To show this, let $M = (Q, \Sigma, \delta, q_0, F)^2$ be a deterministic finite automaton (DFA). Let L be the language over Σ accepted by M ; so $L \subseteq \Sigma^*$. Then, the complementary language, $\Sigma^* - L$, is accepted by the DFA $M' = (Q, \Sigma, \delta, q_0, Q - F)$. In other words, M and M' are the same except that the final states are opposite.
2. Second, by definition the languages denoted by regular expressions are closed under union. Therefore, given that the class of languages denoted by regular expressions are closed under complementation and union, it is simple to show that they are also closed under intersection. Let L_1 and L_2 be languages over the alphabet Σ . Then $L_1 \wedge L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

Returning to equation 2, to answer the question, consider the following equation:

$$(\Sigma^* - L(p)) \wedge L_i(r) = \emptyset \quad (3)$$

Consider the language that is the complement of the language denoted by the policy. If the language denoted by the program's mini-spec, $L(r_i)$, has anything in common with the complementary language of the policy, $\Sigma^* - L(p)$, then clearly, $L(r_i)$ is not a subset of $L(p)$.

Although it can be shown theoretically that two regular expressions can be directly compared to determine if one is a subset of the other, algorithmically the problem is considered PSPACE-complete[3]. Solutions to many PSPACE-complete problems exist, and in fact, these algorithms work well when certain constraints are applied. The APC currently implements one such algorithm. The primary constraint is that the regular expressions that denote the mini-spec and the policy, must be simple enough to be processed during a reasonable processing cycle. For regular expressions that do not meet this constraint, two alternatives are available. A detailed discussion of the algorithm, and these alternatives is provided in [10].

Language Syntax and Usage

Table 1 identifies the basic operators of the language. The precedence is listed from highest to lowest with the loop operator having the highest precedence. Parenthesis are used to override the normal precedence order as the example in Table 1 shows. The first four operators listed, loop, concatenation, union, and parenthesis for grouping, are standard regular expression operators. Note, however, that the loop operator indicates $0 \leq i$ where i is limited by the maximum string length on a particular machine. Thus, the expression a^* denotes a finite language, which differs from the standard definition.

Nonterminal definitions provide user-friendliness by allowing a user to define commonly used expressions. Nonterminal definition names are 1-8 characters in length, all small letters; the definition itself is written in the operators of the language. Nonterminal definitions can be referenced via the angle brackets ($< >$) operator and can be embedded. The depth of macro definitions is machine dependent but it is wise to keep a limit on it. Nonterminal definitions are

²Where Q is the set of all states in the DFA, Σ is the input alphabet, δ represents the transition function, q_0 is the initial state, F is the set of final states, and $q_0, F \subseteq Q$. [4, 9]

Table 1: Syntax of Language

SYMBOL	MEANING	EXAMPLE
*	loop - $0 \leq i$	$a^* \Rightarrow \{e, a, aa, aaa, \dots\}$
	concatenate	$ab \Rightarrow \{ab\}$
()	union	$a \mid b \Rightarrow a \cup b; \{a, b\}$
	grouping	$(a \mid b)^* \Rightarrow \{a, b, aa, ab, ba, bb, \dots\}$
::=	nonterminal definition	$a \mid b^* \Rightarrow \{a, b, bb, bbb, \dots\}$ $id ::= (a \mid b)^*$
< >	nonterminal reference (1)	$\langle id \rangle \Rightarrow (a \mid b)^*$
[]	series (2)	$[a \mid b \mid c \dots]$
{cwd}	current working directory	$\{cwd\}/(a \mid b) \Rightarrow \{cwd/a, cwd/b\}$
{home}	home directory	$\{home\}(a \mid b) \Rightarrow \{home/a, home/b\}$
files	define expression	$files ::= \langle id \rangle$
Notes:		
(1) Nonterminals are 1-8 characters, all small letters.		
(2) Series can be used with nonterminal definitions.		

stored in files; example nonterminal definition files, called **sysdefs** and **unixdefs**, are shown in Figure 2. A file of nonterminal definitions can be referenced via the “#include” mechanism of Unix. The square brackets operator ([]) is used to define a long series such as all the lowercase letters or all the digits. This operator is an implementation enhancement; parenthesis or nothing can be used to represent the same thing, i.e., $(a \mid b \mid c) \equiv a \mid b \mid c \equiv [a \mid b \mid c]$. An improvement to the current language would be to allow $[a - z]$ to indicate all the lowercase letters.

The current working directory operator {cwd} and the home directory operator {home} can be used in systems that have knowledge about filesystem location, such as Unix or Multics. In a Unix system, for example, all directories in the system would include {cwd}/, {home}/, and all other directory locations.

Policies and mini-specs are stored in files. Figure 2 shows the mini-spec for the modification operation for the calendar program. The last line of a mini-spec or policy file must begin with the “files” operator followed by the defines or goes into (::=) symbol as shown in the example in Figure 2. The example shows that the calendar program can create files in the current working directory of the form “cal” followed by a string as defined in the **unixdefs** nonterminal file. The grammar for the language just described is provided in [10].

Writing Policies and Mini-Specs for Real Programs

A mini-spec is written either during program development by a user wishing to submit a program for installation or it can be written for programs that already exist. Detailed information must be available in order to write a mini-spec for an existing program. This information might include source code, detailed design documentation, programmers notes, and test results.

Writing a policy requires knowledge about the particular threat, the system vulnerabilities, and the desired environment. Although some users may have the sophistication for writing a policy, in most cases the policy should be written by a security officer or other security personnel. Section discusses the application of the APC to the virus problem, the development of anti-viral policy, and presents results of using the APC to test for undesirable program behavior (in this case viral behavior) in 125 Unix programs.

<u>sysdefs:</u>		
small	::=	[a b ... z]
large	::=	[A B ... Z]
digit	::=	[0 1 ... 9]
special	::=	[! / ... +]
<u>unixdefs:</u>		
atom	::=	(<small> <large> <digit> <special>)
string	::=	<atom><atom>*
dir	::=	/(<string>/)*
<u>"mini-spec" for calendar:</u>		
#include "sysdefs"		
#include "unixdefs"		
files	::=	/tmp/cal<string> std(err out)

Figure 2: Example Nonterminal Definitions

The Language Preprocessor

The APC first calls a preprocessor to resolve the "#include" statements, and to check the syntax of the mini-spec and the policy. The preprocessor enforces the rule that all expressions must denote a regular language (all expressions must be regular). Regular languages with an infinite number of strings are represented by the "*" operator in the regular expression or a cycle in the Finite State Machine. Non-regular languages do exist and cannot be represented by these constructs. For example, a language such as the one denoted by { a^n : n is prime} has no simple periodicity, is not regular, and cannot be represented by the constructs of regular languages[9].

The preprocessor enforces this rule by making sure that all referenced nonterminal definitions have been defined before they are referenced. A nonterminal is not defined until after the carriage return, prohibiting expressions of the form: $\langle \text{foo} \rangle ::= a | \langle \text{foo} \rangle$. This forces the use of the "*" operator for all loops and is sufficient to enforce that all expressions denote regular languages. The preprocessor, part of the APC, provides an error message and the line number in the file where the error occurred, when a syntax error, such as the one just described, is encountered.

Evaluation of the Language

The language for writing policies and mini-specs is based on regular expressions, which is a commonly accepted notation for representing a name space. It is a straightforward matter to use the language to represent the names of system resources manipulated by programs, such as file and device names, file attributes, environment variables, and system call names. Another application would be to *encode* behavior patterns in a regular expression, such as user profiles, using the constructs of the language.

The primary drawback to this language is that the expressions must be kept simple enough to be processed by the APC during a reasonable processing cycle. A "reasonable processing cycle" will be specific to a particular installation depending on the price, in processing time, one wishes to pay for protection from viruses. This requires some knowledge about regular expressions on the part of the individual writing the mini-spec or policy. In some cases, the mini-spec may have to be broken down into several pieces or simplified according to regular expression transformation rules.

Using the APC

The APC command is as follows:

`apc name1 [name2] [-Idir...]`

name1 is the name of the file that contains the mini-spec; *name2* is the name of the file that contains the policy. If *name2* is not supplied, the default is the system policy.

`-Idir` “`#include`” files are sought first in the current working directory, then in the directories named in the `-I` options.

When the policy is not supplied by the user a default system policy can be used. This allows a user to test out the mini-spec against an individual policy before submitting it to the system administrator for installation. It also allows the user to have an individual policy that is more stringent than the system policy. For example, suppose the system policy allows modifying of any files in any user directory. If a user does not wish to allow modification of the home directory, then the user can write an individual policy that only allows modifications to files not in the user's home directory. The user can then use the APC, supplying the user-specific policy, when deciding whether to execute new programs. The APC returns a message indicating whether or not the program is acceptable according to the policy.

Application of the APC to the Virus Problem

This section discusses the application of the APC to the virus problem. All experiments were conducted under SunOS. The distinguishing characteristic of a computer virus is its ability to infect other programs by *modifying* them to include a copy of the virus. Although there are other behaviors of programs that can be controlled to prohibit viral activity, all of the policies discussed here focus on the modification operation, specifically, the modification of files and directories. All of the policies contain a single rule which specifies the directory and file names where modification is allowed. All of the mini-specs also contain one rule which specifies the directory and file names that the associated program could attempt to modify.

Test Suite

All of the programs in sections 1 & 6 of the Unix Reference Manual[11] were examined for possible inclusion in the test suite³. These programs include editors, compilers, game programs, printing programs, and other basic utility programs available to normal users. The modification behavior of each program were studied by reading the Unix manual pages, looking at source code when available, and talking with Unix developers when necessary. In some cases, the modification activity of a program could not be adequately identified due to the lack of sufficient information. Such programs could not be included in the test suite. A total of 125 programs comprise the test suite.

Many of the programs in the test suite had the same modification behavior. A total of twenty-three unique mini-specs were written to represent the 125 programs in the test suite. Three additional mini-specs were written to simulate programs infected with a real virus. The reason for including “infected” programs was to show whether each policy prevented infected programs from being accepted. All twenty-six mini-specs were tested against each identified policy. The details of the program study, the mini-specs, and the programs they represent, are presented in detail in [10].

Policies

The development of the anti-viral policies was approached from opposite angles. On the one hand, normal user activity was identified and several policies were developed to allow that behavior. On the other hand, several viruses were identified and policies were written to prohibit their behavior. The basic methodology was to develop policies that allowed normal user behavior and prohibited abnormal (viral) behavior.

³All of the programs in these sections of the manual are available to normal users. For purposes of these experiments, programs which require special privileges were not considered.

Policies for Normal User Activity

The first policy considered is a loose policy that allows modification to any directory and any file on the system. The reason for including such a policy is to show the operation of the system with respect to viruses when no restriction is placed on allowable modification activity. This policy is comparable to no policy and all programs in the test suite, including the three "infected" programs, were accepted. At the opposite end of the spectrum of policies is a tight policy that only allows modifications to the standard output and standard error devices. This policy is included to represent a policy that allows very little modification activity. A policy such as this effectively keeps all viruses out of the system, it does not accept any editors, compilers, or programs to manipulate files.

Policy 3 allows modification to files in specific directories: `/dev/`, `/tmp/`, `{cwd}` or `{home}`. This policy does not allow modifications to any other directory, nor does it allow modifications to any subdirectory of these directories. A total of 50% of the programs in the test suite, are accepted by this policy. More importantly, this policy successfully rejects all three "infected" programs. Policy 4 is more restrictive and only allows modifications to the current working directory; this does not include files in subdirectories of the current working directory. Only 42% of the programs in the test suite are accepted by this policy. All three "infected" programs are also rejected. Policy 5 is the opposite of policy 4 in that modifications are allowed to any file located anywhere in the system *except* the current working directory. A total of 59% of programs in the test suite were accepted by policy 5, however, two of the "infected" programs were also accepted. Policy 6 is also very restrictive; this policy only allows modification to objects located in the temporary directory `/tmp/`. Although this policy correctly rejects the three infected programs, it only accepts programs 42% of the programs in the test suite.

Of all the policies in this section, policy 3 which allows modifications to all four specific directories appears to be the best policy in that it accepts that largest percentage of programs. None of the policies accept any compilers or editors.

Policies to Prevent Specific Viruses

Four Unix viruses are identified in this section. The details of each virus are not presented for security reasons. Instead, each virus is described in terms of the name space of directories and/or files that it modifies.

The Murray Unix Virus infects Unix shell programs. Murray looks for shell programs to infect in the user's `bin/` directory and the current working directory. Murray also creates and modifies several files in the current working directory that start with a `."`. Since shell programs are not identifiable by their name, the policy is written to prohibit any modifications to the user's `/bin/` directory or current working directory (modifications to subdirectories of the current working directory are permitted). Furthermore, this policy only allows modification to files in the home directory that do not start with a `."`. Modifications to other files in other locations in the system are permitted. This policy accepts the same set of programs that were accepted by policy 5 - the policy that does not allow modification to the current working directory. Although this policy rejects the mini-spec "infected" with the Murray virus, the other two "infected" programs are accepted.

To simulate the IBM Christmas Tree virus in a Unix environment, policy 8 was written. This virus is not technically a virus since it doesn't copy itself to another program, i.e., the virus doesn't *infect* other programs. Instead, this virus, or worm, sends a copy of itself to all of the electronic addresses of all the users listed in the victim's address alias file. To stop the spread, policy 8 prohibits modification of the mail spool directories, the location where all outgoing mail is queued until it is sent out of the system. This policy accepts 61% of the programs in the test suite. However, it restricts the proper usage of the mail programs so that mail cannot be sent out of the system by anyone. Also, this policy does not reject the other two "infected" mini-specs.

The virus described in [1] is a general virus that searches for any executable file and appends itself to the executable. Since this virus can modify any file anywhere in the system, policy 9

Table 2: Policy Acceptance Rate

Policy	Name	Mini-Specs Accepted	Total Programs Represented	% of Programs Accepted	Infected Programs Accepted
1	Loose	<i>all</i>	125	100%	yes
2	Tight	1	48	38%	no
3	Specific Directory	1,7,10-12, 14,18,20	63	50%	no
4	{ <i>cwd</i> }	1,11,14,20	52	42%	no
5	not { <i>cwd</i> }	1-7,9,10,12, 15,17-19,22,23	74	59%	yes
6	/ <i>tmp</i>	1,10,12	52	42%	no
7	Anti-Murray	same as 5	74	59%	yes
8	Anti-Xmas	1-7,9-12, 14,15, 17-20,23	76	61%	yes
9	Anti-Generic	1,7,18	55	44%	no
10	Anti-Worm	1-6,9,10,12, 14,15,18-20	64	51%	no
11	Combo	1,18	49	39%	no

prohibits all modifications except to the devices. Such a policy is very restrictive and, although it successfully prohibits all viruses, it allows only 44% of programs.

Policy 10 is intended to prohibit the Internet Worm. The Internet worm modified many things on the system. Most important were the sockets that it wrote to in order to transfer the worm from the host machine to the victim machine. The worm used unnamed sockets which makes it impossible to use this scheme to prohibit writing to sockets. The worm also created files beginning with the letter "x" which it later deleted in an attempt to hide itself. Prohibiting modification to files whose name begins with the letter "x" would halt the Internet worm but it would be a simple matter to re-write the worm to use some other letter. Policy 10 does prohibit the "infected" mini-spec which represents a program that is carrying the Internet Worm and it accepts 56% of all useful programs. This policy also successfully rejects the other "infected" programs. However, this policy would be very simple to circumvent. A second generation of this virus could choose a different letter or randomly select a letter other than "x". In this way the virus would be accepted by the policy.

Policy 11 was developed by using the union operator of the language and combining policies 7, 8, 9, & 10. The reason for including such a policy is to experiment with a single policy for all viruses vs. a policy for each individual virus. This policy does successfully reject all three "infected" programs, but it only accepts 39% of all programs. Also, it is easy to see that this policy doesn't prohibit *all* viruses, just those described here.

Evaluation of Empirical Results

Table 2 shows the acceptance rate for each policy just described. Column 1 identifies the policy by number, column 2 lists the number of the mini-specs that were accepted, column 3 shows the total number of programs represented by the accepted mini-specs. Column 4 indicates the percentage of the 125 programs in the test suite that were accepted by each policy. The last column indicates if any of the "infected" mini-specs were accepted.

Eleven total policies were identified: 6 policies allow normal user behavior while 5 policies prevent a specific virus(es). Policies 1, 5, 7, and 8 accepted one or more of the "infected" mini-specs; mini-specs that were included to represent programs infected with a virus. Policies 10 & 11 are considered weak policies as it would be very simple to create a virus to circumvent these policies. Policy 3, which allows modifications only to specific directories (*/dev/*, */tmp/*,

{*cwd*}, {*home*}) accepts the largest number of programs. In general, policies based on normal user behavior accept a larger percentage of programs, especially most necessary programs; those based on specific viruses are easily circumvented.

None of the policies that are effective against viruses accept any editors, compilers, linkers, or other programs considered necessary for normal user operation. This leads to the conclusion that basing the policy on the modification behavior of programs, although an important activity to control for virus prevention, by itself is inadequate. There are two reasons for this. First, the nature of Unix programs is that they either modify *only* standard error and/or standard out (39% of programs in the test suite), or they modify any file in any directory (34% of programs in the test suite). This results in mini-specs that specify program behavior as "all or nothing" for 72% of the programs in the test suite. Clearly, this approach would be more effective if the modification characteristics of Unix programs were restricted. The question is: could this be done easily without a great deal of impact on users?

The second reason is the unavailability of user input. This approach is a static, preventative mechanism, it is applied once, prior to program installation. The mini-spec attempts to capture the potential dynamic behavior of a program but because of its static nature, this results in many programs being rejected at installation time that could operate within the confines of the policy at run-time. For example, suppose the policy states that the only modifications allowed are to */tmp/* and files can have any name as long as they do not begin with the letter "a". If the mini-spec for a particular program modifies files of any name in */tmp/* the program would not be accepted for installation. If the mechanism were applied at run-time instead, the program might execute within the confines of the policy, i.e., not modify any files that begin with the letter "a". Thus, the unavailability of user input results in policies that appear overly restrictive.

The modification behavior of a program is the most obvious characteristic of a computer virus which is why it was chosen as the behavior of focus for this study. However, although important for virus control, policies based on this behavior are not restrictive enough, too restrictive, or can easily be circumvented. Alternative behaviors to be considered include system call patterns, modification of file attributes, modification of environment variables.

Lastly, the fact that the language does not contain a construct for intersection, and especially complementation, results in policies that are more complex than if these operators had been available. Policy 8, which is intended to prevent the Murray virus, is an example of this. Since there is no way to directly express policies such as "anywhere but */bin/*", policies tend to be overly restrictive. An alternative would be to specify all the file names and directories that *cannot* be modified in the policy. Then, if the APC determines that the mini-spec is *not* a subset of the policy then the mini-spec would be considered acceptable.

Evaluation of the Overall Approach

This research has shown that the proposed mechanism can keep viruses out of a system and still accept a percentage of most necessary programs; it is a feasible and practical approach. It has further been shown that controlling the modification of files is an important behavior to control for virus prevention, but results in policies that are too restrictive, not restrictive enough or results in policies that can easily be circumvented. The APC clearly separates the policy from the mechanism that enforces the policy. This will allow future studies to investigate alternative behaviors for virus prevention and control.

On the negative side, the complex expressions that result due to the lack of the intersection and complementation operators of the language, sometimes result in lengthy execution times. However, the approach suggested in the previous section, i.e., testing for whether a given mini-spec is *not* a member of the policy's language, may provide a simple solution to this problem. Also, the inability to capture run-time user input results in a greater number of programs being rejected. A possible solution to this problem is to move the mechanism into the run-time environment that would not require a mini-spec and could operate on the actual file or directory name; a much simpler check would need to be made in this case. The next section discusses future research.

Conclusions and Future Work

The investigation of alternative program behaviors is an obvious extension to this work since it would not require any additional implementation. Since the mechanism and policy are clearly separated, the study of alternative behaviors simply means supplying a new test suite. Several possibilities for other behaviors to investigate include system call patterns, file attribute modifications and environment variables. In the case of system call patterns, a possible approach is to encode the patterns of modification system calls in a regular expression using the constructs of the language. The same is true for file attributes or environment variables although specific names could also be used in the same way as the current study. In any case, developing a new test suite is not a simple matter. In order to provide a useful test suite, a large set of programs should be investigated and studied. As this research has shown, policies based on normal user activity are the most effective.

To accommodate user input, this mechanism could be extended to represent user behavior. For example, user behavior could also be encoded in a regular expression using the constructs of the language. The union operator of the language could then be used to combine a particular user's behavior with that of a specific program. The combined specification could then be checked against the policy. This would involve a study of user behavior, particularly user modification behaviors. The advantage of this approach would be the information regarding each specific user rather than just the program's behavior. This could also be classified by class of users or group of users.

Another obvious extension of this work is to implement the mechanism as a run-time mechanism and to run experiments with test suites in both the static, prevention mode and the dynamic, detection mode. This would provide information regarding prevention vs. detection of viruses.

Lastly, extending this work to different types of systems, such as a DOS or Macintosh system, would be a useful project. In both cases, neither system takes advantage of the memory protection features of the hardware, allowing programs to modify any location in the system. The mechanism described here, especially if implemented as a run-time mechanism, could be used to restrict the modification activity of programs despite the failure to use memory protection. Since many of the DOS viruses encountered directly modify memory addresses, an appropriate behavior to study might be the actual calls for modification rather than the directory and file names as was done in this study.

This research has implemented a mechanism for comparing program behavior against a policy that states allowable program behavior. This approach has been applied to the virus problem and shown to be a practical and feasible approach for preventing computer viruses. This mechanism does not have a high impact on performance, and does not result in inconvenience to users. When used with a filter program such as one described in [2, 6], it does not rely on some number of humans to act in a trustworthy fashion. Most importantly, this approach clearly separates the policy from the mechanism that enforces the policy. In this way a variety of policies and program behaviors can be studied and tested using the APC.

References

- [1] F. Cohen. Computer Viruses. In *Proceedings of the 7th National Computer Security Conference*, pages 240-263, 1984.
- [2] S. Crocker and M. Pozzo. A Verification-Based Virus Filter. In *IEEE Symposium on Research in Security and Privacy*, 1989.
- [3] M.R. Garey and D.S. Johnson. *Computers and Intractability A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, CA, 1979.

- [4] J.E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company, Inc., Reading, MA, 1979.
- [5] P.A. Karger. Limiting the Damage Potential of Discretionary Trojan Horses. In *IEEE Symposium on Security and Privacy*, 1987.
- [6] P. Kerchen, R. Lo, J. Crossley, G. Elkinbard, K. Levitt, and R. Olsson. Static Analysis Virus Detection Tools for Unix Systems. In *19th National Computer Security Conference*, October 1990.
- [7] N. Lai and T.E. Gray. Strengthening Discretionary Access Controls to Inhibit Trojan Horses and Computer Viruses. In *Proceedings of the Summer 1988 USENIX Conference*, 1988.
- [8] R. Levin, E. Cohen, W. Corwin, F. Pollack, and W. Wulf. Policy/Mechanism separation in Hydra. In *Proceedings of the Fifth Symposium on Operating Systems Principles*, November 1975. University of Texas at Austin.
- [9] H.R. Lewis and C.H. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, Inc, Englewood Cliffs, NJ, 1981.
- [10] M.M. Pozzo. *Towards Computer Virus Prevention*. PhD thesis, University of California, Los Angeles, 1990.
- [11] Sun Microsystems, Sun Release 3.2. *Commands Reference Manual*, 1986. Mountain View, CA.

IMPROVEMENT OF DATA PROCESSING SECURITY BY MEANS OF FAULT TOLERANCE

Gilles Trouessin* Yves Deswarte* Jean-Charles Fabre* Brian Randell**

* LAAS-CNRS & INRIA
7, Avenue du Colonel Roche
31077 Toulouse Cedex – France

** Computing Laboratory, The University
Newcastle upon Tyne
NE1 7RU – United Kingdom

Abstract

This paper discusses various different solutions to the problem of reliable processing of confidential information. One of the major difficulties of this problem comes from the fact that conventional techniques for achieving reliability, on the one hand, and security on the other, tend to be in opposition to each other. The different solutions presented here have been classified in three distinct types: two are related to classical security techniques (protection, and encryption) and the third is a new technique, the fragmentation-redundancy-scattering technique, which it is claimed demonstrates a potentially advantageous unified approach to the provision of reliability and security, based on fault tolerance. Finally, a qualitative comparison of these solutions is given, taking into account both dependability, openness and performance criteria.

Keywords: secure architectures, integrity, reliability/availability, protection, encryption, fragmentation.

1. Introduction

In this paper we concern ourselves with the provision of high reliability and availability, and the preservation of data confidentiality, in large scale distributed systems, such as ones based on workstations connected over one or more high speed LANs.

1.1. Problem statement

Dependability, a generic concept – defined as *the trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers* – may be viewed w.r.t. different properties [8] and so enables the definition of a number of different dependability attributes, including: **availability** (w.r.t. *readiness for usage*), **reliability** (w.r.t. *continuity of service*), **safety** (w.r.t. *avoidance of catastrophic consequences on the environment*), **security** (w.r.t. *prevention of unauthorized access and/or handling of information, i.e., provision of data integrity and confidentiality*).

Some of these attributes (reliability/availability and security) are often considered separately because the techniques used to achieve them are usually perceived as being mutually antagonistic. Firstly, *reliability* and *availability* are generally achieved by incorporating mechanisms for tolerating any faults (especially accidental faults) that occur, or that remain despite attempts at fault prevention during the system design process. These techniques will of necessity involve space and/or time redundancy; they can easily take advantage of a distributed computing architecture by means of replicated computation using sets of untrusted¹ (or fallible) processors. Secondly, *security features* are generally achieved by means of fault prevention mechanisms (w.r.t. intentional faults, such as intrusions) whereby critical applications are implemented on physically and/or logically protected computers. Such protection is usually based on the *TCB* (Trusted Computing Base) or *NTCB* (Network Trusted Computing Base) concepts [17] [18].

From the above we see that what can be termed an antagonism between reliability/availability and security arises in at least the two following ways [7]: (i) *accidental-fault tolerance* (by means of replication) increases the number of potential access points to confidential information and thus can reduce the effectiveness of the protection techniques; (ii) *intrusion prevention* (by means of a local *TCB* or a *NTCB* partition) can suffer from the fact that one cannot justifiably rely either on a single *TCB* (which forms a classical “single point of failure”), or on the local *TCB/NTCB* partition of each computer inter-connected to the network.

To be adequately realistic, a solution dealing with this antagonism must, we believe, take into account the following two requirements: (i) *trusted area reduction*, by which we mean that the security provided by a potential

¹ Here we use the term *trusted* component to mean one that is assumed to be highly reliable and available, and impervious to intrusions (i.e., not to be a source of deliberate faults), in its intended environment.

solution should depend on as small as possible a *trusted area*, because it is impossible to place confidence on all of the processors in the network; (ii) *openness*, by which we mean that a potential solution must not be (excessively) software and/or hardware dependent, but instead allow implementation over a network of heterogeneous systems. The former requirement, regarding *trusted area reduction*, would also contribute to the latter one, regarding *openness*, since untrusted processors belonging to same critical application would thus not be security-dependent.

1.2. Reliable processing of confidential information

Let us consider the problem of *reliable processing of confidential information* as involving a combination of the three following features a), b) and c):

- Simple processing* (Fig. 1a): processing (P) is applied to a set of input data (D) in order to obtain a set of output results (R). Both areas are shaded to denote the fact that neither the processor (the lower area) nor the environment containing the I/O (input/output) devices and which provides the inputs and accepts the outputs (the upper area) are trusted.
- Reliable processing* (Fig. 1b): the redundant execution of P (by means of processor replication) in order to provide data integrity for D and R , and reliable processing of P .
- Confidential processing* (Fig. 1c): in this, and indeed all cases where confidentiality is required, input is provided from, and output is delivered back to, a trusted area. Neither of the two regions of Fig. 1c is shaded; this is to indicate that P is executed, and its I/O prepared/received, securely (in similarly trusted areas), in order to preserve the confidentiality of D , R and (perhaps) P .

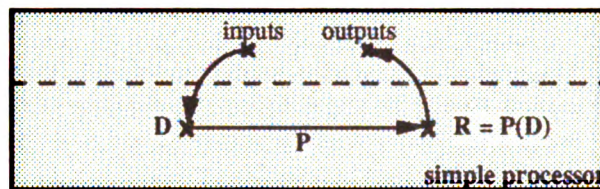


Figure 1a: Simple processing

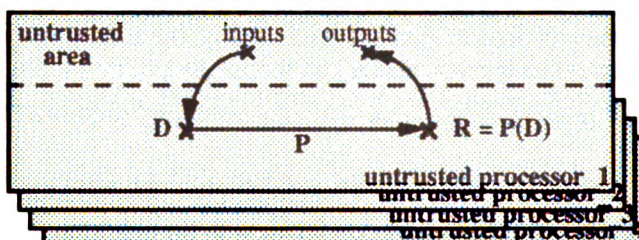


Figure 1b: Reliable processing

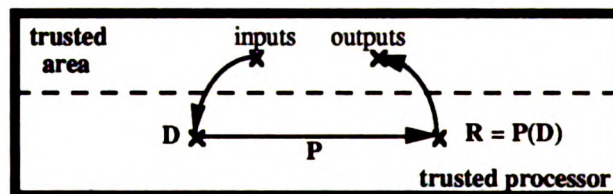


Figure 1c: Confidential processing

2. Achieving Combined Reliability and Security

2.1. Approach 1: Protection

This first approach is based on a classical security technique, *protection*, an intrusion-prevention technique which is based on forecasting and preventing, as far as possible, the different intrusions that could damage overall system security. This technique may be implemented by either of two different solutions: 1) *centralized protection* or 2) *local protection*. In each case, replication is also employed, in order to add both processing reliability and data integrity.

2.1.1. Solution 1.1: Centralized protection and replication

This first solution (Fig. 2) is in fact the logical combination of the features (reliability and confidentiality) represented in Figs. 1b and 1c. As in all cases where confidentiality is required, I/O operations, for each given user, are performed in a trusted area using a similarly trusted processor. However, the processing is replicated and executed by trusted processors that all belong to the same trusted area as that where the data is provided and the results received by the user. Solution 1.1 can be developed on the basis of a centralized TCB as recommended in the *Orange Book* [17], using a specific architecture, i.e., a fault-tolerant computer system, such as Tandem or Stratus systems.

There are two possibilities for preserving the confidentiality of data whilst it traverses the medium used for

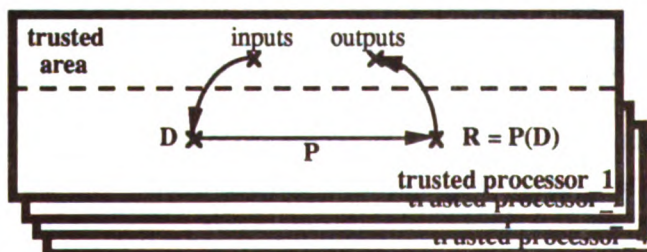


Figure 2: Solution 1.1 – Centralized protection and replication

inter-processor communication, depending on whether or not the medium is considered as part of the trusted area. In the latter case, confidentiality preservation of the whole critical application is based on the encrypting of all communications between processors. In either case, the *trusted area reduction requirement*, and thus also the *openness requirement*, are not adequately met, since all the processors (together, one can be sure, with significant portions of their operating systems), and perhaps the communication medium, are considered as part of the trusted area.

Solution 1.1 is thus in practice perhaps well suited for very specific highly critical applications such as some types of military computation but does not fit well with general-purpose applications which may invoke remote processors and use several distinct networks.

2.1.2. Solution 1.2: Local protection and replication

This second solution (Fig. 3) is in fact a network generalization of the previous solution. I/O operations are performed in a trusted area located on a special trusted processor. Normal processing is still replicated but it is now accomplished in an untrusted area, on processors which are in general untrusted. Each of these processors is however protected by means of a local *TCB* and a *NTCB* partition as recommended in the *Red Book* [18].

An *Authentication—Access Control* scheme (AAC and AAC', in Fig. 3) is needed between the special trusted processor and the other processors involved in the critical application, in order to ensure the overall security of the application.

There is only one possibility for the preservation of the confidentiality of the communication medium since processing is executed in the untrusted area: the medium must be considered as part of the untrusted area and all communications between the different processors must thus be encrypted.

Several hardware and/or software implementations of this solution have been developed, for example: the *Distributed Secure System* [2] [11], the *LOCK co-processor* [12] in connection with the *SDNS* project [15], *Secure Sun OS* [16].

With Solution 1.2, we can see that the *trusted area reduction requirement* is partially respected: (i) *respected*, since each processor involved in executing the critical application is now considered to be untrusted, and to be situated in an untrusted environment area; (ii) *partially*, because each of these untrusted processors must be protected by a local *TCB* and *NTCB* component, which are each in fact a local (albeit perhaps small) trusted software and/or hardware mechanism operating in an untrusted environment. However, this means that the mechanism should therefore be made tamper-proof, so that it cannot be opened without destroying its content. Such tamper-proof devices also need to possess a master key in order to communicate securely with other such devices in the untrusted area; and in practice, must be small and essentially maintenance-free.

The other requirement, openness, is not respected because each implementation of this solution requires the help of a *TCB/NTCB* partition to enforce security on the different processors of the network. This is the main drawback of Solution 1.2, particularly where the *TCB* or *NTCB* component is merely software running on the otherwise untrusted processor, because it is very difficult to protect the component from and by something as complex as an operating system, (e.g., Unix in the *LOCK/ix* project). However when the *NTCB* is in special-purpose hardware, monitoring all communications to and from the untrusted processor (as in the *DSS* project), its task, and that of making it tamper-proof, are more readily achievable. Anyway, in all cases, if any of the local *TCB/NTCB* components are corrupted or replaced by Trojan Horses, all the others are threatened so that the security of the whole network is compromised and the confidentiality of the critical application lost.

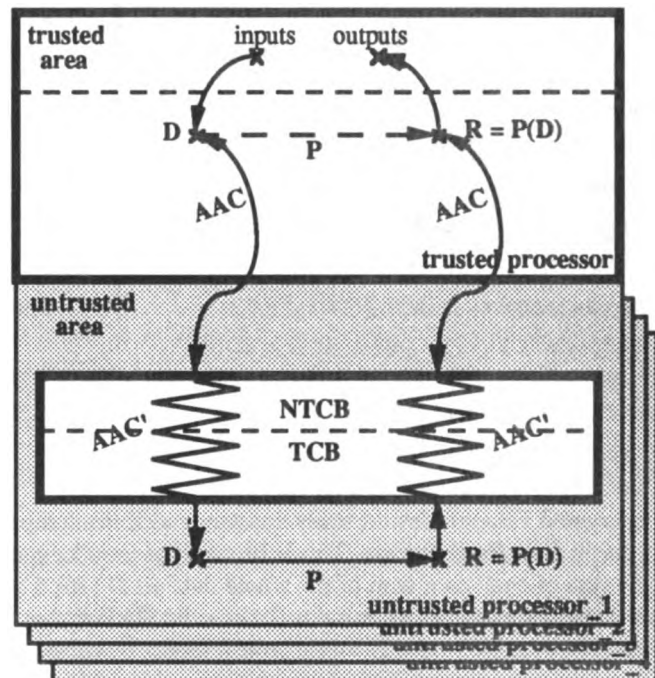


Figure 3: Solution 1.2 – Local protection and replication

2.2. Approach 2: Encryption

This second approach is based on another classical security technique, *encryption*, which is a well established technique for preserving the confidentiality of communications and file archiving. It can be used for preserving the confidentiality of information processing in two different ways: 1) *homomorphic encryption* or 2) *black-box encryption*. In each case, replication is also used, in order to provide both processing reliability and data integrity.

2.2.1. Solution 2.1: Homomorphic encryption and replication

With this solution (Fig. 4), a user's I/O operations are as always performed in a trusted area, and reliability features are obtained by means of processing replication, again in an untrusted area, but in an encrypted way. In the one trusted area, a special trusted processor transforms the data set (D) and the processing (P) by means of a specific kind of encryption technique (C) into an encrypted data set (D') and encrypted processing (P').

Only certain types of encryption, called *privacy homomorphisms* [1] [10] [13], are suitable for such transformations. However, when C is of such a type, P' can be securely accomplished in the untrusted area, by untrusted processors. Encrypted results (R') obtained in the untrusted area can then be de-crypted (C^{-1}) in the trusted area to obtain results in clear (R):

- D thus has an image D' according to C : $D' = C(D)$;
- P also has its own "image" P' depending on both C and P features: P' is a function of (C, P) ;
- R' is thus an image of D' with P' : $R' = P'(D')$.

With Solution 2.1 communication confidentiality is directly preserved by means of encryption and no additional techniques are required for this purpose.

But a restriction must be observed in implementing any scheme based on Solution 2.1. If an intruder can access the encrypted value of any arbitrary constant and if the comparison operator is available then usage of a privacy homomorphism is no longer secure. This is because the intruder can use a simple binary search strategy to discover the encrypted value of each data item of the whole data set D [10]. However in some particular cases (where there is no need for a comparison operator) Solution 2.1 is valid [1] [13]; but these cases are very limited (very specific banking transactions, for example) and thus this approach cannot be considered as providing a general solution.

Because of the above restriction, we can say that Solution 2.1 partially respects the *openness requirement* since processing is securely executed only in some particular cases (if C is a privacy homomorphism and if P does not provide the comparison operator). However, we can say that Solution 2.1 respects the *trusted area reduction requirement* perfectly, since processing is completely executed by untrusted processors, without any need for trusted devices in the untrusted area.

2.2.2. Solution 2.2: Black-box encryption and replication

This solution (Fig. 5) exhibits some common features with the previous solution: I/O operations are performed in the trusted area and processing in the untrusted area, reliability is obtained by replication and confidentiality by encryption. However, homomorphic encryption is replaced by *black-box encryption*. In fact, processing is apparently executed in encrypted form: $R' = P'(D')$, since only encrypted data D' , encrypted processing P' and encrypted results R' can be observed in the untrusted area.

In reality, processing is executed in clear inside a trusted "black box" associated with each untrusted processor. This solution involves three steps:

- encrypted input data (D') is received and de-crypted with C^{-1} : $D = C^{-1}(D')$;
- normal processing P is executed in order to obtain results R : $R = P(D)$;
- results R are encrypted with C in order to obtain R' that can be sent out of the black box securely: $R' = C(R)$.

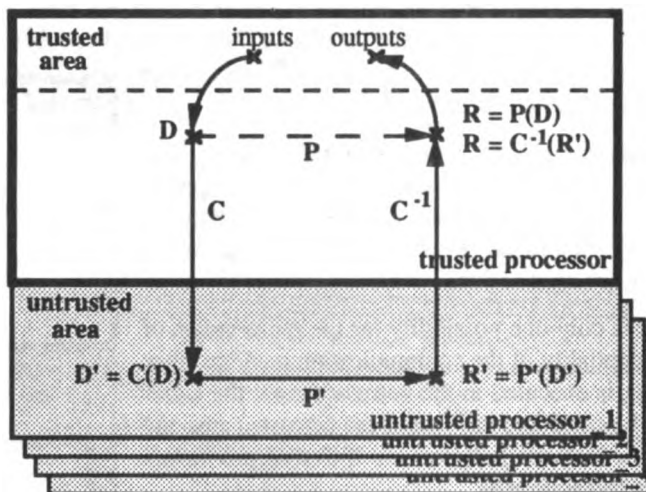


Figure 4: Solution 2.1 – Homomorphic encryption and replication

The trusted black box thus contains a decrypting-processor, a small size memory, a processor and an encrypting-processor. To be really secure, it must be tamper-proof (as described in Section 2.1.2 above).

However, Solution 2.2 suffers from several major drawbacks [7], though the first three listed below are essentially similar to those possessed by Solution 1.2:

- *protection against a Trojan-horse black box*: in order to be qualified as *trusted*, it must not be possible to replace the black box by a *Trojan-horse black box* during its operational life (leave alone during initial installation);
- *management of encrypted addresses*: all data received by the trusted black box, such as addresses, are encrypted and are thus more difficult to decode and use;
- *management of communication keys*: one (or several) master cryptographic key(s) is(are) required in order to allow secure communications, which increases the management complexity of key distribution and use;
- *increase of local memory space*: for management of encrypted addresses or communication keys and local data storage, thus increasing the local memory space required for the black box whereas it ideally should, as mentioned previously, be small.

Because of these drawbacks, we can say that though Solution 2.2 is feasible, like Solution 1.2 it does not meet the *openness requirement*, because the security in the untrusted area is really hardware- and software-dependent (i.e. black box dependent), and it does not meet the *trusted area reduction requirement* very well, since a trusted device (the trusted black box) must be installed essentially in each processor.

2.3. Approach 3: Fragmentation-Scattering

This third approach is based on what can be termed a “unified fault tolerance” technique, the *Fragmentation-Redundancy-Scattering (FRS)* technique, since it provides, in a single mechanism, means of tolerating both accidental and intentional faults, and hence of providing both reliability and confidentiality of data and its processing. Fragmentation involves defining fragments of information so as to ensure that, once isolated into physically separate processors, each fragment is of little value to a potential intruder due to the lack of significant information content in any one processor. (In principle such fragmentation can either be achieved at the programming language level, where it can take advantage of programmer-defined data structuring, or at the operating system level, where it is based on machine-level data types, such as bytes, words and/or pages. Particularly in the former case there is the possibility of requiring, and making use of, programmer-supplied constraints indicating which data items it would be especially undesirable for an intruder to be able to correlate.) Such fragments are then replicated, and the replicated fragments scattered across a (preferably large) number of processors.

FRS has been developed and successfully demonstrated in the context of a secure file archiving system [5] [6] and in the course of research into security management [3] [4]. In the processing context [7] [19] the approach relies on the correct execution of a majority of a set of copies of each of a number of program fragments with their corresponding data fragments, these fragments being widely distributed across a number of untrusted processors. Research to date on the application of FRS to processing¹ has resulted in the devising of two rather different implementation schemes: 1) *fragmentation-scattering and replication* or 2) *fragmentation-scattering and threshold*.

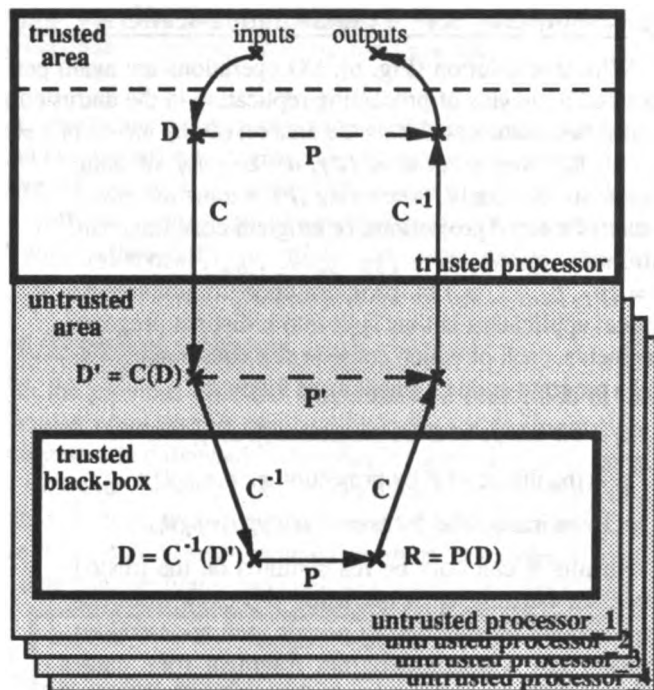


Figure 5: Solution 2.2 –
Black-box encryption and replication

¹ The FRS technique applied to processing is called *Fragmented Data Processing (FDP)*. Some actual examples of this FDP technique are presented in the Appendix to this paper.

2.3.1. Solution 3.1: Fragmentation-scattering and replication

With this solution (Fig. 6), I/O operations are again performed in the trusted area, reliability features are again obtained by means of processing replication in the untrusted area, but in a fragmented fashion. In the trusted area, the trusted processor transforms the data set (D) by means of a set of projections, or data-fragmentation functions, $F = \{f_1, f_2, \dots, f_n\}$, into a set $D = \{d_1, d_2, \dots, d_n\}$ of data fragments. Similarly, processing (P) is transformed by means of a set of projections, or program-code fragmentation functions, $G = \{g_1, g_2, \dots, g_n\}$, into a set $P = \{p_1, p_2, \dots, p_n\}$ of program-code fragments. A critical application is thus split into n distinct program fragments, each of which consists of a data fragment d_i and a program-code fragment p_i , as follows:

- d_i is the image of D by projection f_i : $d_i = f_i(D)$;
- p_i is the image of P by projection g_i : $p_i = g_i(P)$;
- r_i is the image of d_i by processing p_i : $r_i = p_i(d_i)$.

Results R can only be reassembled on the trusted processor because each untrusted processor does not have enough information to permit such re-assembly: perhaps just a single program fragment (one data-fragment d_i , one program code-fragment p_i) and thus one result fragment r_i , for a given application. In practice, several program fragments could however be mapped on the same physical processor, provided that they do not in sum reveal any significant information.

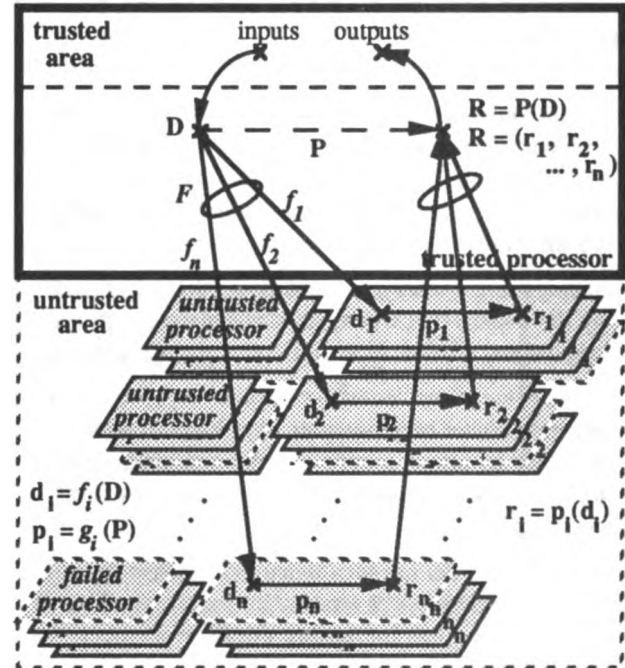


Figure 6: Solution 3.1 –
Fragmentation-scattering and replication

Solution 3.1 possesses two main beneficial features:

- *different classes of fragmentation functions (f_i and g_i) can be defined*: security depends on the way the fragmentation functions (f_i and g_i) are chosen: for a given critical application different classes of fragmentation functions are possible (data and/or program-code driven) and different fragmentation strategies are also possible, thus allowing different security features to be obtained (data and/or program-code confidentiality preservation) [19] [20];
- *security does not depend on f_i or g_i confidentiality*: security does not rely on a potential intruder being ignorant of the semantics of the fragmentation functions (f_i or g_i).

A potentially major drawback of Solution 3.1 is that it might be expensive, in terms both of performance and program development effort. The major issues involved are as follows:

- *additional memory space overheads*: the memory space overheads due to replication are exactly the same as for any other solution using replication; those due to fragmentation come from the fact that there can be an overlapping of the data fragments d_i derived from D . In such a case and if these overheads are important, then another fragmentation strategy (based on a larger, but then admittedly perhaps less secure, fragmentation granularity) might be adopted;
- *increased number of processors*: this is unavoidable, but in the introduction of this paper it was indicated that we are assuming the basic global environment of the problem of *reliable processing of confidential information* in a distributed computing environment. In many such systems, for example university computing networks, a large number of processors can be idle very often [9]; in such cases the provision of reliability/availability by means of processing replication and of security by means of fragmentation-scattering can together take advantage of such unused processing power;
- *communication overheads*: since many more processors are required than with the two previous approaches, much more communication traffic may be induced; for example by data fragment exchanges between distinct program fragments. In such cases, and if communication overheads are significant, this again, might motivate the adoption of another fragmentation strategy, with larger fragmentation granularity;
- *development effort*: if significant development effort is needed to apply the technique to each separate application this would constitute the major cost of this approach, which would probably only be justifiable on very critical

applications which were thereafter to receive extensive use. To date, investigations have been somewhat application-specific, but the prospect of application-independent methods of using the technique is now being considered.

From the above, and from experiments to date, we claim that solution 3.1 respects the *openness requirement*; we believe that most types of critical application can be fragmented, at least to a degree, largely because a wide range of fragmentation possibilities are offered by means of the different fragmentation classes and strategies. The *trusted area reduction requirement* is certainly respected since no trusted software and/or hardware components need be situated in untrusted areas. In particular, the fragmentation functions (f_i and g_i), though they are used in the trusted area and are the basis of security, are not confidential and could actually be held in untrusted areas.

2.3.2. Solution 3.2: Fragmentation-scattering and threshold schemes

This solution (Fig. 7) has some points in common with the previous one: I/O operations are performed in the trusted area, processing in the untrusted area, and confidentiality requirements obtained by means of fragmentation-scattering. But reliability/availability are now obtained by using so-called threshold schemes [14] applied to processing, instead of using processing replication. With the threshold technique, at least s (the threshold number) shadows of a given secret are needed to reconstitute the secret and less than s shadows do not give any information about this secret. Like error correcting codes, the threshold scheme technique thus imposes some redundancy in order to tolerate accidental but also intentional faults (especially intrusions w.r.t. data confidentiality and integrity), and during processing.

In the trusted area, the special trusted processor transforms the data set (D) by means of a specific set of projections, taking into account the threshold scheme, $F^s = \{f_1^s, f_2^s, \dots, f_m^s\}$, into a set of data fragments, $D^s = \{d_1^s, d_2^s, \dots, d_m^s\}$. Similarly, processing (P) is transformed by means of a specific set of projections, $G^s = \{g_1^s, g_2^s, \dots, g_m^s\}$, into a set of program-code fragments: $P^s = \{p_1^s, p_2^s, \dots, p_m^s\}$. A critical application is thus split into $m > n$ distinct program fragments (each consisting of a data fragment d_i^s and a program-code fragment p_i^s) as follows:

- d_i^s is the image of D by projection f_i^s : $d_i^s = f_i^s(D)$;
- p_i^s is the image of P by projection g_i^s : $p_i^s = g_i^s(P)$;
- r_i^s is thus the image of d_i^s by p_i^s : $r_i^s = p_i^s(d_i^s)$.

As with Solution 3.1, R can only be reconstituted on the trusted processor because each untrusted processor does not have enough information, possessing in principal just one data and one program-code fragment for a given application.

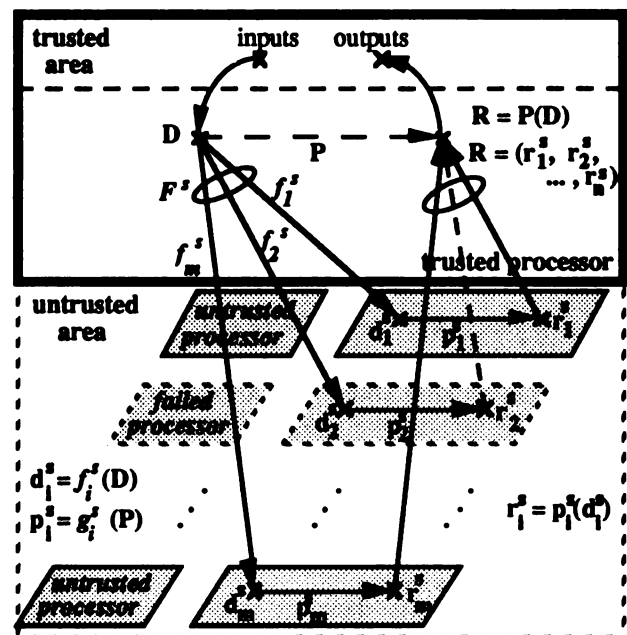


Figure 7: Solution 3.2 – Fragmentation-scattering and threshold schemes

Solution 3.2 possesses many of the advantages and disadvantages of Solution 3.1 - however, it allows reduction of the number of required processors to m , instead of the $n \cdot r$ needed by the previous solution (if r is the replication level) and thus reduces various types of overhead, including communication overheads. But its main drawback is that not all threshold schema are suitable. With Shamir's threshold schema based on polynomials [14] for example, a restricted set of polynomials must be chosen: addition is conserved with this kind of processing, but multiplication becomes quickly expensive (high degree polynomials are required in order to perform multiplications securely) and comparison can be used only if the restricted polynomial set offers the total order property (any shadow of any secret must be comparable and respect the total order property imposed by the secrets).

Thus we can see that this solution meets the *trusted area reduction requirement* fully and that just a restricted set of threshold schemes can be used (high degree and ordered polynomials). Two main problems must also be considered: (i) the security of the fragmentation projections (F^s and G^s) must remain as strong as the (proved) security of the threshold scheme technique; (ii) the cost of these fragmentation projections, i.e., in terms of development effort and required execution time, must not be too important compared to the execution time of the program-code fragments (P^s).

3. Qualitative Comparison

Different comparison criteria must be considered, *dependability*, *openness* and *performance criteria*, to provide a qualitative comparison (Table 1) of the different solutions presented in the previous section.

3.1. Dependability, openness and performance criteria

As explained in the introduction of this paper, *reliable processing of confidential information* is assumed to be concerned with different dependability criteria (*dc*), i.e., goals and requirements, and one openness criterion (*oc*):

- *dc_1* – **reliability**: data processing reliability (and availability) and data integrity;
- *dc_2* – **confidentiality preservation (security)**: preservation of data (and perhaps processing) confidentiality;
- *dc_3* – **trusted area reduction**: non excessive security-dependence on trusted areas;
- *oc_1* – **openness**: non excessive security-dependence on specific software or hardware.

Five performance criteria (*pc*) are considered, since valid solutions must not involve excessive performance overheads:

- *pc_1* – **number of processors**: the number of processors required, not counting those needed for redundancy purposes (see *pc_2*);
- *pc_2* – **redundancy overheads**: the number of processors required for redundancy purposes;
- *pc_3* – **number of messages**: the number of messages induced by the total set of processors;
- *pc_4* – **memory size**: the local (to each processor) memory space required for the solution implementation;
- *pc_5* – **system connectivity**: the number of inter-connection links required between the different processors.

As yet we do not have extensive experimental data concerning the performance and costs of *FRS*, when applied as here to processing, as distinct from file archiving and security management. Therefore Table 1 gives our subjective comparison of the six distinct solutions presented in this paper, taking into account the above dependability and performance criteria. Five values, corresponding respectively to the following qualitative scale, are used to characterize the extent to which the different criteria are satisfied:

- the five values are: —, —, =, + and ++;
- the corresponding qualitative scale is: *very unfavourable*, *unfavourable*, *no influence*, *favourable* and *very favourable*.

Table 1:

Qualitative comparison of the different solutions with respect to dependability, openness and performance criteria

<div> <div>Table 1:</div> <div> Qualitative comparison of the different solutions with respect to dependability, openness and performance criteria </div> </div>	dependability			open- ness	performance													
	rel.	security			<div>reliability/avail- ability/integrity</div>	<div>confidentiality preservation</div>	<div>trusted area reduction</div>	openness	<div>number of processors</div>	<div>redundancy overheads</div>	<div>number of messages</div>	<div>memory size</div>	<div>system connectivity</div>					
	dc_1	dc_2	dc_3	oc_1										pc_1	pc_2	pc_3	pc_4	pc_5
Approach 1: protection	==	-	--	--	++	==	==	==	==									
<i>Sol. 1.1</i> : centralized protection + replication	==	-	--	--	++	==	==	==	==									
<i>Sol. 1.2</i> : local protection + replication	==	-	==	-	++	==	==	==	==									
Approach 2: encryption	==	-	++	==	+	==	==	==	==									
<i>Sol. 2.1</i> : homomorphic encryption + replication	==	-	++	==	+	==	==	==	==									
<i>Sol. 2.2</i> : black-box encryption + replication	==	-	==	-	+	==	==	==	==									
Approach 3: fragmentation-scattering	++	++	++	++	--	==	--	+	-									
<i>Sol. 3.1</i> : fragmentation-scattering + replication	++	++	++	++	--	==	--	+	-									
<i>Sol. 3.2</i> : fragmentation-scattering + threshold schemes	+	+	++	++	==	+	==	+	==									

3.2. Comparison results

Table 1 shows that in our opinion Approach 3 leads to better results w.r.t. the dependability (and openness) criteria: this is due to the fact that it is a unified concept providing both accidental- and intentional-fault tolerance. Approach 1 and Approach 2 globally present better results w.r.t. performance criteria *pc_1*, *pc_2*, and *pc_3*; this is due to the fact individual processing replicas (*D*, *P* and *R*) are not split over different processors, as is the case with Approach 3.

It should be noted that local (but not global) memory overheads are not expected to be significant with Approach 3; this is because more processors are required for almost the same global memory occupation and each processor then needs a smaller local memory space.

From this analysis and the judgements expressed in the Table, we conclude that all of the approaches described here are in principle capable of providing solutions, suitable for at least some situations, to the problem of *reliable processing of confidential information*. Each approach, however, has some weaknesses:

- *Approach 1*: poor reduction of the trusted area, and either poor security openness (Solution 1.1) or poor confidentiality preservation (Solution 1.2, when based on OS-enforced software partitioning);
- *Approach 2*: poor confidentiality preservation, and either poor security openness (Solution 2.1: operator restriction) or poor reduction of the trusted area (Solution 2.2);
- *Approach 3*: though adequate w.r.t. our dependability requirements this approach is somewhat poor w.r.t. performance, a situation which we believe can be ameliorated by exploiting the large range of possible fragmentation strategies already identified for this new technique.

4. Concluding Remarks

The originality and potential attractiveness of the fragmentation-scattering approach is that it provides a unified means of achieving both reliability and security. At this stage, much remains to be discovered about its real advantages and disadvantages, w.r.t. the pre-existing combined techniques against which we have compared it, but we believe the analysis presented above shows that it has considerable promise as a new means of providing fault tolerance against both accidental faults and intentional faults such as intrusions.

Acknowledgements

The authors wish to thank very much Jean-Claude Laprie and David Powell for their earlier basic ideas about the *Fragmentation-Redundancy-Scattering* approach for ensuring high reliability/availability and security, Jean-Michel Fray for his contribution to the concept of *Fragmented Data Processing* for reliable processing of confidential information, and the reviewers of the submitted version of this paper for their constructive comments. This work was partially supported by DRET under convention no. 88.34.051.00.470.75.01 and the PDCS (Predictably Dependable Computing Systems) project no. 3092 of the European ESPRIT programme.

References

- [1] N. Ahituv, Y. Lapid, S. Neumann, *Processing Encrypted Data*, Comm. of the ACM, Vol. 30(9), Sept. 1987, pp. 777-780.
- [2] D.H. Barnes, R. MacDonald, *A Practical Distributed Secure System*, Proc. NEOS'85 (Networks and Electronic Office Systems), London – United Kingdom, IERE, 1985, pp. 83-91.
- [3] L. Blain, Y. Deswarte, *An Intrusion-Tolerant Security Server for an Open Distributed Computing System*, Proc. European Symposium On Research In Computer Security, ESORICS'90, Toulouse – France, Oct. 24-26, 1990, Pub. AFCET, ISBN 2-9036778-9, pp. 97-104.
- [4] Y. Deswarte, L. Blain, J.-C. Fabre, *Intrusion Tolerance in Distributed System*, Proc. 1991 IEEE Symposium on Research in Security and Privacy, Oakland – California, May 20-22, 1991, pp. 110-121.
- [5] J.-M. Fray, Y. Deswarte, D. Powell, *Intrusion Tolerance Using Fine-Grain Fragmentation-Scattering*, Proc. 1986 IEEE Symposium on Security and Privacy, Oakland – California, April 7-9, 1986, pp. 194-201.
- [6] J.-M. Fray, Y. Deswarte, D. Powell, *A Secure Distributed File System based on Intrusion Tolerance*, Proc. 4th International Conference on Computer Security, IFIP/Sec'86, Monte Carlo, Dec. 2-4, 1986, pp. 407-416.
- [7] J.-M. Fray, J.-C. Fabre, *Fragmented Data Processing: an Approach to Secure and Reliable Processing in Distributed Computing Systems*, Proc. 1st IFIP International Working Conference on Dependable Computing for Critical Applications, Santa Barbara - California, Aug. 23-25, 1989, pp. 131-137, published in *Dependable Computing for Critical Applications*, Ed. A. Avizienis, J.-C. Laprie, Springer Verlag 1991, Vol. 4, ISBN 3-211-82249-6 & 0-387-82249-6, pp. 323-343.

- [8] J.-C. Laprie, *A Unifying Concept for Reliable Computing and Fault-Tolerance*, in *Dependability of Resilient Computers*, Ed. T. Anderson, Blackwell Scientific Publications, 1989.
- [9] D.A. Nichols, *Using Idle Nodes in a Shared Computing Environment*, Proc. 11th ACM Symposium on Operating Systems Principles, Austin – Texas, Nov. 1987, pp. 5-12.
- [10] R.L. Rivest, L. Adleman, M. Dertouzos, *On Data Banks and Privacy Homomorphisms*, in *Foundations of Secure Computation*, Ed. R.A. Demillo, D.D. Dobkin, A.K. Jones, R.J. Lipton, Academic Press, 1978, pp. 169-179.
- [11] J.M. Rushby, B. Randell, *A Distributed Secure System*, IEEE Computer, Vol. 16(7), 1983, pp. 55-67.
- [12] S.O. Saydjari, J.M. Beckman, J.R. Leaman, *Locking Computers Securely*, Proc. 10th National Computer Security Conference, Sept. 1987, pp. 129-141.
- [13] J. Seberry, J. Pieprzyk, *Cryptography: An Introduction to Computer Security*, Ed. R.P. Brent, Prentice Hall, 1989, 375 p.
- [14] A. Shamir, *How to share a secret*, Communications of the ACM, Vol. 22(11), Nov. 1979, pp. 612-613.
- [15] E.R. Sheehan, *Access Control within SDNS*, Proc. 10th National Computer Security Conf., Sept. 1987, pp. 165-171.
- [16] B. Taylor, D. Goldberg, *Secure Networking in the Sun Environment*, USENIX Association Summer Conference Proc., Atlanta – Georgia, 1986, pp. 28-37.
- [17] TCSEC, Department of Defense Standard, *Trusted Computer System Evaluation Criteria*, Department Of Defense, DOD 5200.28-STD, Dec. 1985, 121 p.
- [18] TNI, National Computer Security Center, *Trusted Network Interpretation of TCSEC*, National Computer Security Center, NCSC.TG.005, July 1987, 278 p.
- [19] G. Trouessin, J.-C. Fabre, Y. Deswarte, *Reliable Processing of Confidential Information*, Proc. 7th International Conference on Information Security, IFIP/Sec'91, Brighton – United Kingdom, May 15-17, 1991, pp. 210-221.
- [20] G. Trouessin, *Quantitative Evaluation of Confidentiality by Entropy Calculation*, Proc. 4th Computer Security Foundations Workshop, Franconia – New Hampshire, June 18-20, 1991.

Appendix

Depending on the way that fragmentation is performed, it is possible to define different classes of fragmentation techniques, which are concisely described below and more detailed in [19]. In addition, fragmentation granularity is another parameter that can be considered in the choice of a given fragmentation class.

The first class of fragmentation technique relies on fine grain fragmentation (bit or small group of bits), and is in effect a sort of bit-slicing technique: each data item within the whole data structure is split into f fragments of b bits. Thus each of the individual processors has only a local view of the data structure, i.e., b bits out of $f.b$ bits. By this means, the global value of each data item and thus its semantics can be effectively hidden. The code must then be transformed into a set of code “fragments”, each of which has been modified so as to work appropriately with the corresponding fragmented data. Actually, this is not really code fragmentation since the code is simply slightly transformed, and the original program's semantics is unlikely to be effectively disguised from an intruder. The main interest of this class of fragmentation technique is that it allows a quantitative evaluation of confidentiality preservation by means of entropy calculation [20], but its main drawback, due to its restriction to fine granularity only, is that the *openness requirement* is not respected.

A second class can be defined when applied at the module level. Each program fragment (data- and code-fragment) corresponds to a single entity in the whole program structure (an instruction block, a program module or a library function) delimited by breaks in the code sequence. Each such fragment is then replicated in r distinct copies scattered over the network. Actually, this class of fragmentation technique is opposite to the previous one because the code is first fragmented in connection with its structure (this is particularly interesting in the case of modular programming languages or block-structured languages) and then the whole data structure is consistently fragmented (with respect to the first code fragmentation). This means that each code fragment will be associated with its own local variables and this implies also that global variables must be fragmented and shared by several distinct fragments.

From the above two classes of fragmentation technique we can derive a third one. As with the first class, this third class takes into account the data structure in order to apply a first step of fragmentation. This first step is well suited to the preservation of data confidentiality, since its aim is to cut some of the semantic links in the tree that could otherwise be built with the main semantic links of the data structure. And secondly, as with the second class, it is applied at the module level and thus similarly relies on the program structure, so is a technique which is well suited to preservation of code confidentiality. This third class thus is used at the programming language level, and is a compromise between fragmentation at the variable and program module levels. If fine grain fragmentation is required for efficient confidentiality preservation then data and code fragmentation can be applied in alternation to get an overall fragmentation which depends on both data and code structures.

Information Security: Can Ethics Make a Difference?

Corey D. Schou
Associate Professor
Department of Computer Information Systems
Idaho State University

John A. Kilpatrick
Associate Professor
Department of Management
Idaho State University

ABSTRACT

Information is a vital organizational asset that affects ongoing decision making. It has a finite life span therefore if it is delayed in its distribution, it has reduced value; if a proper user fails to have access, it has no value.

The objective of attempts to secure the organizational information system is to see that unauthorized use is not possible, that destructive viruses are not introduced, and that unauthorized study and alteration of records and files does not occur during the distribution of data and information throughout the organization while guaranteeing that proper users have easy access to their information. Are these objectives strictly technical problems, or is it possible and appropriate to broaden the scope to include the ethical issues that are raised as the security system is developed and installed? The argument in this paper is that it is both appropriate and necessary to consider the broader issues.

INTRODUCTION

Information is the lifeblood of an organization that over the years has become recognized as an asset. Although determining the value of this asset from an accounting standpoint is difficult, it should be protected like any other. One of the dominant characteristics facing any firm attempting to become and to stay competitive is the dependence on information processing that relies on computers and computer software. In this paper we attempt to address many of the ethical issues facing managers in organizations as they attempt to cope with the complexity and cost of acquiring, integrating and securing information systems in the workplace. In large organizations, this task is assigned to an Information Resource Manager who is responsible for all aspects of information processing from data entry to the Executive Information System¹. This manager plays an important role in the security of the organization's information assets. It is critical that Information Resource Managers convey the importance of resource security to senior management of the organization.

In the process of performing this task, the manager must balance two competing objectives that are for all practical purposes antithetical. The first is that of ease of access to information to meet the requisite variety needs of decision makers within a system². The second is that of maintaining security, confidentiality and privacy of organizational information assets.

1. Schou, Corey D., "Computer Security: Training Needs for Managers," *Data Management*, Auerbach, September, 1990.

Frequently, this process is viewed as a technical problem rather than the more complex socio-technical problem that should address some of the following issues:

- Whose rights are to be considered?
- To what extent are these rights in conflict?
- What are the responsibilities of the information specialists?
- How honest and trusting are the members of the user community? In what sense do they represent a 'community'? What are the implications, if any, of their holding certain interests in common?
- How trusting ought they to be? What is implied in the use of the term *ought*? Of the term *trust*?

These socio-technical problems are fundamental ethical issues. These issues may or may not be legal issues. The manager should be aware that which is legal is not necessarily a logical equivalence of either ethical or right.³

QUESTIONS OF PURPOSE AND VALUE

Since there is a documented body of law that governs portions of our behavior and a cult of technology which asserts that it can make our electronic information systems invulnerable to external penetration, we tend to rely upon it. To complete the protection of our information assets, we must develop an awareness of value systems. This development must be more than another set of rules and regulations that dictate how we should behave. They should be, on the other hand, an internalized set of behaviors. We should ensure that rules and technology do not become the sole focus of our security activities. These are destined to fail of their own weight in the long term. John LaCarré in one of his novels makes the point about the impact of technology on human activity by stating:

George Smiley: "You've made technique a way of life. Like a whore, technique replacing love."⁴

In a technological environment, it is easy to focus on the techniques designed to accomplish goals and on the technology used to assist in the accomplishment of those goals. At times the tendency is to allow the focus on technique to overshadow the purposes or ends. For example, a common observation of the modern 'rat race' (a revealing metaphor) is that participants spend so much time and energy pursuing the good life that little remains for living. As this result suggests, it is easy to become obsessed with the tools and in the process to forget the purpose of the tools.

Although information security systems are adequate from the successful system control, they do not always take into account the corresponding human impact and implications. This brings us to the question - What is the role played by the values that members of a community hold that form the choices made and the ends toward which those choices are directed? Stated another way, what is the significance of the way a member of the information systems community views the world and his or her relation to that immediate world and to other members of the community? These values and perceptions underlie the choices that individuals make, the goals that are pursued, and the priorities that are established. They affect both the means that are selected and the ends toward which efforts are directed.

2. Beer, Stafford, *The Brain of the Firm*, John Wiley & Sons, New York, NY, 1981.

3. Richards, T., Schou, C.D. & Fites, P.E. "Information Systems Security Laws and Legislation," in *Information Technology Resources Utilization and Management: Issues and Trends*, Idea Group, Harrisburg, Pa., 1990.

4. LaCarré, John, *Tinker, Tailor, Soldier, Spy*, Doubleday, New York, NY, 1986.

ETHICAL SYSTEMS

What are the purposes of computer information systems? Information systems are organizational mechanisms that collect data and distribute information. Frequently these systems rely on electronic devices such as computers; however, the 'office boy' carrying a scrap of paper to the file drawer also meets this definition.

Some systems relate to governmental objectives (e.g., national defense, collection of revenue, monitoring of international trade), some to business purposes and needs (e.g., efficiency and competitiveness), but all must relate at some level to social needs and values. For example, one might argue that a fundamental value is respect for the rights of others. Another might be that the overall objective is a better quality of life for all members of the community.

There are several ways of identifying and deciding ethical issues. One of the most common Judeo - Christian ways of categorizing these approaches is the rules Vs. consequences criteria. The first argues that our actions should be guided by general rules or principles: do not harm; tell the truth; do not steal; have respect for persons as 'ends in themselves.' The second argues that we should assess the *rightness* of an action or decision by the consequences that will likely result. Most commonly the second approach identifies some value or values, and measures an action by the extent to which these values are or are not enhanced, or whether progress is made toward certain goals, such as a better life for all. From a practical standpoint it may be recognized that, for most people, over a span of time and in different situations, both approaches will be used. That is, in general some ethical rule may seem appropriate but under extreme circumstances exceptions to the rule or principle will appear ethically acceptable because of the likely consequences.

ETHICS AND INFORMATION SYSTEMS

For an information system to function effectively and efficiently, there must be a free flow of data and information among all participants. In the ideal situation, there would be no barriers to this flow; this would improve the probability that 'perfect information' is in the hands of the decision makers. Of course, for this to occur, there would have to be perfect confidence and trust within the organization.

Confidence and Trust

Information - adequate, relevant, timely, understandable - is a precondition of an efficient and free society. Yet it is a means to power . . . Therefore, the rights to create property in information, to withhold, to disclose, to determine when and how disseminated are critical⁵.

In this section we are interested in the ethical issues involving the creation, control, use, abuse, dissemination, protection, manipulation or alteration, examination and destruction of information and its attendant data in computer systems. In order for the above information activities to take place efficiently and legitimately, there must be some minimal level of trust and confidence in the systems which handle the information. Is it also necessary for there to be some minimal level of trust among and between the various users of the system?

Assuming that such a level is necessary, what are the preconditions in order for this confidence and trust to exist? It appears clear that first there must be a proven and recognized history of dependability, both within the firm and with similar systems.

5. Behrman, Jack, "Information Disclosure, the Right to Know and the Right to Lie" in Behrman, *Essays on Ethics in Business and the Profession*, PrenticeHall, Englewood Cliffs, NJ., 1988, 79.

By raising these issues in the context of the firm's culture or atmosphere, one ethical principle is implied: that there must be respect for persons and certain property rights. This falls within the first approach identified above, which argues for the assessment of choices in light of certain ethical principles or rules. Actions which result in intrusion, examination, alteration or destruction of information belonging to others might be judged as morally wrong because they violate the principle of respect for persons. The second approach, that of looking at the consequences of an action, might suggest that in order for a community to meet the needs of its members, individuals within the community must be able to have some confidence in systems of communication. According to this view, it could be argued that actions that unduly interfere with the smooth operation of information and communication systems, or that diminish the confidence and trust in these systems, should be judged as unethical.

Definitions

As a starting point for determining ways of evaluating actions, it is appropriate to construct several definitions. The term *legitimate* is fundamental to the notion of balancing rights and responsibilities. For the purposes of this paper, it is argued that for an action or behavior affecting an information system to be legitimate, it must aid in the achievement of one or more objectives of the system without unduly interfering with progress toward other accepted objectives. The definition can be applied to the ethical management of information. One objective of the system is to provide information that is without deception and is understandable, timely, relevant, complete and appropriate to the user. Upon examination, it can be seen that this definition suggests both the practical and ethical elements of managing computer information systems.

SPECIFIC CONCERNS RELATING TO THE DESIGN OF SECURE SYSTEMS

Those involved in the design of a secure information system must be aware of the conflicting rights, responsibilities and needs of system users and professionals, and of the implications of some of these conflicts. Some paradoxical assertions may serve to illustrate:

- For people to have trust in an information system, the manager must trust no one.
- Systems which are truly trustworthy must use control processes that inhibit use.

Another way of putting the problem, as Clifford Stoll suggests in his book, *The Cuckoo's Egg*,⁶ is that as administrative controls are added to ensure trustworthiness, the system becomes more difficult to use. This means that the people for whom the system is designed end up finding another, less trustworthy but more easily accessible system to use. The term administrative controls refers to those policies and procedures imposed by a manager that are designed to regulate the individuals and activities covered by the policies and procedures.

Administrative controls are designed and implemented to make sure that people act in the way that managers desire. Generally this means, in ways that advance organizational objectives through fixed procedures. This may be something as simple as standardizing the ways employees claim reimbursements for job-related expenses. It may mean something as broad as the budget process, which attempts to regulate the activities of and to set standards for the entire firm. Frequently, however, it also refers to the need to regulate behavior when it is perceived that:

6. Stoll, Clifford A., *The Cuckoo's Egg*, Doubleday, New York, NY, 1989.

- a) there is motivation to engage in activities for personal, as opposed to organizational, reasons; and
- b) those activities are potentially harmful to the organization, to organizational values or to other organizational members.

If the interests of individuals always coincided with those of the organizations with which he or she lives and works, there would be very little need for administrative controls. It is at the point where these interests diverge that the need for controls arise. Further, some conflicts arise because of simple misunderstandings, some arise because of differences in perceptions, some are due to different priorities, world-views or values, and some come about because of individual malevolent intent.

Finally, there are those instances where it is in an individual's self-interest for everyone else to exercise a degree of moral restraint while he or she exercises none. This can be seen as the *free-rider* problem or, to use Garrett Hardin's excellent metaphor, it is the "tragedy of the commons"⁷. In this environmental fable, the members of the community maintain their livestock on the commonly held grazing grounds. Animals can safely be added until the carrying capacity of the grounds are reached. However, it is to the benefit of any individual community member to add animals to his herd on the commons. The overall costs of degradation are borne by the community but the benefits accrue to the individual community member. The tragedy is that individuals can safely benefit in the short run while the long-term costs are dispersed. Greed is rewarded. One lesson for members of the community is that, unless they are willing to eliminate all cooperative efforts, the exercise of some moral restraint by each individual is necessary.

EXAMPLES OF ETHICAL ISSUES CONFRONTED IN ORGANIZATIONS

As long as the information system consists of 'office boys' carrying paper from place to place, the problems are less complex. If he takes something home - he has stolen - he is wrong. However, when the organization begins to rely on electronic means, this issue becomes more clouded. The same individual can take or send electronic images of the same information without overtly changing it. (After all, what is the value of a simple '0' or '1'.) The following are examples of some problems that are uniquely electronic.

Pirated Software

One of the more obvious and most prevalent problem deals with the use of pirated software. The temptations are obvious and the risk of disclosure is slight. Why then the concern? There are several ethical issues here, but perhaps the overriding one is that of the failure to recognize intellectual property.

As with many ethical concerns, one can arrange many positions along a continuum. In this instance, one can take an extreme individualist or ethical egoist position, and argue that pirating another's software is not a big issue, and is useful for financially strapped organizations. Further, one can argue that it is the responsibility of the developer to take measures to limit the ease of pirating. In any case, is it stealing if the property isn't gone?

At the other end is the argument that:

- a) there are rights that are being violated while copying;
- b) that no community can exist that refuses to acknowledge and protect the rights of its members; and

7. Hardin, G., "The Tragedy of the Commons," *Science*, 162, December 1968, 1243-1248.

- c) that progress will be limited unless there is some incentive for individuals to develop tools that will prove useful in solving the problems of the community.

The manager then must address the issue of whether to allow - profit from - the pirating of another's intellectual creation, or, if the policy is to ensure that this does not occur within the business, what policies will be required to ensure that it does not occur.

Criminal Entry

Even if one has problems recognizing intellectual property, physical property is easier to define. This situation is analogous to the problem of the 'office boy' If someone breaks your physical lock, or physically enters your premises, there is little question about 'right'.

However, the problem of unwarranted entry into proprietary electronic information systems with criminal intent is more complex. Using technological means, each firm will obviously wish to ensure that its own system will not be so penetrated. What of information gained either inadvertently or through the wizardry of an employee who also happens to enjoy the challenge of breaking into another institution's information systems? Since any technological means of protection may be compromised by 'wizardry' it is important that one engender an atmosphere of 'correctness' within the organization.

Computer Surveillance & Employee Records

In a 1931 speech, George Bernard Shaw observed:

An American has no sense of privacy. He does not know what it means. There is no such thing in the country.

At the time he may have been correct; however, the American society has matured during the last sixty years. Even though Supreme Court candidates have been unable to define the absolute nature of the rights of privacy on a constitutional basis, most Americans believe that they have a vested right of privacy based on the Fourth Amendment to the Constitution⁸. This for the most part protects us from our government.

Computerization of information systems has made the communication and dissemination of information about companies and individuals an accepted procedure. The issue of computer surveillance and employee records involves questions about the uses of databases that may involve invasion of privacy, either the customer's or employee's, and employee monitoring in the workplace. This latter involves the inclusion of a piece of software in the information system which monitors and times or otherwise measures the activities of operators. Is this a legitimate managerial exercise of administrative control, or is it an unwarranted intrusion into the employee's privacy? Put another way, should the firm legitimately be concerned only with the quantity and quality of the employee's activities, or may it also surreptitiously monitor the employee on a minute by minute basis? Questions of the impact on morale aside, how far may the manager extend his or her control over the activities of the employee? The sensitivity of this issue becomes more acute when the ability to control is magnified or enhanced by the computer's capacities. One other issue in this category involves the cross-reading or matching across information systems of employee or customer records. Again, the issue involves the right to privacy of employ-

8. Amendment IV Right of search and seizure regulated. The right of people to be secure in their persons, houses, papers, and effects against unreasonable searches and seizures and not be violated, and no warrants shall issue, but upon probable cause, supported by oath or affirmation and particularly describing the place to be searched, the persons or things to be seized.

ees and customers. Formerly, this may have been an ethical concern only in firm's large enough to have extensive databases. Today, even small organizations may have the computer capacity, or have access to databases that give the firm the capacity to intrude into the privacy of employees and customers.

The owner/manager of a small firm, then, is faced with many of the same ethical dilemmas that managers in large firms face. Dealing with the issues may be more difficult in that the small firm manager must be all things to all people, with little time for contemplating the complexities of the ethics of the computer age.

Gaming

An example of an issue of interest with perhaps least clear cut ethical stands is the use of company facilities for office games, such as 'rotisserie baseball' and 'fantasy hockey'. Employees face an ethical choice over the extent to which such 'enlivening' activities can legitimately be carried on during company time.

Managers face the need to balance productivity interests with maintaining a livable working environment that is not so rigid and controlling that the quality of work life drives off good employees.

SOURCES OF GUIDELINES AND CODES OF ETHICS

There are a not less than five organizations that have chosen to address directly the ethical issues posed by the rapid expansion of information technology they are:

- British Computer Society,
- Institute of Electrical and Electronic Engineers,
- Institute for Certification of Computer Professionals,
- CCP and
- The Data Processing Management Association.

The Data Processing Management Association (DPMA) has developed a code of ethics and a separate 'Standards of Conduct.'⁹

Standards of Conduct

These standards are derived from the code of ethics and are specific statements of behavior that no true professional will violate. Excerpts are provided below, as examples of ethical guidelines that are being developed by industry professionals:

In recognition of my obligation to management I shall:

- Not misuse the authority entrusted to me.
- Not misrepresent or withhold information concerning the capabilities of equipment, software or systems.

In recognition of my obligation to my fellow members and the profession I shall:

- Be honest in all my professional relationships.
- Not use or take credit for the work of others without specific acknowledgement and authorization.

In recognition of my obligation to society I shall:

- Protect the privacy and confidentiality of all information entrusted to me.
- To the best of my ability, insure that the products of my work are used in a socially responsible way.
- Never misrepresent or withhold information that is germane to a problem or situation of public concern nor will I allow any such known information to remain unchallenged.

9. *DPMA Code of Ethics*, Data Processing Management Association, 505 Bussie Highway, Park Ridge IL.

- Not use knowledge of a confidential or personal nature in any unauthorized manner or to achieve personal gain.

In recognition of my obligation to my employer I shall:

- Avoid conflict of interest and insure that my employer is aware of any potential conflict.
- Protect the privacy and confidentiality of all information entrusted to me.
- Not exploit the weakness of an information system for personal gain or personal satisfaction.

SUMMARY

If, due to security restrictions, an information system cannot disseminate its contents to those who need access, it fails. Technology alone does not solve the problem. It is a human problem.

It is of benefit to each user if everyone exercises discretion, judgment and professional respect for other's rights in the use of a computer information system. Each knows then that the system can be 'trusted.' It means that the system manager will be less concerned with intrusions or violations of rights and professional courtesies, respect and so on. But it also means that if an individual does desire to access another user's files, to change data, steal information, study someone else's personnel file, install a Trojan horse or release a virus, it is much easier to do so. The implicit trust in the system makes it easy for an individual user to violate that trust. Self-restraint thus can be seen as a prerequisite for any activity requiring trust.

The violation of the trust, if discovered, necessitates a higher level of administrative control, new restrictions placed on access, and that additional procedural processes be installed. The violations have caused a reduction in the efficiency and effectiveness of the system. A fundamental consideration for the manager, then, is to assess the role of trust, the desirable and achievable level of trust to be sought, and the implications of these choices for the firm and individuals affected.

This dilemma serves to highlight the ethical considerations facing the manager. For smaller organizations, it is further complicated by resource limitations, both financial and human. What balance between absolute confidence in the security of the system and completely free access for users is desirable? What are the tradeoffs between rights and responsibilities, costs and benefits implied by the security or control provisions that are contemplated? What values lie behind the choices made? As the level of security increases, and with it the consequent increase in the level of confidence or trust in the system, what other legitimate values are diminished or threatened? In general, this is the age-old question of the balance between individual and community interests. In specific terms, it is the question of how to optimize the legitimate and responsible use of computer information systems while eliminating unauthorized use and protecting the rights of users and other affected parties.

To generalize the issues raised here:

- If people will not exercise moral restraint, systems will develop controls for protection;
- The controls for protection will prove burdensome and inefficient;
- The systems will fail;
- They will still be necessary as the threat comes, not from responsible users but from 'mavericks' with what is arguably an essentially anti-community ethic;
- The systems will fail to be secure.

**"INFORMATION SECURITY RISK ANALYSIS AND RISK MANAGEMENT:
WHICH APPROACH ?"**

Prof. Jan H.P. Eloff

Karin P. Badenhorst

Department of Computer Science, Rand Afrikaans University

P.O Box 524, Johannesburg, 2000, Republic of South Africa

FACSIMILE: 27-11-489-2138 TELEPHONE: 27-11-489-2842

copyright 1991 EloffBadenhorst

ABSTRACT

The IRR research model as proposed in this paper can be seen as an important first phase of a research process, aimed at formulating a new approach to risk analysis, risk assessment and risk management within the information technology environment. Over the past decade, considerable resources and efforts have gone into developing and automating risk analysis methods, in an attempt to make risk analysis more easily applicable and as a whole more successful. This resulted in a large number of automated techniques, methods and packages being currently available on the information security software market. The perspective the authors took in preparing this paper was to address the question "Which approach combined with underlying business philosophies and business technologies ?" This opposes the question usually asked by organisations, namely "Which package ?"

KEYWORDS: risk analysis; risk assessment; risk management; risk resolution; information security methodology; information technology; environmental risk assessment; financial risk management.

0. INTRODUCTION

Information risk assessment is a vital business management task.[15] General managers usually have a high appreciation for risks relating to the continuation of their business. However, in practice the authors observed that a considerable amount of apprehension are still felt by many managers of organisations regarding the application of information technology risk analysis.

A fundamental issue of information security is rooted in the conflict between efficiency and control. This is exactly why risk analysis and risk management is such an important part of an overall information security function within an organisation. The objective of a risk analysis and risk management exercise is to find the optimum balance between efficiency, control and the cost of such control for an organisation. As management problems addressed in information security are usually more economic and politically based than technical, this should provide management with sufficient motive to conduct a risk analysis exercise.

Management approaches problems with subjective rather than objective solutions. On the other hand, risk analysis technology has traditionally focussed on objective or deterministic issues. Effective management should use risk analysis and risk management techniques in their proper role - as a management tool, not as a substitute for good judgement.[15]

The process of risk analysis and risk management in the context of information technology is concerned, firstly with the identification and measurement of risks related to information technology, and secondly with the control and minimisation of such risks. For the remainder of this paper we will refer to the process of information technology risk analysis and risk management as Information Risk Resolution (IRR).

1. IRR: INFORMATION RISK RESOLUTION

IRR has for a number of years been applied in the computer related industry without substantive rate of success. Research done since 1983 identified an increasing dissatisfaction with previously and currently available IRR methods and approaches.[16] Based on current literature and practical experience the authors came to some conclusions regarding IRR methods:

- It should be comprehensive in terms of handling all aspects of an IRR process, so that one does not have to apply more than one method and/or tool to accomplish meaningful results.[16] On the other hand it is the authors' opinion that IRR should not be so elaborate, that it defeats the other objective, namely to make it simpler and less time-consuming.

- It should also be comprehensive in terms of information security. It must be flexible enough to cover all aspects of computer and information security, as well as the interdependencies amongst those aspects.[20] The authors believe that IRR should be addressed from a multi-dimensional as well as a multi-disciplinary perspective. The multi-disciplinary concept stems from functional computer security levels (hardware, software, personnel, program controls, etc.). The interrelationships between tasks within these functional security levels (such as identifying threats related to the physical computer room, and determining the cost of logical access controls) constitute a multi-dimensional character.[1]
- The authors are of opinion that the assessment of risk is a functional rather than a financial issue. Evident from the application of IRR in organisations is the fact that IRR is usually performed by functions such as Audit (internal and/or external), and Finance.
- A method must be flexible enough to be "calibrated" to an environment. This also holds true for the maintainability of such method - it must respond to changes in the nature of a company's business.[9] The authors agree that it must be possible to customise an ideal IRR methodology for specific types of industry and varying management styles.
- A more qualitative and less quantitative method seemed preferable to refinements of existing quantitative methods.[16] The reasoning behind this is that quantitative figures can be misleading, because the fact that a figure is exact, does not necessarily mean that the assumptions on which the figure is based, are reliable.[25]
- Methods should reduce the amount of time, cost and overhead of performing an IRR. Such methods should therefor preferably be automated.[16]
- A risk analysis program should not be some arcane program applied on an ad hoc basis, when some unusual expense needs to be cost-justified. It should rather be an integral part of the business system.[7]
- It is very important that the method must be credible and trusted to the people that apply it, or those that rely on the results thereof.[9]

- IRR should always be applied within the perspective and as part of a comprehensive information security methodology. The likelihood of success will be greatly enhanced if it is not seen as a stand alone exercise.[2]

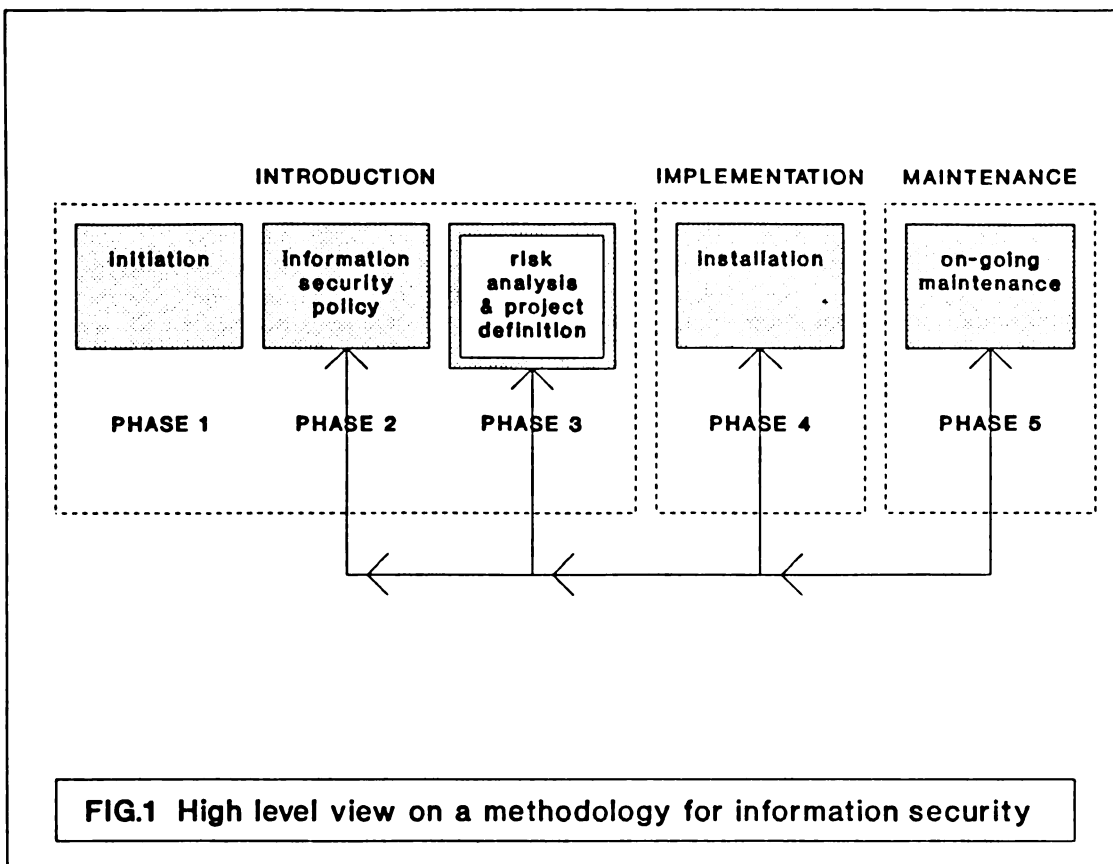
Consequently a number of automated packages were developed so as to make IRR "hopefully" more successful. This resulted in a large number of automated techniques, methods and packages being currently available on the software market.

CRAMM for example makes use of qualitative scalar techniques, whereas IRAM's quantification of risk is based on formal Bayesian probability theory and decision models.[20,14] MARION assesses business risks quantitatively and/or qualitatively making use of sophisticated mathematical and statistical principles.[19] RANK-IT is based on the so-called Delphi techniques, where expert opinion plays a major role in the assessment of risk.[12] LAVA, on the other hand, makes use of binary tree concepts - it uses hierarchical disaggregation structures to link questionnaires with event trees for vulnerability assessment.[26] It further makes use of linguistic algebra and fuzzy set theory. It is clear that divergent enabling techniques and approaches are used from one method to the next.

We are somewhat concerned that the root of the problem has not been addressed. To add to this problem, we are of opinion that information security risk analysis is a controversial issue amongst information security specialists, auditors, information technology managers, insurance surveyors and line managers, who respectively approaches IRR from an individual biased point of view. This statement is based on practical experience in major organisations as well as the overall impression gathered at international computer security congresses.[22]

The question to be addressed should not be "Which package to use?" but instead "Which approach combined with underlying business philosophies and business technologies?" This problem brought us to the idea of "What makes Risk Analysis successful in the general business or public context?" The authors then decided to attempt a new approach to IRR through investigation of existing risk analysis techniques and methods which normally turns out successfully in business functions outside the scope of information technology.

The above mentioned formulates the scope for the remainder of this paper.



The remainder of this paper therefor aims at addressing approaches to risk analysis as highlighted in phase 3 of the so-called IS-Methodology.

3. THE IRR PHASE

Gathered from a literature overview the following risk related business functions received considerable coverage regarding IRR:

- Environmental (especially health related risk)
- Engineering (especially nuclear risk)
- Finance (especially investment related risk)
- Insurance
- Computerised Business Information Systems (CBIS)

Many risk related functional philosophies inherent to appropriate **business functions** include risk management in financial terms and plant failure analysis in engineering terms. Within these business functions, various techniques are applied in the process of risk analysis and risk management, such as statistical short term forecasting techniques which are applied in financial risk management.[21] The concept of risk balancing is a technique used in environmental risk resolution. It is clear from the last mentioned that a technique plays an important part in the execution of IRR. The authors therefor decided to import the idea of "enabling **technologies**" (statistical short term forecasting techniques, risk balancing, heuristics, etc.), by so referring to the techniques inherent to specific approaches.

2. THE IS-METHODOLOGY

We believe that another factor that negatively influenced the success rate of IRR applications, is the fact that IRR is often attempted as a standalone exercise and on a piece meal, ad hoc basis. IRR should rather be placed within the context of an overall information security function in an organisation. The position of the IRR phase within an information security methodology can be seen in figure 1. The IS-Methodology as presented usually consists out of five phases:[1]

PHASE 1 - Initiation: The management of an organisation has to be committed to the need for an information security function. Such function should be initiated and guided by a steering committee.

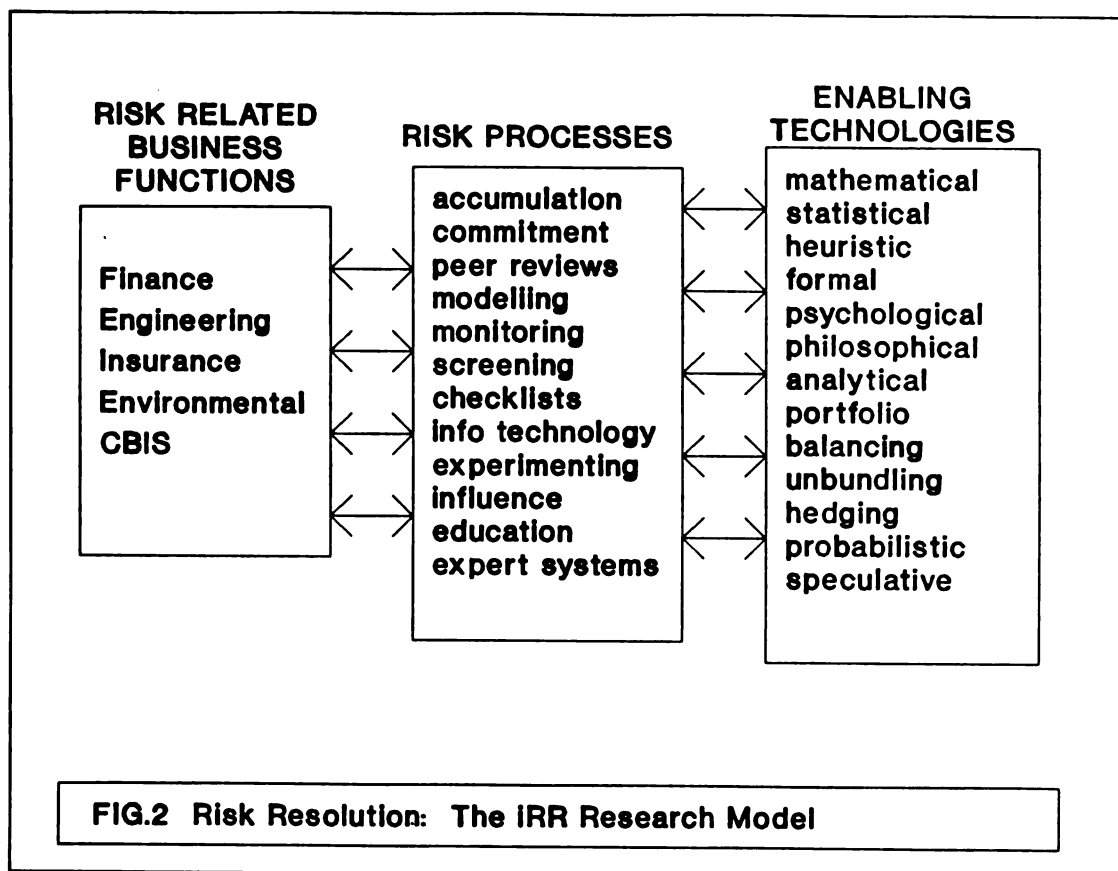
PHASE 2 - Information Security Policy: The definition and acceptance of a formal information security policy, which is in line with organisational strategies and company mission.

PHASE 3 - RISK ANALYSIS AND PROJECT DEFINITION: Information security risks and associated potential losses need to be determined (if possible quantified) and weighed against factors such as productivity, cost of controls, and benefit, in order to select cost-effective safeguards. The objective of this phase is a well-defined project plan for the installation of the acceptable level of safeguards.

PHASE 4 - Installation: The timely installation of the information security safeguards as set out in the project plan.

PHASE 5 - Maintenance: The on-going maintenance includes the review of the status of information security, on a regular basis. It also requires information security program controls to become part of the business analysis and systems development process.

Apart from the concepts business function and enabling technology one also has to address the issue of the means and utilities that are used with enabling technologies during the risk resolution process. Modelling, monitoring, screening, questionnaires and checklists, and computer technology are examples of such means used within business functions, referred to as risk processes in the context of our research. The following table illustrates the conceptual relationships between risk related business functions, risk processes and enabling technologies.



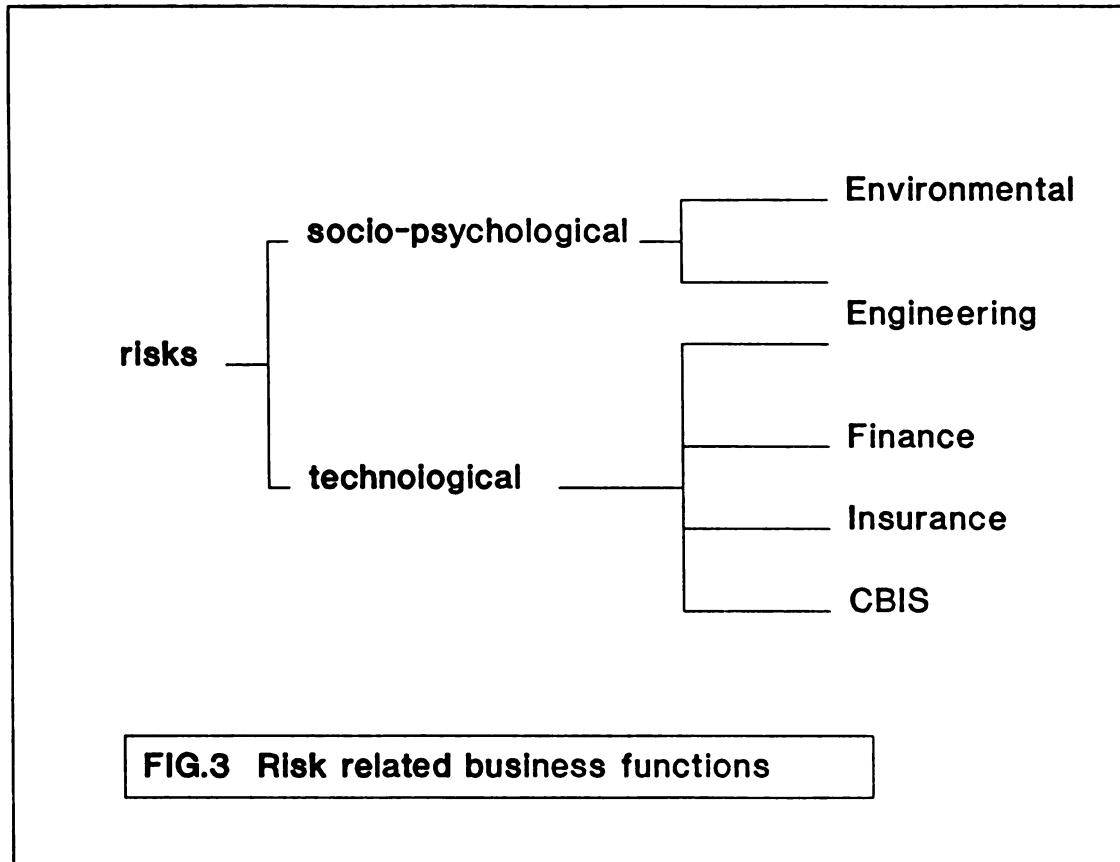
From the above diagram can be seen that the IRR Research Approach is made up of the following basic components:

Risk related business functions,
Risk processes, and
Risk resolving enabling technologies.

More than one enabling technology and risk process might be used within each risk related business function, as shown by the literature overview undertaken by the IRR project team. Quantitative methods, for example, which are statistical and mathematical in origin, are applied in the general management process to reduce the risk involved in strategic decision-making. Such a risk resolution exercise can be further facilitated by means of processes such as spreadsheets and computer technology. In the same way more than one business function, enabling technology and risk process might be used within the application of the IRR methodology.

4. RISK RELATED BUSINESS FUNCTIONS

From a literature viewpoint, risks are generally classified into one of two categories, namely (i) risks related to business functions usually associated with socio-psychological issues, and (ii) risks related to technological issues, as indicated by the following diagram:



The risk related business functions Engineering, Insurance and CBIS will be defined and briefly discussed. Enabling technologies and risk processes within the Environmental and Financial business functions will be discussed in more detail.

4.1 ENGINEERING

From current literature it is clear that risk analysis and risk management in the engineering environment is mostly concerned with system or plant failure analysis. Coverage of nuclear engineering risk assessment constitutes a major part of workshops, seminars, research and subsequent literature. Probabilistic event trees and fault trees are the most prominent enabling technologies applied with regard to risk resolution in the engineering environment. Other approaches are mostly analytical in origin.[23]

4.2 INSURANCE

The theory of risk associated with the insurance industry date as far back as 1909. This classical theory was then mostly associated with life insurance mathematics. The theory of risks has since been expanded to include not only short term insurance risks and other aspects of the insurance business such as reinsurance, but also risks related to strategic decision making in general financial business planning. Enabling techniques applied in insurance risk theory include amongst others, stochastic processes, the time-dependent variation of risk exposure, and the Monte Carlo technique.[4]

4.3 COMPUTERISED BUSINESS INFORMATION SYSTEMS (CBIS)

Software risk management is an emerging discipline whose objectives are to identify, address, and eliminate software risk items. Such risk factors could become either threats to the successful operation of software, or result in major rewrites of software. Enabling technologies and risk processes used in software risk management include amongst others, network analysis, decision trees, risk exposure analysis, the Delphi technique, statistical decision analysis, checklists, cost and performance models.[5]

4.4 ENVIRONMENTAL

Environmental risk is a hazard or danger which threatens the environment, for example the risk of a nuclear accident caused by human error or by natural disaster. It is the probability or chance of an environment (i.e. human, nature, etc.) suffering an adverse consequence, or of encountering some loss. Environmental risk management involves the search for a 'best route' between social benefit and environmental risk. It is a balancing or trading-off process in which various combinations of risks are compared and evaluated against particular social gains.

Risk research has been sponsored by industry and government in many countries, largely because public opposition to some technological development has created powerful constraints on further expansion in, for example, the nuclear power industry.[27,25,24]

4.4.1 ENVIRONMENTAL: THE IMPORTANCE FOR IRR

Risk identification and risk estimation are steps in an environmental risk analysis exercise. Some risk processes used within risk identification and estimation include modelling, monitoring (surveillance), testing and screening. Enabling technologies include psychological perception, quantitative techniques, heuristics, balancing of risks, probabilistic binary event tree analysis, the concept of reasonableness, and risk rationalisation versus risk reduction.

The following aspects have been addressed in literature on environmental risk assessment. The utilisation of these as enabling technologies in IRR seems interesting and possible:

PSYCHOLOGICAL PERCEPTION

Psychological impact results in social and environmental risk perception to differ greatly from one person to another.[25]

In studies on judgements of positive versus negative values, it has been shown that values guiding our behaviour are more negative on the negative side than positive on the positive side. This means that we are generally more sensitive to increases in loss (i.e. negative risk) than to increases in gains (i.e. positive risk).[25]

In the balance between gains (positive risks) and potential losses (negative risks), or efficiency (positive risk) and risk resolution (controls for negative risk), does the above statement with respect to psychological impact hold true for the IRR environment? Do information technology managers also regard the risk of the loss of a computer service as having 'more value' than the actual economic benefit of utilising the best information technology to provide a service?

QUANTITATIVE TECHNIQUES

Difficulty in the quantification of environmental risks is often experienced. There are plenty of examples of risk estimates which are often quoted (e.g. the risk of a disaster at a nuclear power plant), which can very well be uncertain by a factor of 100 or 1000. But as soon as a figure is given, many people tend to forget this and accept the figure as a fact. Quantification of risks in IRR methods have for some time been treated with the same kind of scepticism. This resulted in current research to be aimed at qualitative rather than quantitative approaches.

HEURISTICS

Heuristics have been applied in simplifying environmental risk analysis. This resulted in conclusions being deficient. It is often quite debatable if such conscious deficiency is justified.[25] The same reasoning would apply to IRR. Two thinking paradigms have been identified in the field of IRR, namely rational/analytical versus intuitive/heuristic.[6] It is the author's opinion that the former is obviously more technical whereas the latter is heavily influenced by psychological perception. A general distinction has been made in literature between risk analysis approaches as being either technological or psychological.[18]

Finance, which has been identified as another major risk related business function, will be discussed next.

4.5 FINANCE

In organisations, risk management in the narrow sense has been dealing with the organisational aspects of assessing and limiting risk. Pure risks are limited to events with detrimental consequences to a company, such as risks threatening assets, labour potential or financial potential of a company and are the result of accidental and probable events. In contradistinction there are speculative risks, which involve the possibility of both gain and loss. The resolution of the latter is usually understood as being financial risk management.[3]

Any financial instrument used within the financial business function, can be viewed as having a unique combination of characteristics, such as yield, duration, size, marketability, and inherent risk profile. Such risk profiles go hand in hand with financial innovation. Financial transactions reallocate various categories of risk among lenders, borrowers and financial intermediaries. The inherent risks associated with finance, include price (market) risk, credit risk, liquidity risk, settlement risk, country and transfer risk, and the investment risk associated with stock trading.[10]

Enabling technologies applied in financial risk management include techniques such as strategic switch analysis, duration and maturity gap analysis, immunisation, portfolio techniques, the unbundling of risks, and quantitative decision tree modelling.

4.5.1 FINANCIAL: THE IMPORTANCE FOR IRR

SWITCH ANALYSIS

Switch analysis is a technique whereby a switch transaction takes place, i.e. the selling of a stock in a portfolio and the simultaneous purchase of a different stock. Owing to different stock volatilities, some stocks will appear to offer better value than others given a particular "view" on interest rates, thus reducing possible negative risk associated with a portfolio of stocks.[10]

Financial switch analysis and the environmental balancing of risks are similar in concept, as they both try to minimise risk to an optimum level. Within the IRR process, instead of reducing risks by means of costly safeguards, why not use the concept of switching by comparing risks and replacing risks with suitable alternatives ?

MATURITY GAP ANALYSIS

Maturity gap analysis is a flow concept exclusively used for interest rate risk management, while duration, as a stock concept, embraces interest rate, investment, and capital risk analysis. Duration and maturity gap analysis may help a bank to fashion financial strategies for the current, or next, financial year that will give it the accounting profits it needs.

There also seem to be some similarity between the time-dependent variation of risk exposure used in insurance risk theory and the duration and maturity gap analysis techniques used in financial risk management, as both involve time factors. The time-change factor also plays an important role in IRR, because of the dynamic character of the information technology environment.

PORTFOLIO THEORY

The central idea of portfolio theory is that the total risk of an investment can be reduced by spreading it over a pool of assets.[3]

In the application of financial portfolio techniques, the application of a risk reducing measure is comparable to an investment in an asset. If the security of certain values is based on a single measure, the total of values at risk is exposed if the measure fails. If, however, a combination of measures, viz. a portfolio of measures, has been applied, the failure of an individual component will still result in a reduced risk.[3]

5. CONCLUSION

In this paper the question "Which approach combined with underlying business philosophies and business technologies ?" instead of "Which package ?" has been addressed, because the authors felt that research into underlying business philosophies related to risk analysis could contribute in resolving the dilemma that so often governs the application of IRR. The authors also strongly support the concept that IRR should be placed within the context of an overall information security methodology, such as the IS-Methodology.

The basic components of the IRR research model have been identified as: Business Functions, Risk Processes and Enabling Technologies. The business functions Environmental and Finance have been discussed so to demonstrate the applicability of these concepts to the issues surrounding Information Risk Resolution. The discussion on Environmental risk analysis appears to be very appropriate to the much discussed topic of Disaster Recovery Planning for the computer facilities of an organisation. The possibility of applying some of these enabling technologies in IRR raises the question: how can they be adapted for the information technology environment ? The last mentioned requires further research and will be reported on in a follow-up paper.

REFERENCES

- [1] Badenhorst K.P. & Eloff Jan H.P., "Framework of a Methodology for the Life Cycle of Computer Security in an Organisation", *Computers & Security*, 8 (1989) 433-442, Elsevier Science Publishers Ltd.
- [2] Badenhorst K.P. & Eloff Jan H.P., "Computer Security Methodology: Risk Analysis and Project Definition", *Computers & Security*, 9 (1990) 339-346, Elsevier Science Publishers Ltd.
- [3] Bauknecht Kurt & Strauss Christine, "Portfolio techniques to Support Risk Management and Security", IFIP/Sec 90 on 'Computer Security and Integrity in our Changing World', The 6th International Conference and Exhibition on Information Security, Espoo (Helsinki), Finland, May 23-25 1990.
- [4] Beard R.E., Pentikäinen T. & Pesonen E., "Risk Theory, The Stochastic Basis of Insurance", Third Edition, Chapman and Hall, London, 1984.
- [5] Boehm Barry W., "Software Risk Management", IEEE Computer Society Press, Washington D.C., 1989.
- [6] Caelli William J., "Information Security: The Next Decade", Seminar organised by Computer Society of South Africa in conjunction with IFIP Committee (TC11), Cape Town, South Africa, 18 May 1990.
- [7] Carroll John M., "Coding Ethics and Law into Risk Analysis", Proceedings of Compsec 90 International, London, 10-12 October 1990.
- [8] Crouch Edmund A.C. & Wilson Richard, "Risk/Benefit Analysis", Ballinger Publishing Company, Cambridge Massachusetts, 1982.
- [9] Dorey Dr Paul G, "Computer Security Risk Management: Practical Experiences of a User", Proceedings of Compsec 90 International, London, 10-12 October 1990.
- [10] Falkena H.B. & Kok W.J. (Ed.), "Essays on Financial Risk Management", Macmillan Press Ltd, London, 1988.
- [11] Gardner Philip E., "Evaluation of Five Risk Assessment Programs", *Computers & Security*, 8 (1989) 479-485, Elsevier Science Publishers.
- [12] Gilbert Irene E., "Automated Risk Management Software Tools", National Computer Systems Laboratory & Computer Security Management and Information Exchange Group, National Institute of Standards and Technology, Gaithersburg, Maryland, U.S.A., 1990.
- [13] Godfrey A.I., "Quantitative Methods for Managers", Edward Arnold Publishers Ltd, London, 1977.

- [14] Guarro Sergio B., "Principles and Procedures of the IRAM Approach to Information Systems Risk Analysis and Management", Computers & Security, 6 (1987) 493-504, Elsevier Science Publishers.
- [15] Hutt A.E., Bosworth S. & Hoyt D.B., "Computer Security Handbook", Second Edition, Macmillan Publishing Company, New York, 1988, pp.22-32, 299.
- [16] Katzke Dr. Stuart W., "NBS Perspectives on Risk Analysis: Past, Present and Future", from Minutes of the Federal Informaiton Systems Risk Analysis Workshop, The Air Force Computer Security Program Office, U.S.A., 1985, pages 2.3-2.5.
- [17] Kunreuther Howard (Ed.), "Risk: A Seminar Series", IIASA Collaborative Proceedings Series, CP-82-S2, International Institute for Applied Systems Analysis, Laxenburg Austria, 1982.
- [18] Kunreuther Howard C. & Ley Eryl V (Ed.), "The Risk Analysis Controversy: An Institutional Perspective", Proceedings of a Summer Study on Decision Processes and Institutional Aspects of Risk held at IIASA, Laxenburg, Austria, 22-26 June 1981, Springer Verlag Berlin 1982.
- [19] Lamère J.-M., Leroux Y. & Tourly J., "La Sécurité des Réseaux, Méthodes at Techniques", Dunod Informatique, Bordas, Paris 1987.
- [20] Moses Robin H & Glover Ian, "A Model of Risk Analysis and Management", Proceedings of the 2nd Annual Canadian National Computer Security Conference, Ottawa, March 1990.
- [21] O'Donovan T.M., "Short Term Forecasting: An Introduction to the Box-Jenkins Approach" John Wiley & Sons Ltd., Chichester, 1983.
- [22] Proceedings: IFIP/Sec 90 (Espoo Helsinki Finland) 23-25 May 1990, Compsec 90 (London U.K.) 10-12 October 1990, 13th National Computer Security Conference (Washington D.C. U.S.A.) 1-4 October 1990.
- [23] Proceedings of the International Ans/Ens Topical Meeting on PROBABILISTIC RISK ASSESSMENT - September 20-24, 1981, Port Chester, New York. Sponsored by AMERICAN NUCLEAR SOCIETY, EUROPEAN NUCLEAR SOCIETY.
- [24] Shrader-Frechette K.S., "Risk Analysis and Scientific Method", D.Reidel Publishing Company, Dordrecht 1985.
- [25] Sjöberg Lennart (Ed.), "Risk and Society", The Risks and Hazards Series: 3, Allen & Unwin (Publishers) Ltd., London, 1987, p 156.
- [26] Smith S.T. & Lim J.J., "Framework for Generating Expert Systems to Perform Computer Security Risk Analysis", First Annual Armed Forces Communications and Electronics Association Symposium and Exposition on Physical and Electronics Security, Philadelphia, August 19-21, 1985.
- [27] Whyte Anne V. & Burton Ian (Ed.), "Environmental Risk Assessment", Scope 15, John Wiley and Sons, Toronto, 1980.

INFORMATION SYSTEMS SECURITY: A COMPREHENSIVE MODEL

Capt John R. McCumber
Joint Staff/J6K
The Pentagon
Washington, DC 20318-6000

INTRODUCTION

At speech to the 13th National Computer Security Conference on 3 October 1990, Michelle VanCleave, Assistant Director for National Security Affairs, Executive Office of the President stated, "We need a comprehensive model for understanding the threat to our automated information systems." I believe I have developed that model. This model not only addresses the threat, it functions as an assessment, systems development, and evaluation tool. The model is unique in that it stands independent of technology. Its application is universal and is not constrained by organizational differences. As with all well-defined fundamental concepts, it is unnecessary to alter the premise even as technology and human understanding evolve.

Computers communicate. Communication systems compute. The evolution of technology has long since eliminated any arbitrary distinction between a computer and its communication components or a communications network and its computing system. Some organizations have attempted to deal with the phenomenon by marrying these functions under common leadership. This has resulted in hyphenated job descriptions such as Computer-Communications Systems Staff Officer and names like Information Technology Group. Unfortunately, these names can mask an inappropriate or poorly executed realignment of organizational responsibilities. Ideally, management will recognize there is a theoretical-as well as organizational-impact.

The same is true for the security disciplines. Merely combining the communications security (COMSEC) and computer security (COMPUSEC) disciplines under an umbrella of common management is unacceptable. Even if we address the other, albeit less technical, aspects of information systems security such as policy, administration, and personnel security, we still fail to develop a comprehensive view of this evolving technology. The reason for this becomes clear when we are reminded it's the information that is the cornerstone of information systems security. In this sense, any paradigm which emphasizes the technology at the expense of information will be lacking.

THE NATURE OF INFORMATION

Defining the nature of information could be a tedious task. To some it represents the free-flowing evolution of knowledge; to others, it is intelligence to be guarded. Add to this the innumerable media through which information is perceived and we have a confusing array of contradictions. How can we present a study of information that has universal application?

It may be best to develop a simple analogy. The chemical compound H_2O means many things to all of us. In its liquid state, water means life-giving sustenance to a desert-dwelling Bedouin; to a drowning victim, it is the vehicle of death. The same steam we use to prepare vegetables can scald an unwary cook. Ice can impede river-borne commerce on the Mississippi River or make a drink more palatable. Science, therefore, does not deal with the perception of the compound, but with its state.

As the compound H_2O can be water, ice, or steam, information has three basic states which I've already depicted. At any given moment, information is being transmitted, stored, or processed. The three states exist irrespective of the media in which information resides. This subtle distinction ultimately allows us to encompass all information systems technology in our model.

It is possible to look at the three states in microcosm and say that processing is simply specialized state combinations of storage and transfer; so, in fact, there are only two possible states. By delving to this level of abstraction, however, we go beyond the scope and purpose of the model. The distinction between the three states is fundamental and necessary to accurately apply the model. For example, cryptography can be used to protect information while it's transferred through a computer network and even while it is stored in magnetic media. However, the information must be available in plaintext (at least to the processor) in order for the computer to perform the processing function. The processing function is a fundamental state which requires specific security controls.

When this information is needed to make a decision, the end user may not be aware of the number of state changes effected. The primary concern will be certain characteristics of the information. These characteristics are intrinsic and define the security-relevant qualities of the information. As such, they are the next major building block of our information systems security model.

CRITICAL INFORMATION CHARACTERISTICS

Information systems security concerns itself with the maintenance of three critical characteristics of information: confidentiality (Pfleeger's "secrecy"), integrity, and availability [PFL89]. These attributes of information represent the full spectrum of security concerns in an automated environment. They are applicable for any organization irrespective of its philosophical outlook on sharing information.

CONFIDENTIALITY

Confidentiality is the heart of any security policy for an information system. A security policy is the set of rules that, given identified subjects and objects, determines whether a given subject can gain access to a specific object [DOD85]. In the case of discretionary access controls, selected users (or groups) are controlled as to which data they may access. Confidentiality is then the assurance that access controls are enforced. The reason

I prefer the term confidentiality to secrecy is merely to avoid unwarranted implications that this is solely the domain of armies and governments. As we will see, it is a desirable attribute for information in any organization.

All organizations have a requirement to protect certain information. Even owners of a clearinghouse operation or electronic bulletin need the ability to prevent unwanted access to supervisory functions within their system. It's also important to note the definition of data which must be protected with confidentiality controls is broadening throughout government [OTA87]. Actual information labeling and need-to-know imperatives are aspects of the system security policy which are enforced to meet confidentiality objectives. The issue of military versus civilian security controls is one which need not impact the development of a comprehensive representation of information systems security principles.

INTEGRITY

Integrity is perhaps the most complex and misunderstood characteristic of information. As I stated, we seem to have a better foundation in the development of confidentiality controls than those which can help insure data integrity. Pfleeger defines integrity as "assets (which) can only be modified by authorized parties" [PFL89]. Such a definition unnecessarily confines the concept to one of access control.

I propose a much broader definition. Data integrity is a matter of degree (as is the concept of "trust" as applied to trusted systems) which has to be defined as a quality of the information and not as who does/does not have access to it. Integrity is that quality of information which identifies how closely the data represent reality. How closely does your resume reflect "you"? Does a credit report accurately reflect the individual's historical record of financial transactions? The definition of integrity must include the broad scope of accuracy, relevancy, and completeness.

Data integrity calls for a comprehensive set of aids to promote accuracy and completeness as well as security. This is not to say that too much information can't be a problem. Data redundancy and unnecessary records present a variety of challenges to system implementors and administrators. The users must define their needs in terms of the information necessary to perform certain functions. Information systems security functions help insure this information is robust and (to the degree necessary) reflects the reality it is meant to represent.

AVAILABILITY

Availability is a coequal characteristic with confidentiality and integrity. This vital aspect of security insures the information is provided to authorized users when it's requested or needed. Often it's viewed as a less technical requirement which is satisfied by redundancies within the information system such as back-up power, spare data channels, and parallel data bases. This perception,

however, ignores one of the most valuable aspects of our model which this characteristic provides. Availability is the check-and-balance constraint on our model. Because security and utility often conflict, the science of information systems security is also a study of subtle compromises.

As well as insuring system reliability, availability acts as a metric for determining the extent of information system security breaches [DOJ88]. Ultimately, when information systems security preventive measures fail, remedial action may be necessary. This remedial activity normally involves support from law enforcement or legal departments. In order to pursue formal action against people who abuse information systems resources, the ability to prove an adverse impact often hinges on the issue of denying someone the availability of information resources. Although violations of information confidentiality and integrity can be potentially more disastrous, denial of service criteria tend to be easier to quantify and thus create a tangible foundation for taking action against violators [CHR90].

The triad of critical information characteristics covers all aspects of security-relevant activity within the information system. By building a matrix with the information states positioned along the horizontal axis and the critical information characteristics aligned down the vertical, we have the foundation for the model.

SECURITY MEASURES

We've now outlined a matrix which provides us with the theoretical basis for our model. What it lacks at this stage is a view of the measures we employ to insure the critical information characteristics are maintained while information resides in or moves between states. It's possible, at this point, to perceive the chart as a checklist. At a very high level of abstraction, one could assess the security posture of a system by using this approach. By viewing the interstices of the matrix as a system vulnerability, you can attempt to determine the security aspects of an information system as categorized by the nine intersection areas. For example, you may single out systems information confidentiality during transmission or any intersection area for scrutiny.

The two-dimensional matrix also has another less obvious utility. We can map various security technologies into the nine interstices. Using our example from above, we note it is necessary to protect the confidentiality of the information during its transmission state. We can then determine which security technologies help insure confidentiality during transmission of the information. In this case, cryptography would be considered a primary security technology. We can then place various cryptographic techniques and products within a subset in this category. Then we repeat the process with other major types of technology which can be placed within this interstice. The procedure is repeated for all nine blocks on our grid. Thus we form the first of three layers which will become the third dimension of our model-security measures.

TECHNOLOGY

The technology layer will be the primary focus of the third dimension. We will see that it provides the basis for the other two layers. For our purposes, we can define technology as any physical device or technique implemented in physical form which is specifically used to help insure the critical information characteristics are maintained through any of the information states. Technology can be implemented in hardware, firmware, or software. It could be a biometric device, cryptographic module, or security-enhanced operating system. When we think of a thing which could be used to protect the critical characteristics of information, we are thinking of technology.

Usually, organizations are built around functional responsibilities. The advent of computer technology created the perception that a group needed to be established to accommodate the new machines which would process, store, and transmit much of our vital information. In other words, the organization was adapted to suit the evolving technology. Is this wrong? Not necessarily; however, it is possible to create the impression that technology exists for technology's sake. Telecommunications and computer systems are simply media for information. The media need to be adapted to preserve certain critical characteristics with the adaptation and use of the information media (technology). Adaptation is a design problem, but use and application concerns bring us to the next layer.

POLICY AND PRACTICE

The second layer of the third dimension is that of policy and practice. It's the recognition of the fact that information systems security is not just a product which will be available at some future date. Because of our technology focus, it's easy to begin to think of security solutions as devices or add-on packages for existing information systems. We are guilty of waiting for technology to solve that which is not solely a technological problem. Having an enforceable (and enforced) policy can aid immeasurably in protecting information.

A study has shown 75% of Federal agencies don't have a policy for the protection of information on PC-based information systems [OTA87]. Why, if it is so effective, is policy such a neglected security measure? It may be due in part to the evolving social and moral ethic with regard to our use of information systems. The proliferation of unauthorized software duplication is just another symptom of this problem. Even though software companies have policies and licensing caveats on their products, sanctions and remedies allowed by law are difficult if not impossible to enforce. No major lawsuit involving an individual violator has come before our courts, and it appears many people don't see the harm or loss involved. Although there are limits established by law, it seems we as "society" accept a less stringent standard.

Closely associated with the matter of policy is that of practice. A practice is a procedure we employ to enhance our

security posture. For example, we may have a policy which states that passwords must be kept confidential and may only be used by the uniquely-authenticated user. A practice which helps insure this policy is followed would be committing the password to memory rather than writing it somewhere.

The first two layers of the third dimension represent the design and application of a security-enhanced information system. The last building block of our model represents the understanding necessary to protect information. Although an integral aspect of the preceding two layers, it must be considered individually as it is capable of standing alone as a significant security measure.

EDUCATION, TRAINING, AND AWARENESS

The final layer of our third dimension is that of education, training, and awareness. As you will see, were the model laid on its back like a box, the whole model would rest on this layer. This phenomenon is intentional. Education, training and awareness may be our most prominent security measures, for only by understanding the threats and vulnerabilities associated with our proliferating use of automated information systems can we begin to attempt to deal effectively with other control measures.

Technology and policy must rely heavily on education, training, and awareness from numerous perspectives. Our upcoming engineers and scientists must understand the principles of information security if we expect them to consider the protection of information in the systems they design. Currently, nearly all university graduates in computer science have no formal introduction to information security as part of their education [HIG89].

Those who are responsible for promulgating policy and regulatory guidance must place bounds on the dissemination of information. They must insure information resources are distributed selectively and securely. The issue is ultimately one of awareness. Ultimate responsibility for its protection rests with those individuals and groups which create and use this information; those who use it to make critical decisions must rely on its confidentiality, integrity, and availability. Education, training, and awareness promises to be the most effective security measure in the near term.

Which information requires protection is often debated in government circles. One historic problem is the clash of society's right to know and an individual's right to privacy. It's important to realize that these are not bipolar concepts. There is a long continuum which runs between the beliefs that information is a free flowing exchange of knowledge and that it is intelligence which must be kept secret. From a governmental or business perspective, it must be assumed that all information is intelligence. The question is not should information be protected, but how do we intend to protect the confidentiality, integrity, and availability of it within legal and moral constraints?

THE MODEL

OVERVIEW

The completed model appears as Figure 1. There are nine distinct interstices, each three layers deep. All aspects of information systems security can be viewed within the framework of the model. For example, we may cite a cryptographic module as technology which protects information in its transmission state. What many information system developers fail to appreciate is that for every technology control there is a policy (sometimes referred to as doctrine) which dictates the constraints on the application of that technology. It may also specify parameters which delimit the control's use and may even cite degrees of effectiveness for different applications. Doctrine (policy) is an integral yet distinct aspect of the technology. The third layer-education, training, and awareness-then functions as the catalyst for proper application and use of the technology based on the policy (practice) application.

Not every security measure begins with a specific technology. A simple policy or practice often goes a long way in the protection of information assets. This policy or practice is then effected by communicating it to employees through the education, training, and awareness level alone. This last layer is ultimately involved in all aspects of the information systems security model. It may also be solely an educational, training, or awareness security control. The model helps us understand the comprehensive nature of information security that a COMSEC/COMPUSEC perspective cannot define.

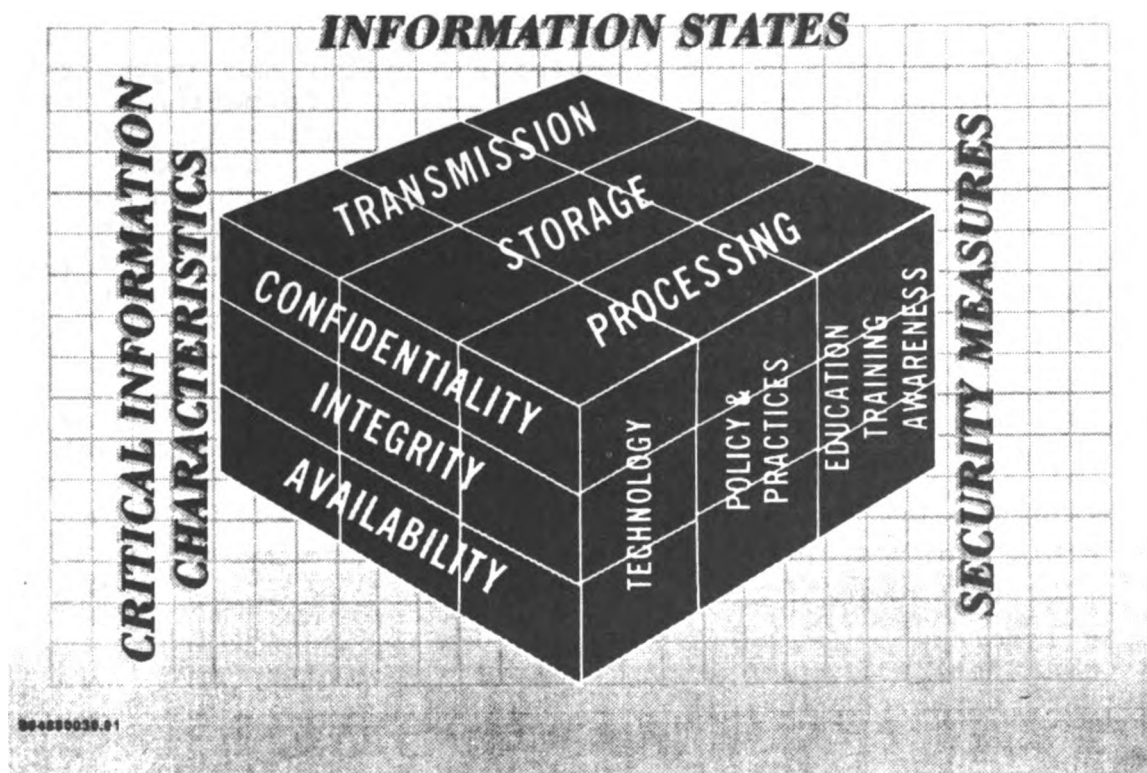


Figure 1

USE OF THE MODEL

The model has several significant applications. Initially, the two-dimensional matrix is used to identify information states and system vulnerabilities. Then, the three layers of security measures can be employed to minimize these vulnerabilities based on a knowledge of the threat to the information asset. Let's take a brief look at these applications.

A developer would begin using the model by defining the various information states within the system. When an information state is identified, one then works down the vertical path to address all three critical information characteristics. Once vulnerabilities are noted in this fashion, it becomes a simple matter of working down through the three layers of security measures. If a specific technology is available, the designer knows that policy and practice as well as education, training, and awareness will be logical follow-on aspects of that control. If a technology cannot be identified, then policy/practice must be viewed as the next likely avenue. (Again, the last layer will be used to support the policy/practice.) If none of the first two layers can satisfactorily counter the vulnerability then, as a minimum, an awareness of the weakness becomes important and fulfills the dictates of the model at the third layer.

Another important application is realized when the model is used as an evaluation tool. As in the design and development application, the evaluator first identifies the different information states within the system. These states can be identified separately from any specific technology. A valuable aspect of the model is the designer needn't consider the medium.

After identifying all the states, an evaluator or auditor can perform a comprehensive review much the same way the systems designer used the model during the development phase. For each vulnerability discovered, the same model is used to determine appropriate security measures. The third dimension of the model insures the security measures are considered in their fullest sense. It is important to note that a vulnerability may be left unsecured (at an awareness level in the third layer) if the designer or evaluator determines no threat to that vulnerability exists. Although no security practitioner should be satisfied with glaring vulnerabilities, a careful study of potential threats to the information may disclose that the cost of the security measure is more than the loss should the vulnerability be exploited. This is one of the subtle compromises alluded to earlier.

The model can also be used to develop comprehensive information systems security policy and guidance necessary for any organization. With an accurate understanding of the relation of policy to technology and education, training, and awareness, you can insure your regulations address the entire spectrum of information security. It's of particular importance that corporate and government regulations not be bound by technology. Use of this model allows management to structure its policy outside the

technology arena.

The model functions well in determining requirements for education, training, and awareness. Since this is the last layer, it plays a vital role in the application of all the security measures. Even if a designer, evaluator, or user determines to ignore a vulnerability (perhaps because of a lack of threat), then the simple acknowledgement of this vulnerability resides in the last layer as "awareness". Ultimately, all technology, policies, and practices must be translated to the appropriate audience through education, training, and awareness. This translation is the vehicle which makes all security measures effective. For a more complete understanding of the nuances of education, training, and awareness see [MAC89].

The twenty-seven individual "cubes" created by the model can be extracted and examined individually. This key aspect can be useful in categorizing and analyzing countermeasures. It's also a tool for defining organizational responsibility for information security. The example shows a policy security measure for protecting the confidentiality of information while it is being processed. By considering all 27 such "cubes", the analyst is assured of a complete perspective of all available security measures. Unlike other computer security standards and criteria, this model connotes a true "systems" viewpoint.

CONCLUSION

The information systems security model acknowledges information, not technology, as the basis for our security efforts. The actual medium is transparent in the model. This eliminates unnecessary distinctions between COMSEC, COMPUSEC, TECHSEC, and other technology-defined security sciences. As a result, we can model the security relevant processes of information throughout an entire information system-automated or not. This important aspect of the model eliminates significant gaps in currently-used security architecture guidance for information systems.

I developed this model to respond to the need for a theoretical foundation for modeling the information systems security sciences. The organizational realignments which have recognized the interdependence of several complementary technologies will need refinement in the near future. We can begin that process now by acknowledging the central element in all our efforts-information. Only when we build on this foundation will we accurately address the needs of information systems security in the next decade and beyond.

REFERENCES

- [CHR90] Interview with Agent Jim Christy, Chief, Air Force Office of Special Investigations, Computer Crime Division, 26 March 1990
- [DOD85] Department of Defense Trusted Computer System Evaluation Criteria, DoD 5200.28-STD, Department of Defense, Washington, DC, December 1985
- [DOJ88] Basic Considerations in Investigating and Proving Computer-Related Federal Crimes, U.S. Department of Justice, Justice Management Division, Washington, DC, November 1988
- [HIG89] Higgins, John C., Information Security as a Topic in Undergraduate Education of Computer Scientists, Proceedings of the 12th National Computer Security Conference, November 1989
- [MAC89] Maconachy, W.V., Computer Security Education, Training, and Awareness: Turning a Philosophical Orientation into Practical Reality, Proceedings of the 12th National Computer Security Conference, November 1989
- [OTA87] U.S. Congress, Office of Technology Assessment, Defending Secrets, Sharing Data: New Locks and Keys for Electronic Information, OTA-CIT-310, Washington, DC: U.S. Government Printing Office, October 1987
- [PFL89] Pfleeger, Charles P., Security in Computing, Prentice-Hall, 1989

INTEGRATING B2 SECURITY INTO A UNIX SYSTEM

Kevin Brady

UNIX System Laboratories, Inc.
190 River Road, Summit NJ 07901

Overview

Within the last few years the integrity of many computer systems has been violated in a variety of ways, the most prevalent of which has been via "virus" attacks. These attacks feature software which, either intentionally or accidentally, result in a compromise of system security and subsequently result in hundreds of thousands of dollars of damage in the form of compromised/lost data or computer downtime. Currently, most attacks are detected long after the fact. Unfortunately, by the time the intrusion is detected, significant damage is done. In the case of a virus, it is likely to have spread throughout an entire network of computers.

With the advent of systems containing additional security features such as access control lists, least privilege, and mandatory access control, the question arises, do these systems meet the challenge of preventing system security violations and containing virus programs while still retaining the "look and feel" of a traditional UNIX system?

This paper focuses on the features added to UNIX System V Release 4.1 Enhanced Security (SVR4.1ES) intended to raise the overall level of system security to the B2/F-B2 level.

1. Motivation

By the late 1980's, increased concerns regarding the privacy of computerized data, fear of unauthorized access, and concerns regarding system and data integrity led to a demand within the UNIX community for a higher level of system security. This in turn led to the inclusion of enhanced security features such as those present in SVR4.1ES. While the model for some enhanced features, such as mandatory access control (MAC), have their origins with the Trusted Computer Systems Evaluation Criteria (TCSEC), many others, such as discretionary access control, represent extensions of existing features within the UNIX system. The combination of these features, specifically least privilege and enhanced access control (MAC & DAC), not only provides an environment that is more resistant to penetration and compromise than the UNIX systems that preceded it but also provides compatibility for existing applications and retains the "look and feel" of the UNIX system.

The following is a brief discussion of the approach used for feature definition followed by a description of the key features found in the SVR4.1ES system; Least Privilege/Trusted Facility Management, Enhanced Access Control (Mandatory & Discretionary), Trusted Path, and Auditing.

2. Least Privilege

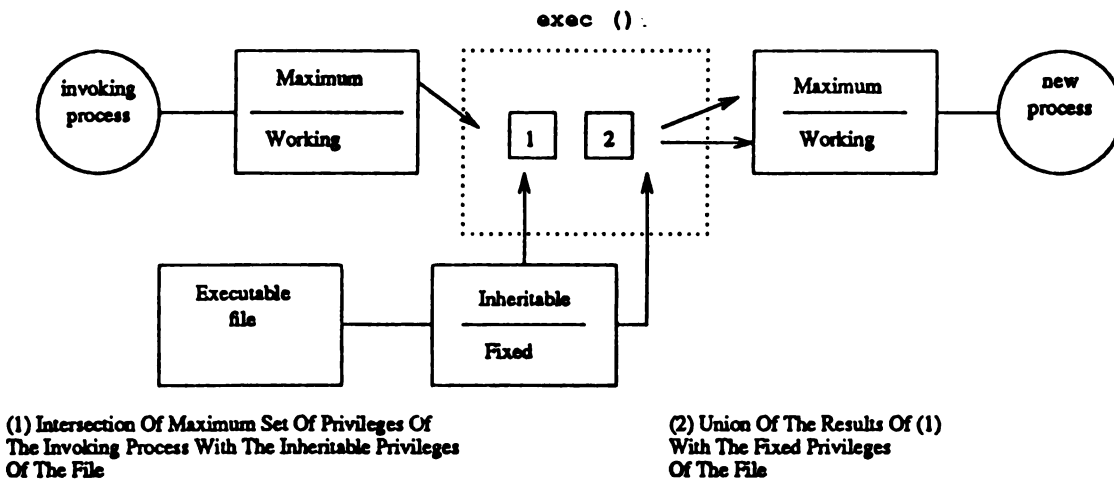
A frequent form of system security subversion is accomplished by the acquisition of "super user" or UID 0 privileges. Historically the UNIX system had a single privileged identity, that of "root" assigned the User Id (UID) of 0. Both file access rights and privilege (i.e., the ability to circumvent the system security policy) were based upon the UID. Due to the dual nature of the UID, once the all powerful user identity of "root" was acquired, the attacker was then able to freely circumvent the system security policy, usually without detection. This type of attack exploits several weaknesses with the historical "root"/UID 0 permission/privilege scheme.

The SVR4.1ES least privilege feature provides the ability for administrators to invoke tasks requiring privilege without requiring "root" access. In previous versions of UNIX, any attempt to execute a sensitive system service (e.g., mount a file system) required the use of a "privilege." In System V, there has been traditionally one such privilege, commonly called "root" or "superuser", which is signified by a

UNIX is a registered trademark of UNIX System Laboratories, Inc.

process whose effective user id is 0. In SVR4.1ES, this single superuser privilege is subdivided into a finer grain set of privileges designed to ensure that sensitive system services execute with the minimum amount of privilege required to execute the task.

In SVR4.1ES, a process has a maximum and working set of privileges associated with it. The maximum set represents the most privilege the process could ever attain and the working set represents the minimum set of privileges required to execute the task. A executable file may have associated with it an inheritable or fixed set of privileges. A inheritable privilege is a privilege which is kept (i.e., left "turned on") only if it already existed in the process. A fixed privilege is a privilege which is always given to the process independent of the previous process privileges. When a file is exec'ed these sets are computed as illustrated in the following diagram:



Note: The fixed and inheritable privilege sets are disjoint; a privilege cannot be present in both sets at the same time.

For compatibility with the current UNIX setuid mechanism, SVR4.1ES supports the concept of fixed file privileges. When a file is executed that has fixed privilege(s), those privilege(s) are added (unioned) with the maximum privilege set of the invoking process forming the maximum and working privilege sets for the resulting process. Note that the fixed privileges are not added to the maximum or working privilege sets of the invoking process.

For example if a site determined that all users should be able to execute the `ps` command and not be subject to mandatory or discretionary access control checks, the administrator would use the `filepriv` command to set the `p_DACread` and `p_MACread` privileges as fixed privileges. Any user invoking `ps` would then acquire the `p_DACread` and `p_MACread` privileges for the duration of the execution of the `ps` command.

For an additional degree of protection, system applications are written such that all privileges in the working set are turned off prior to `exec`. Thus the exec'ed process must explicitly set the privileges which it requires to properly execute. Since all privileges in the working set are dropped prior to `exec`, even if a rogue version of a command were executed it would have inherited no privileges, thus no damage would have occurred. Note that only the active privileges (i.e., the working set) were dropped. This allows a properly exec'ed application to turn on the correct set of privileges upon execution (since the privileges still exist in the maximum set).

2.0.1 Trusted Facility Administration (TFM)

The trusted facility administration (*tfadmin*) facility redefines the way in which the role/privilege assignment mechanism works. In current UNIX systems, an administrator will `login` (or `su`) to an administrative identity. Upon assumption of the identity, all file access rights (and privileges in the case of "root"/UID 0) associated with the identity are assumed by the administrator; all subsequent processes assume these privileges. With this in mind, there are several scenarios by which the vulnerabilities of the

system may be exploited. For example logged in as "root" the administrator invokes:

```
$ date 010191 (set system date & time)
$ mail
```

Since a full pathname was not specified the administrator is relying on the PATH variable being properly set such that the correct commands are executed. Thus the administrator is very vulnerable to attack via trojan horse programs. In this example if the administrator's PATH is not properly set (likely if the administrator assumed the identity via *su*), rogue versions of *mail* or *date* could be executed resulting in the administrator giving "root" privileges away unknowingly. Since all of the attributes associated with the "root" identity are passed to child processes via *exec*, all processes invoked by the administrator execute with privilege, regardless of need. This in turn often results in the execution of code which is not expecting to run with "root" privilege and was not designed with trust in mind. This is especially dangerous with commands that in turn execute other commands or that feature escapes to the shell. For example, an administrator escapes to the shell from *mail* and executes *cat*. Since *mail* was running as "root", the *cat* command was also executed as "root". If a rogue version of *cat* was executed, "root" privilege has inadvertently been given away.

With *tfadmin* there are no privileges inherent with a given user identity, rather privileges are associated with a defined role and are only acquired through execution of *tfadmin*. The *tfadmin* command has associated with it an administrator controlled data base. The data base contains entries in the following format:

```
role:alias:command:privilege(s)
```

-for example-

```
secadmin:date:/bin/date:p_sysops
```

Considering the example above:

```
$ tfadmin date 010191
$ mail
```

Upon execution, the *tfadmin* command searches its database for an entry for *date* for the "role" invoking *tfadmin*. If a match is found, the command is executed (via its fully qualified pathname) only with the explicit privileges needed to perform the requested operation. In this case, only the *sysops* privilege is needed to set the date, thus this is the only privilege passed to the process executing *date*. The next command *mail* requires no privilege to run, therefore execution via *tfadmin* is unnecessary. Since *tfadmin* will only associate privilege with a defined entry, if the administrator invoked:

```
tfadmin mail
```

the command would fail since no database entry would be defined for *mail* (since *mail* does not require privileged execution).

3. Mandatory Access Control

In order to meet customer needs for high data integrity, Mandatory Access Control (MAC) labels have been added to SVR4.1ES. With the addition of Mandatory Access Control, all processes, files, and IPC objects must have a security label. While the DAC mechanism allows permissions to be set at the discretion of the owner of an object, the MAC mechanism is set by the system administrator and enforced by the system. The mandatory access control policy follows a modified Bell-LaPadula model [2] that can be summarized as "read equal or down" and "write equal." For instance, a process at level "top-secret" can read a file at level "secret," and a process at level "secret" would only be able to write to a file at level "secret."

Administrators are responsible for determining and setting up the discrete set of labels at which a user can log in. An administrator also sets a login level range on a terminal line, such that when a user attempts to login, the label specified by the user must dominate the login-low label on the terminal line and in turn be dominated by the login-high label on the terminal line.

By default, SVR4.1ES supports 256 classifications and 1024 categories though the system can be configured to support values up to 65535 and 2097152. For reasons of disk space and performance, SVR4.1ES implements MAC labels with an "indirection" scheme. Each named classification/category tuple (i.e., fully qualified label) is associated with a unique level identifier also known as a LID. The LID serves as a system "pointer" to the fully qualified label name and is the value which is stored in the inode. For reasons of user convenience, each fully qualified label may be assigned an "alias" name. The "alias" name is a short hand representation of the fully qualified label. For example, the "alias" for the label:

TopSecret:projectA,projectB

may be: TS

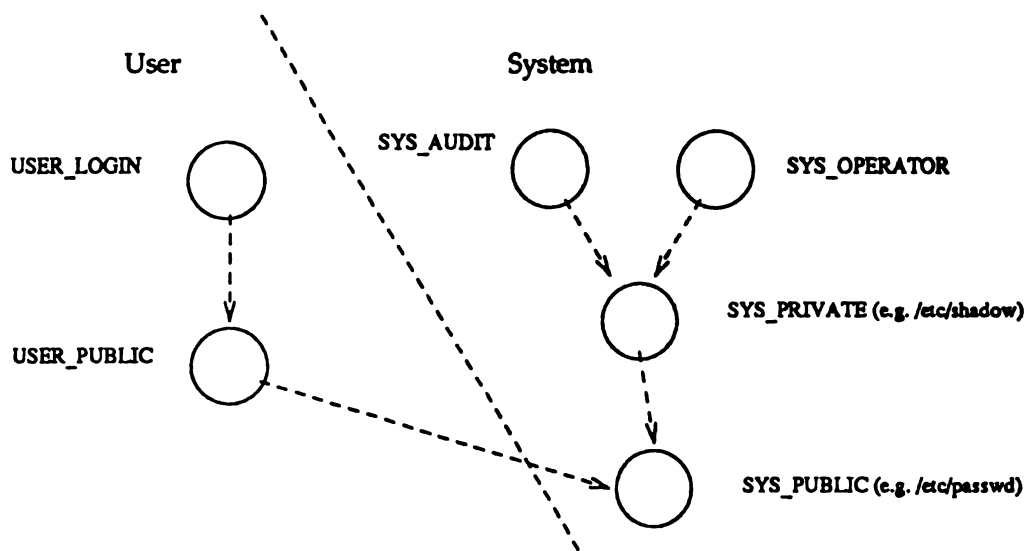
The kernel uses the LID as the primary method of label reference. When the kernel is requested to check access, the LIDs involved in the access determination are compared. If write access is requested, the LIDs themselves are simply compared (since the system enforces a policy of write equal and the LIDs are guaranteed to be unique). For example, if write access to a file with a lid of 10045 is requested by a process with a LID of 10045, access is granted since the LIDs are equal. However if write access is requested to the same file by a process with a LID of 10046 access is denied since the LIDs are not equal. Since the system supports a policy of "read down" the access check required for a read operation requires an additional step. Since no hierarchy can be determined by the comparison of two LIDs (i.e., LID 10046 is not guaranteed to dominate LID 10045), the binary representation of the fully qualified labels of the two LIDs needs to be compared. For reasons of system performance, the binary representation of the labels are kept in a cache, the size of which is a system tunable that may be increased or decreased as required. For example if a read operation was requested to a file with a LID of 10045 by a process with a LID of 10046, the system would do the following:

- Check to see if the binary representation of the LIDs to be compared is already in the cache.
- If the binary representation of both LIDs are not in the cache, the system reads the LID database and brings the binary representation of the LID(s) into the cache.
- The binary representation of the LIDs are compared to determine if a dominance relationship exists (i.e., read access). If so, access is granted; if not access is denied.

4. MAC Access Isolation

An additional form of data integrity, access isolation, can be achieved by judicious use of mandatory access control levels. By setting up a label hierarchy such that user defined labels are disjoint (i.e., do not dominate) from system defined labels, the system is partitioned such that users are prohibited via MAC from reading, modifying, or executing sensitive system files, and administrators are protected from inadvertently executing untrusted code. The following picture illustrates how such a lattice may be defined:

Access Isolation Mechanism



In the lattice depicted above, the levels `USER_PUBLIC` and `USER_LOGIN` are defined for non-administrative use. The level `USER_PUBLIC` is defined for non-administrative user files and commands (eg., *emacs*, databases, etc). The level `USER_LOGIN` is defined for non-administrative system access; by default all non-administrative users access the system at this level. The levels `SYS_PUBLIC`, `SYS_PRIVATE`, and `SYS_AUDIT` are defined for administrative and system use. The level `SYS_PUBLIC` is defined for files/commands which are accessible to both administrators and users (eg., mail, mount, date). The level `SYS_PRIVATE` is defined for administrative system access and is not accessible by non-administrative users. The level `SYS_AUDIT` is reserved for storage of the system audit trail.

Considering the lattice defined above, the commands *date* and *mail* would be labeled at `SYS_PUBLIC`. Since both the user and system portions have read access to data labeled at `SYS_PUBLIC`, both administrators and users have execute permission for these commands. Since the user does not have write permission at the `SYS_PUBLIC` level (MAC restricts write access), a user cannot plant a trojan horse at this level. Note that since the level `SYS_PRIVATE` dominates `SYS_PUBLIC`, the administrator does not require either mandatory or discretionary override privilege to access these files. Thus the administrator executing these commands does not have mandatory access control override permissions and therefore may only execute commands and read files at levels which are dominated by `SYS_PRIVATE`. Since the administrator at `SYS_PRIVATE` does not dominate either `USER_PUBLIC` or `USER_LOGIN` and does not acquire the privilege required to circumvent mandatory access control, the administrator is protected from invoking trojan horse programs planted at this level by users.

4.1 Discretionary Access Control

SVR4.1ES provides two complimentary DAC mechanisms: UNIX file permission modes and TRUSIX conformant access control lists (ACLs). The UNIX file permission modes are retained from previous releases of UNIX System V for compatibility. Users already familiar with UNIX file permissions will find that this mechanism still works as expected.

The SVR4.1ES ACLs are designed to satisfy the B3 level Orange Book requirements while still retaining compatibility with the UNIX file mode scheme. The SVR4.1ES ACL mechanism allows for finer control than existing file permission bits by providing the ability for the owner of an object to grant or deny access by other users to the granularity of a single user.

For convenience, SVR4.1ES ACLs also allow specification of access rights to members of groups as defined to the system in the administrative file */etc/group*. ACLs can also be arbitrarily large; that is, the number of ACL entries is not limited by the system. The system administrator can set the maximum

number of entries per ACL by setting a tunable parameter. (Naturally, as ACLs get larger, processing gets slower, which induces a practical limit on the number of ACL entries.)

In SVR4.1ES, an ACL is associated with every file system object and IPC object. ACLs for file system objects are stored in the associated inode, the first 7 entries are stored in the inode, additional entries are stored in indirectly referenced disk blocks. ACLs for IPC objects are stored in an internal structure associated with the instantiation of the IPC object.

An ACL contains all the DAC access information for the object with which it is associated. For the sake of compatibility, file permissions are displayed as usual in the expected situations, and operations on files behave as they would be expected to on any UNIX System V-based operating system. However, in SVR4.1ES, file permission bits are actually translated into and stored as ACL entries. The ACL entries which are derived from the file owner, file owner group and other permission bits are called base entries. Permission can be granted or denied beyond the base entries by inclusion of additional ACL entries. A simple SVR4.1ES ACL would appear as follows (note the numbers in parenthesis are used to indicate the association between the permission bits, owner and group and the ACL. They do not appear in SVR4.1ES ACLs):

```

(4)(3)(6)      (2) (3)      (1)
rwxr-xr-x+  1  fred demo  73 Jan 6 20:27  run.sh

#file: run.sh      (1)
#owner: fred      (2)
#group: demo      (3)
user:rwx (4) ...
user:larry:-x
group:r-x (5) ...
group:sys:-x
class:r-x
other:r-x (6)

```

or 'ing these entries provides class entry

Notes:

+ sign indicates file has an associated ACL

the class entry is always equal to the group permission bits. Thus stat'ing the file provides the maximum permission granted by the ACL

An ACL consists of the following types of entries, which must be in the following order:

- **user entry** - This entry is derived from the file owner permission bits; it contains a user ID and the permissions associated with it. There is always one entry of this type, which represents the object owner and is denoted by a null (unspecified) user ID. There may be additional unique user entries.
- **group entry** - This entry is derived from the file group permission bits; it contains a group ID and the permissions associated with it. There is always one entry of this type, which represents the object owning group and is denoted by a null (unspecified) group ID. There may be additional unique group entries.
- **other entry** - This type of entry contains the permissions granted to a subject if none of the above entries have been matched. There is exactly one of these entries in an ACL.
- **class entry** - This type of entry contains the maximum permissions granted to the file group class. There is exactly one of these entries in the ACL. The class entry indicates the maximum permission allowed by the ACL. Additionally, this entry acts as a mask and provides compatibility for existing applications which obtain file access permission via *stat* and attempt to change file status via *chmod*, for example:

Modification of mode bits & ACL using chmod

Before chmod 000

`rwxf-rf-x-`

```
#file: run.sh
#owner: fred
#group: demo
user::rwx
user:larry:--x
group::r-x
group:sys:---
class:r-x
other:r-x
```

After chmod 000

```
#file: run.sh
#owner: fred
#group: demo
user:---
user:larry:--x
group::r-x
group:sys:---
class:---
other:---
```

After chmod 755 (re-set mode bits)

`rwxf-rf-x-`

```
#file: run.sh
#owner: fred
#group: demo
user::rwx
user:larry:--x
group::r-x
group:sys:---
class:r-x
other:r-x
```

Referring to the example above; notice that the ACL entries for file owner, other and file group class are changed to reflect the intended setting of the permission bits (via *chmod()*). No additional ACL entries are modified. The intended effect of the *chmod 000* is accomplished by using the file group class entry as a mask. Note that the file owner group entry was not modified by the *chmod*. This is due to the fact that the SVR4.1ES implementation treats the file owner group as an additional ACL entry.

- **default entry** - This type of entry may only exist on a directory. These entries are similar to the entries described above, except that they are never used in an access check, but are used to indicate the user, group, and other ACL entries that should be added to a file created within the directory.

4.2 Trusted Path

The SVR4.1ES trusted path feature is a streams module which ensures that the user's password is being requested by login and not by a malicious program that masquerades as a system program to gain sensitive information. The SVR4.1ES trusted path mechanism is only invoked at *login* time and is not directly invocable by the user.

The user invokes the trusted path and subsequently gains access to the system via a terminal using the Secure Attention Key (SAK). By default the SAK is a line drop though it can be configured by the administrator to be a character or asynchronous line condition, such as a break.

The SVR4.1ES trusted path feature works as follows:

1. A user requesting access to the system enters the SAK.
2. The system identifies the SAK before any line discipline is applied.
3. On detecting the SAK, the TCB terminates any current login session, permanently puts open connections in a state such that they can no longer be used for terminal I/O, and eventually reinitiates the login sequence.
4. If login is not completed within the login timeout period, the login program will enter a mode where login interaction cannot proceed until the SAK is entered again.

4.3 Audit

Hand in hand with the ability to penetrate system security is the ability to do so without detection. On most UNIX systems the only record of process execution is the information saved by the UNIX systems process accounting facility. While this data provides some insight as to what may have occurred on the system, it can be spoofed and does not provide sufficient granularity of data to fully determine the actions of an intruder. Additionally, existing UNIX process accounting provides no granularity, it is an all-or-nothing feature; either accounting is enabled for all (known) events, for all users or it is completely disabled. Since the recording of accounting data is done on an all-event, all-user basis, a good deal of system resources are expended; for this reasons, it is frequently not used. These shortcomings have been corrected in SVR4.1ES with the addition of system auditing. Like accounting, auditing records events which occur on the system. However, in addition to simply recording the occurrence of events, auditing also records the parameters

associated with the event and the outcome of the event. Granularity is provided at both the event and user level, that is, the administrator can select specific events which will be audited and can specify the users for whom those events are audited. Since the system's audit daemon runs with a mandatory access control level which is disjoint from all defined user levels, the presence of the audit daemon (i.e., the ability to detect auditing) is undetectable by unprivileged users. SVR4.1ES provides an audit mechanism capable of recording and reporting on all security-related events that occur on the system.

All security-related events that occur on the system can be audited, including those events identified as being associated with covert channels. SVR4.1ES associates most audit events with a system call. For example the `mk_dir` and `rm_dir` events map the `mkdir` and `rmdir` system calls. Since system administrators tend to think in terms of system events, SVR4.1ES provides the concept of an event class. The class mechanism allows for a logical grouping of event types. For example, the `mk_dir` and `rm_dir` events fall into the `dir_make` class. Since auditing tends to generate large amounts of data and since an administrator may wish to select most but not all of the event types within a class, SVR4.1ES permits selection by both event type and class. Additionally the selections can be intermixed (i.e., a class may be selected and one or more types within the class may be turned off).

Since a certain sub-set of applications may wish to add records to the audit trail, the SVR4.1ES audit feature provides the ability for applications to add their own free-format records to the audit trail. Multiple site or application records may be defined. These added records can be selected and later reported using the standard SVR4.1ES selection and reporting tools.

Events which are deemed critical to the integrity of the system (i.e., events critical to the integrity of the audit trail) are always audited whenever auditing is enabled regardless of the system wide and per-user event masks. These events are called *fixed* events. Other events are auditable at the discretion of the system administrator; these are called *selectable* events.

As stated above, events may be set on either a system wide or per-user basis. System wide events are selected by the administrator with the `auditset` command. `auditset` may also be used after auditing is enabled to specify additional events to be audited or to de-select events that no longer require auditing.

Per-user audit masks may be designated for each user by using the `useradd` command. These masks are permanent - whenever auditing is enabled and the user is logged on, events specified in these masks will be audited. The set of *fixed* events along with the system wide and per-user audit masks are or'ed together to form the user's process audit mask.

Each auditable event, when audited, generates an associated audit record; collected for each event audited are a time stamp, the user identity, object name, level of the process (subject) causing the event, privileges used, an identification of the type of event, and an indication of the success or failure of the event. Other information specific to the event type is also collected. The `auditprt` command is used to select, format and print data from the log file.

5. Summary

This paper has described several security features that provide a high degree of protection against unauthorized access, viruses, and trojan horses. In most cases, system security is compromised by exploitation of an administrative oversight such as incorrect setting of file mode bits. Meticulous use of the security features already present in the UNIX system can eliminate or greatly reduce most breaches of system security. However, since most, if not all, of the current UNIX system security features rely solely on administrator discretion, no matter how carefully a system is administered, mistakes can and do occur. When mistakes do occur, the system is left vulnerable in some area. System enforced features such as mandatory access control and least privilege eliminate or greatly reduce the amount of compromise that can occur if an administrative flaw is detected and exploited. Thus the burden of system protection is no longer solely dependent on the administrator.

6. REFERENCES

- [1] Department of Defense. *Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD, December, 1985.

- [2] Bell, D. E. and LaPadula, L. J. *Secure Computer System: Unified Exposition and Multics Interpretation*, MITRE Corporation, MTR-2997, March 1976.

KNOWLEDGE-BASED COMPUTER SECURITY ADVISOR*

W. J. Huntzman and M. B. Squire
Safeguards Systems Group, MS-E541
Los Alamos National Laboratory
P.O. Box 1663
Los Alamos, NM 87545

ABSTRACT

The rapid expansion of computer security information and technology has included little support to help the security officer identify the safeguards needed to comply with a policy and to secure a computing system. Los Alamos is developing a knowledge-based computer security system to provide expert knowledge to the security officer. This system includes a model for expressing the complex requirements in computer security policy statements. The model is part of an expert system that allows a security officer to describe a computer system and then determine compliance with the policy. The model contains a generic representation that contains network relationships among the policy concepts to support inferencing based on information represented in the generic policy description.

I. INTRODUCTION

The field of computer security is continuing to expand the information security technology available to address security concerns in computing systems. The advances are often directed towards technological solutions of a multidimensional problem, but the nontechnical areas have received little, if any, serious effort towards improving the entire security environment surrounding a computing system. The use of trusted computing systems alleviates the problem somewhat by implementing the nondisclosure policy in a standard manner [1]. However, this approach does not address other equally important security issues such as other policy components (e.g., personnel security and physical security) or the interaction between the Trusted Computing Base (TCB) and the security features in the local environment (e.g., administrative procedures).

This paper describes an effort initiated at Los Alamos to create a knowledge-based system to act as an "expert" Advisor to a security officer. The Advisor will consider the total environment, including policy requirements, when identifying the security needs for a computing system. The Advisor provides an automated capability to support the system certification process. System certification, as described in References 2 and 3, requires an analysis of the system security features, threats against the system, and the system operating environment according to an information security policy. The Advisor system is designed to be used during the development of a secure system and when reviewing or certifying the security of an existing system for compliance with a policy.

Most policy statements are complex and difficult to interpret for a local computing system environment. This difficulty generally arises from the desire for the policy to allow the maximum flexibility for changes in the hardware or software configurations of a computing system. Experts from the policy-making organizations will also sometimes give conflicting advice regarding policy implementation for a particular system. The lack of clear guidance on applying the policy and the absence of a consistent approach to implementation suggest that a uniform methodology is needed to aid the security officer in interpreting and applying security policies.

*Work supported by the US Department of Energy, Office of Safeguards and Security.

The methodology being developed as part of the Advisor provides a consistent decomposition and interpretation of policy statements into a knowledge base that can be used to guide the selection of safeguards for a specific computing system. The Advisor architecture is designed to

- support the semantic or conceptual representation of a complex system;
- if appropriate, collect and organize information about local or site-specific policies;
- support automated reasoning about the represented system;
- manage the use of uncertain or incomplete information in the knowledge networks;
- support "what-if" experimentation to adjust the local environment implementation description; and
- provide, on request, justification or explanation of each decision throughout the process.

The architecture supports multiple representations of policies, regulations, local or site-specific implementations of the policies, and the interdependencies between the various concepts and implementations. The architecture is designed to allow the development of user oriented interfaces that display information in a manner consistent with the user's vocabulary and operating environment. The Computer Security version of the Advisor will implement the Department of Energy (DOE) Classified Computer Security Program defined in DOE Order 5637.1 [3].

II. POLICY REPRESENTATION ISSUES

A policy statement is intended to guide personnel in constructing a local environment that has some general property, such as a safe or secure environment. Policy statements are usually written by, or with the help of, experts in the field. Policy implementors, however, often lack the complete understanding to interpret the exact meaning of the policy. Some statements may be unclear, such as "Procedures for identifying and authenticating users must be addressed." This may be either an oversight by the policy writer or a deliberate ambiguity to allow flexibility of interpretation. If it is for flexibility, the implementor must decide how to interpret the intent and then implement a solution. Typically this solution must then be approved by an approval or accrediting authority who may have a different interpretation of the policy. Some organizations also allow implementors to create unique interpretations and implementations of the policy requirements, subject to approval by the accrediting authority. Regardless of the allowed flexibility, there are some characteristics that seem to be shared by all policy statements.

A. Property/Requirement Coupling

When a policy is broken down into specific requirements, the requirements can be expressed as a coupling of a specific problem and the expected solution. These requirement/solution pairs can be viewed as a list of IF/THEN statements. For example, a policy statement could be

IF a computer processes classified information
THEN it must have identification and authentication procedures.

We call the IF clause a property, and the THEN clause a requirement. A property is the activity or condition that must be present or practiced to meet the requirement. We refer to this coupling of property and requirement as a property/requirement (p/r) couple. Most instances of a p/r couple can be further decomposed. The property can be expressed as a nested set of conjunctions and disjunctions of objects, relations, and attributes. Similarly, the requirement can also be expressed as a nested set of conjunctions and disjunctions.

B. Existence/Event Coupling

Policy statements also have a distinction between passive p/r couples and active p/r couples. A passive policy statement does not explicitly or implicitly require invoking a specific requirement based on some action by a subject, such as a user or process. For example, the following could be viewed as a passive p/r couple because there is no explicit requirement to invoke the requirement.

IF a computer processes classified information
THEN it must have identification and authentication procedures.

However, in most policies there is either an explicit or implied requirement to respond to action by a subject. For example, the implied active part of the policy statement in the above example, could be

IF a subject attempts to logon to a computer
THEN identification and authentication procedures must be invoked.

We refer to the passive part of this policy element as the existence and to the active part as the event. These pairs of existence and event p/r couples are referred to as existence/event (e/e) couples. It is possible that either the existence or the event could be empty. For example, there is no related event p/r couple in the following:

IF a computer processes classified information
THEN it must be in a protected area.

Property/requirement couples based on events are slightly more complicated and can be modelled as state changes in the policy knowledge network. Many policies require that certain procedures be done periodically. These can be modelled as an event, namely, the passage of time. For example, it may be required that a computer system is reviewed annually. This can be modelled as the event of a year passing or a time-related transition.

Problems based on existence will be referred to as "vulnerabilities," and solutions based on existence as "safeguards." We will refer to problems based on events as "attacks" and to solutions based on events as "responses." An interesting property of most policy statements is that whenever an existence problem occurs, then the expected solution is also based on existence. Similarly, problems based on events have solutions based on events.

C. Hierarchical Order of Policy Statements

Policy statements are often hierarchically arranged. First, the e/e couples can be arranged by some categorical hierarchy. For example, all e/e couples relating to "Personnel Security" can be grouped into one category, which in itself can be a category in "Computer System Security." Also, each property or requirement can be composed of subproperties/subrequirements. The subproperties/subrequirements can also be further refined with the subordinate items categorizing and defining their parents. Figure 1 depicts the general representation of a policy element used by the Advisor model.

D. User Defined Solutions

Some policies allow users to develop their own solutions to policy requirements. This approach effectively allows the user to modify the hierarchy under the requirement part of one or more p/r couples. Often the security officer is allowed to create a specific solution to the problem as long as it satisfies the general intent of the policy. The Advisor model allows a controlled capability for security officers to substitute approved alternative solutions.

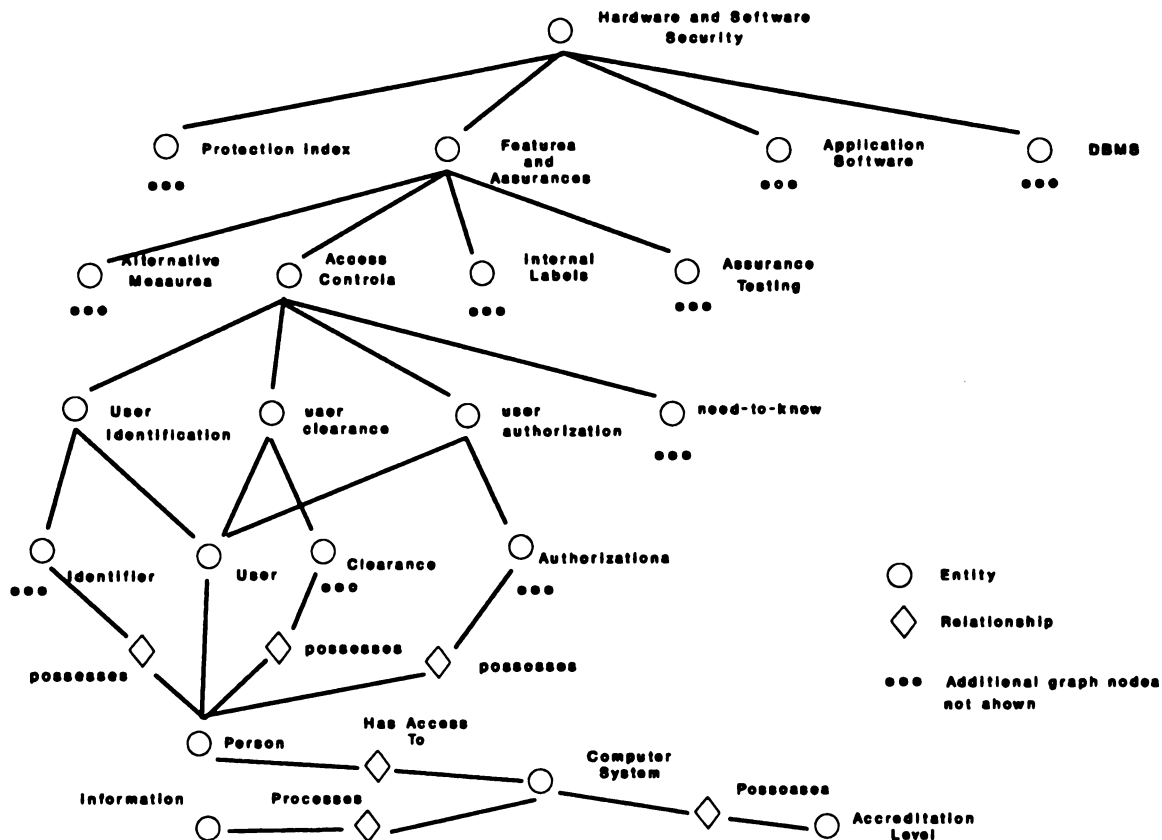


Figure 1. Conceptual graph of policy fragment.

III. POLICY REPRESENTATION

A. Policy Representation Requirements

An acceptable representation of a policy statement must be able to represent the domain addressed by the policy, differentiate between the policy concepts and instances, and support a categorical organization of the policy. First, we must be able to accurately represent the policy domain. In addition to properties and requirements, we must be able to represent relationships between properties and requirements, interactions between events and p/r couples, and time. For example, suppose we wished to represent a personnel security policy for a secure computer system. We must be able to represent such concepts as computers, classification levels, and users. We must also be able to represent relationships between these concepts, such as the relationship between a computer and its users. We also must be able to differentiate between instances and concepts. For example, if the policy states that all classified computers must be in protected areas, we want to be able to differentiate between the concept of a classified computer and a particular instance of a classified computer. The representation approach must also support a categorical hierarchy for the e/e couples. The Advisor model also allows for controlled modifications to the hierarchy when the policy supports implementor flexibility. The user modifications are restricted to properties already defined in the policy domain. For example, if the policy allows substitution of physical protection for user identification and authentication, then the user must be restricted to selection of known and approved physical

protection properties when making the modification. Events must also be represented. For example, we must be able to represent the event of a user login to the computer. We must also be able to model procedures, such as the generation and distribution of authenticators.

B. Advisor Model Representation

The Advisor model uses conceptual graphs [4] to represent policy information. The policy representation conceptual graph contains three types of nodes: category, policy, and network. Category nodes are used to organize the high-level segments of the policy. Policy nodes represent e/e couples. A policy node may be connected to up to four network nodes. Network nodes represent the policy node's existence p/r couple and its event p/r couple. Network nodes are the clauses of the IF/THEN structures. The generic representation of policy nodes and network nodes is given in Figure 2.

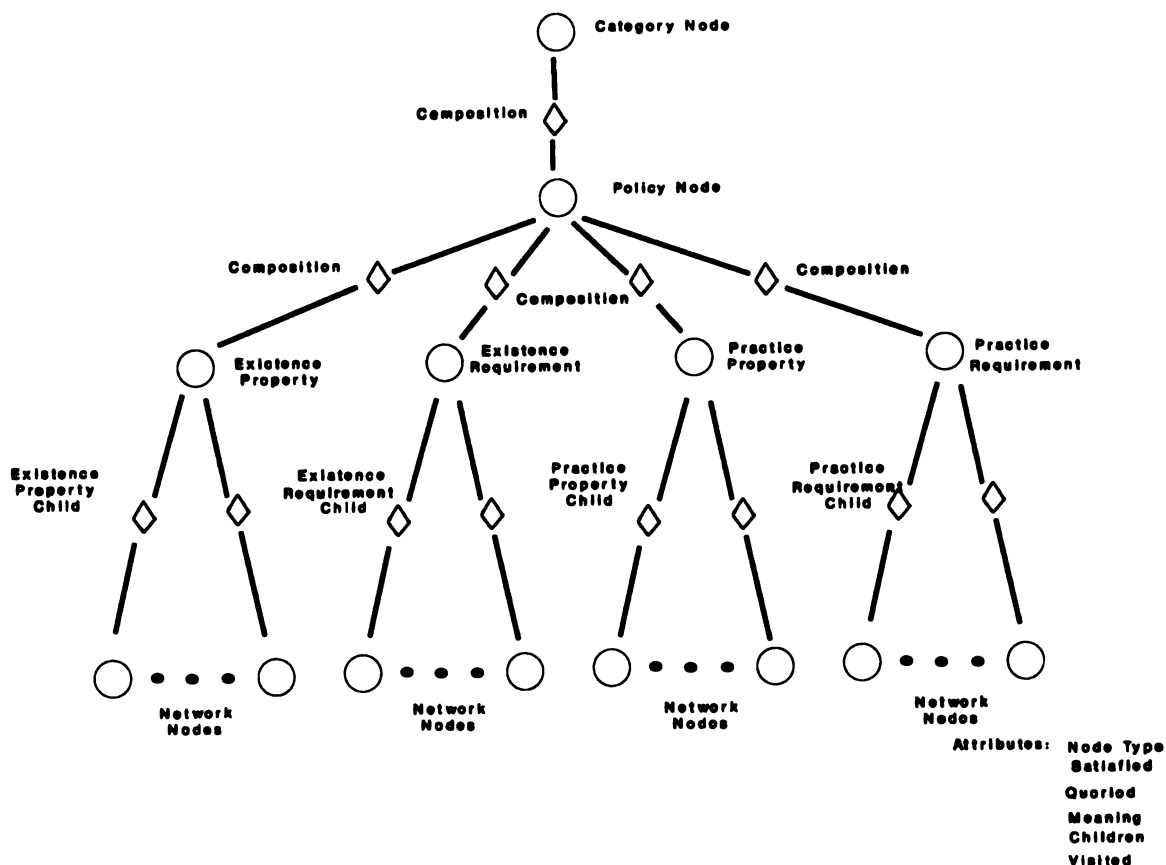


Figure 2. Generic Advisor model.

C. Advisor Architecture

The Advisor architecture, shown in Figure 3, contains two different networks [5]. The Computing Environment network is composed of network nodes that are used to guide and collect user-supplied descriptions of the local computing environment (instantiations). The Policy network contains category, policy, and network nodes that represent the policy. The Analysis component is software that evaluates the instantiations against the policy and reports the results. The Developer Interface contains facilities for creating and maintaining the Policy and Computing Environment Networks. The User Interfaces provide capabilities to allow the user to enter, view, and manipulate information in a user-friendly manner.

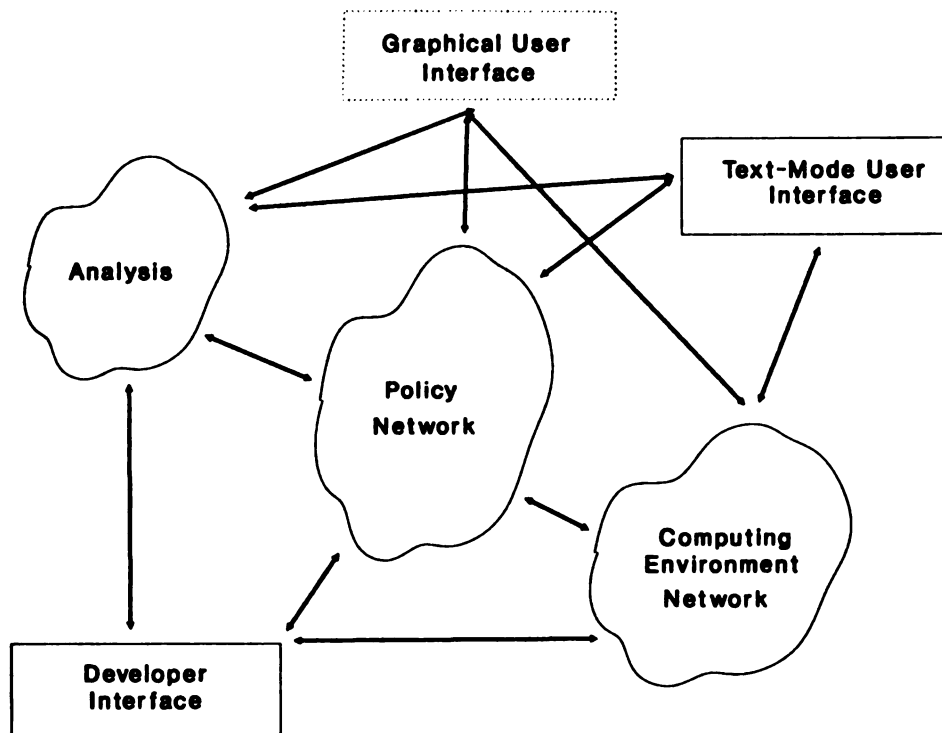


Figure. 3. Advisor architecture.

The Policy network also supports a global analysis of the policy statement by supporting the representation of multiple policies and networks of attacks/vulnerabilities and responses/safeguards that represent everything the policy must address. If a node in the Policy network cannot be associated with an attack, then either the attack/vulnerability network is incomplete and must be expanded, or that property or requirement is superfluous and should not be in the policy statement. If there is an attack/vulnerability that does not match any property or requirement, then this attack/vulnerability is not addressed by the policy, indicating an incompleteness of the policy statement. A similar analysis can be performed with responses/safeguards.

D. Advisor Knowledge Network

The interior nodes of the knowledge network may be either AND, OR, or XOR (exclusive or) nodes. Each interior node will have a node-type attribute (either AND, OR, or XOR), a satisfied attribute (YES/NO), and a meaning attribute. AND nodes require that all of their children be addressed during instantiation and analysis. OR nodes represent redundant information and allow any of their children to be addressed during instantiation and analysis. XOR nodes are used when policy elements conflict with each other and only one child will be considered during instantiation and analysis. An example of conflicting policy elements might be an audit trail that recorded every keystroke entered by a user and normal password security. The complete audit trail would contain user- and file-access passwords, while password security would not allow the passwords to be exposed in a clear form.

The satisfied attribute specifies whether or not the user has supplied the information for an instantiation of this node and whether or not the node is satisfied by the instantiation. An AND node is considered satisfied only if all of its children are satisfied. An OR node is considered satisfied if any of its children are satisfied. An XOR is considered satisfied if one of its children is satisfied and the other is not. If both children of an XOR node are satisfied a conflict is reported to the user.

The meaning attribute is used with network nodes in the Policy network to provide a linkage between the Policy and Computing Environment networks. The meaning attribute contains the name of a node in the Computing Environment network that is expected to contain a user-supplied instantiation. During the analysis phase, the Policy network is searched for the meaning attribute strings that are used to extract the instantiations for further analysis in the Policy network.

E. System Evaluation

The leaf nodes of the Computing Environment network contain the user provided instantiations and allow the Advisor to query the Policy network to determine if a p/r couple is satisfied. This information on the satisfied attribute of the child is then used by the parent concept to determine whether or not it is satisfied. A leaf node in the Computing Environment network is considered satisfied if an instantiation for the concept has been provided by the user. The information on the satisfied attribute of the leaf node is propagated to the top of the Computing Environment network where it is used to determine if the parent p/r couple is satisfied. The satisfaction of a p/r couple is then used to determine the satisfaction of individual policy couples in the Policy network.

IV. USER INTERFACE

The Computer Security Advisor implementation is designed to support the needs and activities of all of the positions identified in the Department of Energy (DOE) Classified Computer Security Program. These positions include Computer Security Program Manager (CSPM), Computer Security Operations Manager (CSOM), Computer Security Site Manager (CSSM), and Computer System Security Officer (CSSO). The CSPM is responsible for establishing the classified computer security policy for the DOE. The CSPM is also responsible for developing and maintaining a definition of the threats to DOE and contractor facilities. The CSPM may, under certain circumstances, be an accrediting authority for complex computer systems or systems that cross CSOM responsibility boundaries. The CSOM position is typically assigned to an individual in the DOE Operations Office and is responsible for oversight and guidance of the computer security program implemented by the Operations Office and any DOE contractors reporting to the Operations Office. The CSOM is responsible for review and approval of ADP Security Plans for all computer systems processing classified information in the DOE office or contractor facilities. The CSOM is typically the accrediting authority for these computer facilities. The CSSM is the individual responsible for the classified computer security program at the site or facility. The CSSM is the principal contact point and coordinator for all communications and interactions between the site and the CSOM. The CSSM is responsible for review of all ADP Security Plans and the certification of the computer systems during the accreditation process. The CSSM is also responsible for defining and implementing site-wide computer security procedures. The CSSO is the security officer responsible for defining and implementing the computer security procedures and mechanisms for a computer system that processes classified information. The CSSO is also responsible for generating and maintaining the ADP Security Plan and the ADP Security Test Plan.

The user interface of the Computer Security Advisor is based on the windowing system supported by Sun Microsystem's Open Look software. The user is presented with a series of successively detailed windows that are oriented to the particular function requested by the user.

The initial window, Figure 4, allows the user to select the desired interaction level (security officer, reviewer, or developer).

The security officer window, Figure 5, allows the user to select operations to load or save the Policy and Computing Environment networks (FILE button), exit the Advisor (QUIT button), edit or display the Policy and Computing Environment networks (EDIT button), describe a computer system (CREATE SYSTEM button), or evaluate the described system against the policy requirements (ANALYSIS button). The ANALYSIS and CREATE

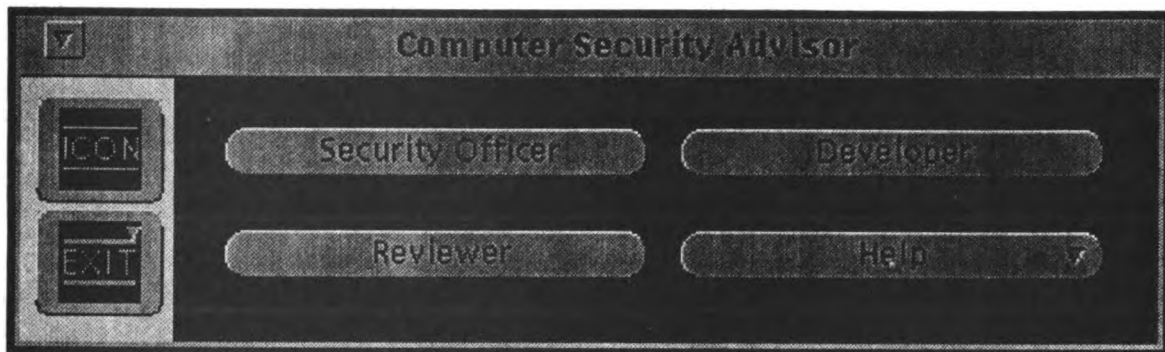


Figure 4. Initial Advisor screen.

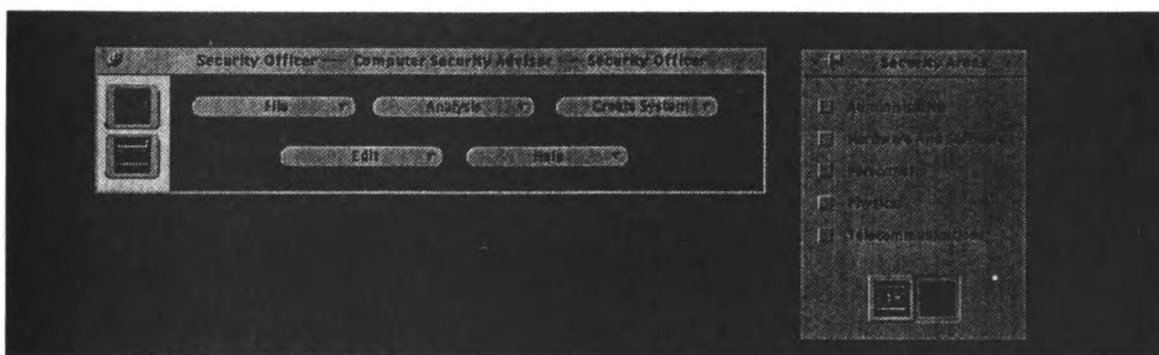


Figure 5. Initial security officer screen.

SYSTEM functions allow the user to analyze or describe the entire system environment or select a specific subset of the environment defined by the policy network. The DOE policy is divided into personnel security, physical security, telecommunications security, administrative security, and hardware/software security sections.

The CREATE SYSTEM functions guide the user through the process of specifying the instantiations of the computer system. The Advisor searches the Computing Environment network for concepts that must be instantiated to satisfy the policy. When a required concept is found, the user is asked to respond if the concept is present or practiced in the local environment. If appropriate, the user is also asked to identify the instance (e.g., name or procedure title). Figure 6 contains an example of the instantiation activity.

The ANALYSIS functions initiate the evaluation of the computer system against the policy requirements. After the evaluation is completed, the results are displayed for the user. Figure 7 contains a sample display showing the results of an analysis. If all p/r couples in the Policy network are satisfied, then only a single line is displayed stating that the top level policy network node was satisfied. If one or more p/r couples are not satisfied, then the unsatisfied p/r couple(s) are displayed with all subordinate p/r couples that contributed to the failure of the top level p/r couple.

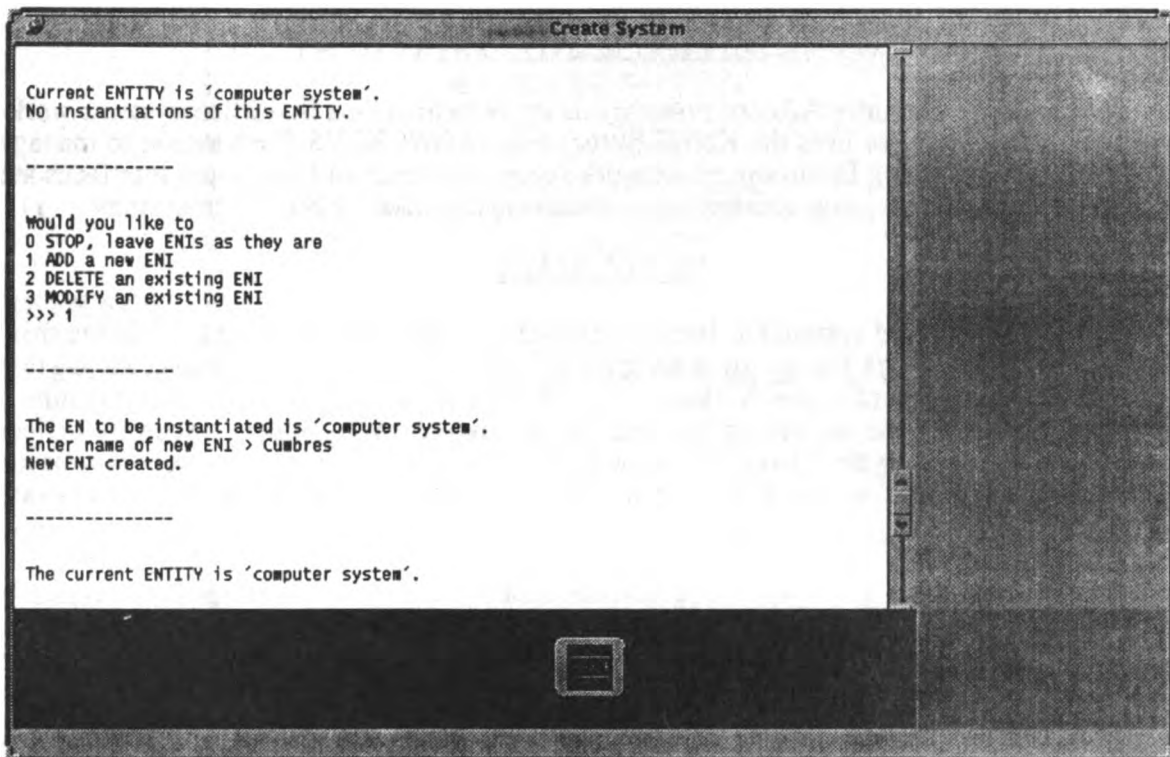


Figure 6. Instantiation window.

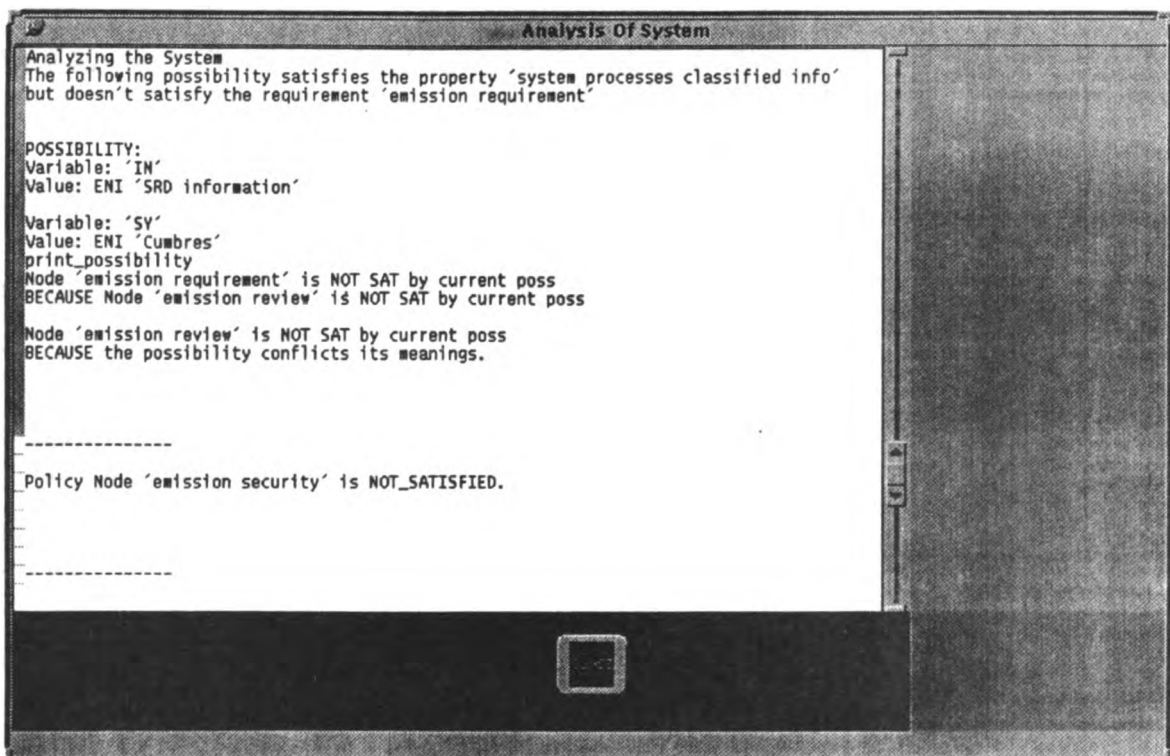


Figure 7. Analysis window.

V. IMPLEMENTATION

The Computer Security Advisor prototype is implemented on a Sun Microsystems workstation in C. The Advisor uses the KNET library from KONEXSYS Corporation to manage the Policy and Computing Environment network space. The user and developer interfaces are implemented in the Open Look windowing environment provided by Sun Microsystems.

VI. SUMMARY

A knowledge-based system has been developed to collect and organize knowledge from computer security experts for use by a security officer. The Advisor includes a model that incorporates all aspects of a policy statement. The Computer Security Advisor contains a generic description of the desired policy and the user interface to support a security officer description of the local system and analysis of policy compliance. The system is policy-based and contains the flexibility needed to support changes in policies and hardware and software technology.

REFERENCES

1. "Department of Defense Trusted Computer System Evaluation Criteria," DOD 5200.28-STD, National Computer Security Center (1985).
2. National Research Council, *Computers At Risk: Safe Computing in the Information Age* (National Academy Press, Washington D.C., 1990).
3. "Classified Computer Security Program," Department of Energy Order 5637.1, Department of Energy (January 1988).
4. J. F. Sowa, *Conceptual Structures—Information Processing in Mind and Machines* (Addison-Wesley, Massachusetts, 1984).
5. N. V. Findler, Ed., *Associative Networks* (Academic Press, New York, 1979).

THE LOGISTICS OF DISTRIBUTING A SMART TOKEN

**Dawn A. Brown
Department of Defense
Ft. George G. Meade, MD 20755-6000
(301) 859-4360**

ABSTRACT

This paper will address the logistics of distributing a smart token on a computer system. A smart token is an identification and authentication device for a host computer system. This paper will address the logistics from four perspectives. The first perspective will discuss why the smart token, WATCHWORD Generator was implemented on DOCKMASTER. A cost analysis, including procurement of the smart token, batteries, man hours, and maintenance is the second perspective. The third perspective discusses how the smart token will enhance the security of the host computer system. How DOCKMASTER will respond when a user is trying to access the system with the WATCHWORD Generator implemented is the fourth perspective. With a successful method of identifying and authenticating users of the computer system, the system is less susceptible to penetration.

INTRODUCTION

DOCKMASTER is the National Security Agency's (NSA) unclassified computer system that directly supports the missions and functions of the National Computer Security Center (NCSC). DOCKMASTER was established as the Information Security Showplace for dissemination and exchange of Information Security data. DOCKMASTER executes the Multics Operating System which was granted a B2 security rating based on the guidelines defined in the Department of Defense Trusted Computer System Evaluation Criteria, also known as the "Orange Book".

With the increasing number of computer penetrations, it is vital that each computer user is correctly identified when accessing a computer system. The process of correctly identifying each computer user is called authentication. The primary authentication device on DOCKMASTER is the WATCHWORD Generator.

The WATCHWORD Generator is a portable, hand held authentication device that is used in conjunction with the user's password during the login process. Each WATCHWORD Generator is assigned a unique Personal Identification Number (PIN) and Secret Key. During the login process, the user must correctly authenticate his/her login process by using the WATCHWORD Generator. The WATCHWORD Generator will generate a different response during each login process based on the "Challenge" generated from DOCKMASTER. If the correct response to the "Challenge" is not entered the user will be denied access to DOCKMASTER.

WATCHWORD GENERATOR IMPLEMENTATION

With every computer system there should be a means of authenticating who is trying to access the system. As with most computer systems, DOCKMASTER uses the userid and password option as a means of authenticating each user. However, should this option be the only means of authenticating users? The answer depends on several questions. For example, what type of data (unclassified, classified, proprietary) is the user trying to access, should the user have access to this data, and

is the data restricted to specific users. If the answer to any of these questions is yes, then the userid and password option should not be the only means of authenticating users.

In 1987, DOCKMASTER Management was faced with the question, how can we enhance the protection of restricted data while also authenticating each user. A decision was made to add an additional layer of security to the login sequence that would identify and authenticate each user requesting access to restricted data. A month long operational test consisting of twenty-one users accessing DOCKMASTER through various methods (Direct Dial, Tymnet, Telnet, etc.) was conducted. Based on the conclusions of the test, the WATCHWORD Generator was chosen as the most effective way to add the additional layer of security to DOCKMASTER.

COST ANALYSIS

There are overhead costs involved in the implementation and use of the WATCHWORD Generator. Some of the overhead costs include:

- a. The WATCHWORD Generator software.
- b. The WATCHWORD Generator devices.
- c. The WATCHWORD Generator batteries.
- d. The WATCHWORD Generator Administrator duties.
- e. Maintenance and recovery of the WATCHWORD Generators.
- f. Replacement WATCHWORD Generators and batteries.

The initial overhead cost of the WATCHWORD Generator includes procuring the software for the WATCHWORD Generators. This software is necessary to communicate with the host computer. Additionally the cost of one device for each user that requires authentication by the system must be incurred. The WATCHWORD Generator costs approximately ninety dollars each. Given a user population of five hundred, the total cost to procure the WATCHWORD Generator is approximately forty-five thousand dollars. This figure may appear to be substantial at the outset, but consideration should be given to the thousands of dollars that will be saved when the WATCHWORD Generator is implemented.

When a computer system is compromised, time and money must be spent on tracing the path of the computer hacker, notifying users of the penetration so that they can change their passwords and ensure that their data was not compromised, and investigating why the penetration occurred. The cost involved in this whole process can be substantial. The time and money that must be invested if the computer system is compromised will not have to be incurred if the WATCHWORD Generator is implemented. The chances of a computer system being compromised with the WATCHWORD Generator implemented is virtually zero. The advantages of implementing the WATCHWORD Generator out way the disadvantages considerably.

The WATCHWORD Generator is battery operated, thus the cost of the batteries is a second overhead cost. Each WATCHWORD Generator requires two calculator or equivalent batteries. The cost per set of batteries for the WATCHWORD Generators is less than one dollar. As with the cost of the WATCHWORD Generator device, the cost of the batteries is minute compared to the advantages and additional security that the WATCHWORD Generator will bring to the computer system.

The third overhead cost includes the actual man hours involved in implementing the WATCHWORD Generator. Every computer system should have one or more individuals that concentrates on the security of the system. This person is usually called the Computer Security Officer (CSO). The CSO may be a prime candidate to implement the WATCHWORD Generator since the WATCHWORD Generator does add an additional layer of security to the computer system. However, the CSO does not have to implement the WATCHWORD Generators,. A WATCHWORD Generator Administrator (WGA) should be appointed.

The WGA responsibilities should include, but are not limited to, installing batteries into the WATCHWORD Generator device, assigning a unique PIN to each device, keying each device with a unique secret key, recording each device in the controllers and database, maintaining an accurate inventory of WATCHWORD Generators and batteries, and ensuring the return of unused WATCHWORD Generators for reissuance.

Each device requires approximately fifteen minutes to implement on the computer system. Based on the number of devices that will be implemented at one time, the number of man hours invested is also minimal. The relatively small number of man hours invested is small price to pay for the numerous advantages that implementing the WATCHWORD Generator will provide.

Ensuring the return of unused WATCHWORD Generators may require the most man hours. For example, if a user changes job positions, relocates, is fired, or if the company moves, it is the responsibility of the WGA to locate the user and ensure the return of the WATCHWORD Generator. A Standard Operating Procedure (SOP) should be established to deal with problems such as the ones listed above. With a well defined SOP the WGA should not have any problems in deciding what the next step should be in ensuring the return of the WATCHWORD Generators.

The life span of the batteries for the WATCHWORD Generators is approximately two years. Therefore, to minimize user inconvenience, a system of exchanging WATCHWORD Generators must be implemented. The WGA must issue each user a new WATCHWORD Generator. Each WATCHWORD Generator must have a new PIN as well as a new secret key. The purpose of issuing a new PIN and secret key is to enhance key management and security of the computer system.

During the exchange phase of the WATCHWORD Generators, each user will have two WATCHWORD Generators for a short period of time, but only one WATCHWORD Generator will be used to authenticate the user. The WGA must explain to the user population the procedures of why, when, and how the replacement WATCHWORD Generator will be used. This process can become extremely confusing if a detailed plan is not implemented. The exchanging of WATCHWORD Generators will enhance the security of the computer system by reducing the chances of a users PIN and or secret key being compromised. The longer a user utilizes the same PIN the greater the possibility that their PIN will be compromised.

Some may argue that it would be easier and less time consuming to issue new batteries to each user. This would not be a feasible method because once the batteries are removed the memory is automatically erased. Once the PIN and secret key is erased, the device will no longer be able to function as a smart token.

The cost involved in the exchange process is also minimal. If an adequate number of WATCHWORD Generators and batteries are procured during the initial phase, the

only cost that should occur is the cost of mailing the replacement WATCHWORD Generators and the man hours to implement the exchange process.

WATCHWORD GENERATOR AND SECURITY

To reiterate, the implementation of the WATCHWORD Generator can only enhance the security of the computer system. Some of the enhancements include, as a minimum:

- a. Providing the user community with a secure processing environment.
- b. Identifying and authenticating each user to ensure that they have access to information they need.
- c. Restricting sensitive data to only specific users who have access to review such data.
- d. Providing an extra layer of security for the user and the computer system in the event that the password is compromised.
- e. Reducing the probability that the computer system will be compromised.

Each user is assigned a unique PIN and secret key, however, the secret key is not known to the user. The secret key is entered into the WATCHWORD Generator by the WGA before it is issued to the user and is not accessible by the WATCHWORD Generator. Because each PIN and secret key is unique for each WATCHWORD Generator, a computer hacker would have to physically have the WATCHWORD Generator, userid, password, and PIN of the user whom account he/she is trying to compromise.

DOCKMASTER LOGIN WITH THE WATCHWORD GENERATOR

When a DOCKMASTER user logs in with the WATCHWORD Generator the sequence of identification and authentication begins. After the user enters his/her userid and password, DOCKMASTER will "Challenge" the user for a response. At this point the user must enter his/her PIN into the WATCHWORD Generator followed by the seven-digit system "Challenge". The WATCHWORD Generator will generate a seven-digit "Response" that the user will enter into DOCKMASTER. If the user has correctly entered in his/her userid, password, PIN, Challenge, and Response, DOCKMASTER will allow the user access to the system. If any of the above elements were entered incorrectly, DOCKMASTER will not grant access to the system.

If the PIN is entered incorrectly, the secret key will be unable to generate a correct response to the "Challenge". Although a "Response" will be generated, it will not be correct, therefore the user will not gain access to DOCKMASTER. Also if the "Challenge" is entered incorrectly into the WATCHWORD Generator, a "Response" will be generated for that "Challenge" not the system generated "Challenge". Since the wrong "Challenge" was entered, thus generating an incorrect "Response", DOCKMASTER would deny the user access to the system.

FUTURE OF THE WATCHWORD GENERATOR ON DOCKMASTER

The WATCHWORD Generator has been an overwhelming success on DOCKMASTER. Although the implementation of the WATCHWORD Generator on DOCKMASTER caused minimal user frustration, the majority of the DOCKMASTER user population view the implementation as a positive step toward better computer security.

Where do we go from here? There are two options that the WATCHWORD Generator offer that can be utilized by the DOCKMASTER user community. The first option includes user authenticating login to DOCKMASTER. The user can send a "Challenge" to the host computer, DOCKMASTER, and the host computer will generate a "Response". If the correct "Response" is given, the user will know that he/she is logging into the correct computer system.

The second option includes issuing the user two PINs and secret keys. The WATCHWORD Generator has the capability of storing two PINs and secret keys for user identification and authentication. This option will add another step to the identification and authentication sequence as well as enhance security. This option would be excellent for System Administrators. Because of the privileges that System Administrators have, this option would greatly decrease the chances of a computer hacker compromising a System Administrator's account.

Although neither of the options are being implemented on DOCKMASTER in the near future, the options still remain open. Before either option is implemented, a need assessment will be thoroughly conducted and based on the conclusions the options may or may not be implemented.

CONCLUSION

With the growing concern for computer security, the implementation of the WATCHWORD Generator on DOCKMASTER has greatly reduced the chances of the system being compromised. Although no system is one hundred percent capable of preventing a successful penetration, the WATCHWORD Generator does provide that extra layer of security.

The advantages of implementing a smart token on a computer system outweighs the disadvantages considerably. Providing a secure processing environment for computer users is one of the the main concerns of computer security and the implementation of a smart token would be a step in the right direction for ensuring computer security.

14TH NATIONAL COMPUTER SECURITY CONFERENCE

**October 1-4, 1991
Omni Shoreham Hotel
Washington, D.C.**



**PROCEEDINGS
VOLUME II**

**Information Systems Security:
Requirements & Practices**

Welcome!

The National Computer Security Center (NCSC) and the Computer Systems Laboratory (CSL) are pleased to welcome you to the Fourteenth Annual National Computer Security Conference. We believe that the Conference will stimulate a vital and dynamic exchange of information and foster an understanding of emerging technologies.

The theme for this year's conference, "Information Systems Security: Requirements & Practices," reflects the continuing importance of the broader information systems security issues facing us. At the heart of these issues are two items which will receive special emphasis this week -- Information Systems Security Criteria (and how it affects us) and Education, Training, and Awareness. We are working together, in the Government, Industry, and Academe, in cooperative efforts to improve and expand the state-of-the-art technology to information systems security. This year we are pleased to present a new track emphasizing the integration of information security solutions. These presentations will provide you with some thoughtful insights as well as innovative ideas in developing your own solutions. Additionally, we will be presenting an educational program which addresses the automated information security responsibilities. This educational program will refresh us with the perspectives of the past, and will project directions of the future.

We firmly believe that security awareness and responsibility are the cornerstone of any information security program. For our collective success, we ask that you reflect on the ideas and information presented this week; then share this information with your peers, your management, your administration, and your customers. By sharing this information, we will develop a stronger knowledge base for tomorrow's foundations.


JAMES H. BURROWS
Director
Computer Systems Laboratory


PATRICK R. GALLAGHER, JR.
Director
National Computer Security Center

Engr
QA
76
.9
A25
N27
1991
V.2

Dr. Marshall Abrams
James P. Anderson
Jon Arneson
Devolyn Arnold
James Arnold
Al Arsenault
V.A. Ashby
David Balenson
Dr. D. Elliott Bell
James W. Birch
W.Earl Boebert
Dr. Martha Branstad
Dr. John Campbell
Lisa Carnahan
R.O. Chester
David Chizmadia
Dorothea deZafra
Donna Dodson
Karen Doty
Dr. Deboah Downs
Jared Dreicer
Ellen Flahavin
Daniel Gambel
L. Dain Gary
Virgil Gibson
Dennis Gilbert
Irene Gilbert
Captain James Goldston, USAF
Dr. Joshua Guttman
Douglas Hardie
Ronda Henning
Dr. Harold Highland, FICS
Jack Holleran
Hilary H. Hosmer
Russell Housley
Howard Israel
Dr. Sushil Jajodia
Wayne Jansen

National Institute of Standards and Technology
Department of Defense
Department of Defense
Air Force Academy
The MITRE Corporation
Trusted Information Systems, Inc.
Trusted Information Systems, Inc.
Secure Systems, Inc.
Secure Computing Technology Corporation
Trusted Information Systems, Inc.
Department of Defense
National Institute of Standards and Technology
Martin Marietta
Department of Defense
Public Health Service
National Institute of Standards and Technology
CISEC
The AEROSPACE Corporation
Los Alamos National Laboatory
National Institute of Standards and Technology
Grumann Data Systems
Mellon National Bank
Grumann Data Systems
National Institute of Standards and Technology
National Institute of Standards and Technology
AFCSC
The MITRE Corporation
Unisys Corporation
Harris Corporation
Compulit, Inc.
National Computer Security Center
Data Security, Inc.
XEROX Information Systems
AT&T Bell Laboratories
George Mason University
National Institute of Standards and Technology

Conference

Referees

Carole Jordan
 Dr. Maria M. King
 Leslee LaFountain
 Steven LaFountain
 Paul A. Lambert
 Dr. Carl Landwehr
 Robert Lau
 Dr. Theodore M.P. Lee
 Steven B. Lipner
 Teresa Lunt
 Dr. William V. Maconachy
 Sally Meglathery
 Dr. Jonathan Millen
 Warren Monroe
 William H. Murray
 Noel Nazario
 Ruth Nelson
 Peter Neumann
 J.D. Nichols
 Steven Padilla
 Nick Pantiuk
 Donn Parker
 Richard Pethia
 Dr. Charles Pfleeger
 Kenneth Rowe
 Professor Ravi Sandhu
 Marvin Schaefer
 Dr. Roger R. Schell
 Emilie J. Siarkiewicz
 Suzanne Smith
 Brian Snow
 Professor Eugene Spafford
 Mario Tinto
 James Tippet
 Eugene Troy
 LTC. R. Vaughn, USA
 Grant Wagner
 Kenneth vanWyk
 Howard Weiss
 Roy Wood
 Carol Worden

Defense Investigative Service
 The AEROSPACE Corporation
 Department of Defense
 Department of Defense
 Motorola GEG
 Naval Research Laboratory
 Department of Defense
 Trusted Information Systems, Inc.
 Digital Equipment Corporation
 SRI International
 National Security Agency
 ISSA
 The MITRE Corporation
 Hughes Aircraft
 Deloitte & Touche
 National Institute of Standards and Technology
 GTE
 SRI International
 Independent Consultant
 SPARTA
 Grumann Data Systems
 SRI International
 Carnegie Mellon University
 Trusted Information Systems, Inc.
 Department of Defense
 George Mason University
 Trusted Information Systems, Inc.
 GEMINI
 Rome Air Defense Center
 Los Alamos National Laboatory
 Department of Defense
 Purdue University
 Department of Defense
 Department of Defense
 National Institute of Standards and Technology
 U.S. Naval Academy
 Department of Defense
 Carnegie Mellon University
 SPARTA
 Department of Defense
 State of Minnesota

14th National Computer Security Conference

Table of Contents

x Authors Cross Index

Tutorials

- 1** From Tuples to Trusted Subjects to TDI: A Brief Tutorial on Trusted Database Management Systems
John R. Campbell, National Security Agency
- 13** Tutorial Series on Trusted Systems
Joel E. Sachs, Dr. William F. Wilson, Arca Systems, Inc.

PAPERS (refereed)

- 15** Accreditation Strategy for the Air Force Satellite Control Network (AFSCN)
Lt Col William Price, USAF, Air Force Space Command
Michael O'Neill, Frank White, CTA, Inc.
- 25** An Analysis of Application Specific Security Policies
Daniel F. Sterne, Martha Branstad, Trusted Information Systems, Inc.
Brian Hubbard, SPARTA, Inc.
Barbara Mayer, Atlantic Research Corporation
Dawn Wolcott, MITRE Corporation
- 37** Another Factor in Determining Security Requirements for Trusted Computer Applications
David Ferraiolo, National Institute of Standards and Technology
Karen Ferraiolo, Grumman Data Systems
- 45** Apparent Differences Between the U.S. TCSEC and the European ITSEC
Dr. Martha Branstad, Dr. Charles Pfleeger,
Trusted Information Systems, Inc.
Dr. David Brewer, Gamma Secure Systems, Ltd.
Mr. Christian Jahl, Mr. Helmut Kurth, IAGB Software Technology
- 59** Auditing of Distributed Systems
D. Banning, G. Ellingwood, C. Franklin, C. Muckenhirn, D. Price,
SPARTA, Inc.
- 69** Building a Multi-Level Application on an Untrusted DBMS in a UNIX System V/MLS Environment - A Project's Experience
David S. Crawford, Canadian Department of National Defence
- 78** Building a Multi-Level Secure TCP/IP
Deborah A. Fitcher, Brian K. Yasaki, The Wollongong Group
Ron L. Sharp, AT&T Bell Laboratories
- 88** The Cascade Problem: Graph Theory Can Help
John A. Fitch, III, Lance J. Hoffman, George Washington University

- 101 **A Case Study for the Approach to Developing a Multilevel Secure Command and Control Information System**
James Obal, Supreme Allied Commander Atlantic
William Grogan, Contel Federal Systems
- 110 **Contractors and Computer Security--Awareness, Education, and Performance**
Ronald E. Brunner, Ronald G. Brunner & Associates
- 120 **Covert Channel Analysis Planning for Large Systems**
Lee Badger, Trusted Information Systems, Inc.
- 137 **Dealing With a Malicious Logic Threat: A Proposed Air Force Approach**
Howard L. Johnson, Information Intelligence Sciences
Chuck Arvin, Earl Jenkinson, CTA, Inc.
Captain Bob Pierce, USAF, Electronic Security Command
- 147 **Developing Applications on LOCK**
Richard O'Brien, Clyde Rogers, SCTC
- 157 **The Development of a Low-To-High Guard**
Michelle J. Gosselin, MITRE Corporation
- 167 **DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype**
Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che-Lin Ho,
Karl N. Levitt, Biswanath Mukherjee, University of California, Davis
Stephen E. Smaha, Steven R. Snapp, Haystack Laboratories, Inc.
James Brentano, Pacific Gas and Electric Company
Lt. Tim Grance, USAF, Daniel M. Teal, USAF,
United States Air Force Cryptologic Support Center
Douglass L. Mansur, Lawrence Livermore National Laboratories
- 177 **A Distributed Implementation of the Transform Model**
Ravi S. Sandhu, Gurpreet S. Suri, George Mason University
- 188 **Employee Privacy and Intrusion Detection Systems: Monitoring on the Job**
Lorrayne J. Schaefer, The MITRE Corporation
- 195 **Experience of Commercial Security Evaluation**
Peter Fagan, Julian Straw, Secure Information Systems Limited
- 205 **Experiences in Multi-Level Security on Distributed Architectures**
Karl A. Siil, AT&T Bell Laboratories
- 215 **An Expert System Application for Network Intrusion Detection**
Kathleen A. Jackson, David H. DuBois, Cathy A. Stallings,
Los Alamos National Laboratory
- 226 **Formal Verification of a Network Security Device: A Case Study**
Hicham N. Adra, William Sandberg-Maitland,
CGI Information Systems & Management Consultants
- 237 **A Framework for Advancing Integrity Standardization**
Terry Mayfield, Stephen R. Welke, John M. Boone,
Catherine W. McDonald, Institute for Defense Analyses

- 246 **A Framework for Developing Accreditable MLS AISs**
R. K. Bauer, J. Sachs, M. L. Weidner, W. F. Wilson, Arca Systems, Inc.
- 257 **Generalized Framework for Access Control: Towards Prototyping the ORGCON Policy**
Marshall D. Abrams, Jody Heaney, Osborne King, Leonard LaPadula, Manette Lazear, Ingrid Olson, The MITRE Corporation
- 267 **Honest Databases That Can Keep Secrets**
Ravi Sandhu, Sushil Jajodia, George Mason University
- 283 **Identifying and Controlling Undesirable Program Behaviors**
Maria M. King
- 295 **Improvement of Data Processing Security by Means of Fault Tolerance**
Gilles Trouessin, Yves Deswarte, Jean-Charles Fabre, LAAS-CNRS & INRIA
Brian Randell, Computing Laboratory, The University Newcastle upon Tyne
- 305 **Information Security: Can Ethics Make a Difference**
Corey D. Schou, John A. Kilpatrick, Idaho State University
- 313 **Information Security Risk Analysis and Risk Management: Which Approach?**
Professor J.H.P. Eloff, K.P. Badenhorst, Rand Afrikaans University
- 328 **Information Systems Security: A Comprehensive Model**
Capt. John R. McCumber, USAF, Joint Staff, the Pentagon
- 338 **Integrating B2 Security into a UNIX System**
Kevin Brady, UNIX System Laboratories, Inc.
- 347 **Knowledge Based Computer Security Advisor**
William Huntman, M. B. Squire, Los Alamos National Laboratory
- 357 **The Logistics of Distributing a Smart Token**
Dawn Brown, Department of Defense
- 362 **A Method to Detect Intrusive Activity in a Networked Environment**
L. Todd Heberlein, Biswanath Mukherjee, Karl Levitt, University of California
- 372 **Model Based Intrusion Detection**
Thomas D. Garvey, Teresa F. Lunt, SRI International
- 386 **Notification: A Practical Security Problem in Distributed Systems**
Vijay Varadharajan, Hewlett-Packard Laboratories
- 397 **Output Perturbation Techniques for the Security of Statistical Databases**
Kasinath C. Vemulapalli, Elizabeth A. Unger, Kansas State University
- 407 **An Overview of Informix-Online/Secure**
Rammohan Varadarajan, Informix Software, Inc.
- 417 **Peeling the Viral Onion**
Russell Davis, Planning Research Corporation, Inc.

- 427 **Practical Models for Threat/Risk Analysis**
Mark W.L. Dennison, Kalman C. Toth
CGI Information Systems & Management Consultants, Inc.
- 436 **Predicate Differences and the Analysis of Dependencies in Formal Specifications**
D. Richard Kuhn, National Institute of Standards and Technology
- 446 **Preventing Weak Password Choices**
Eugene H. Spafford, Purdue University
- 456 **Putting Policy Commonalities to Work**
D. Elliott Bell, Trusted Information Systems, Inc.
- 472 **Reconciling CMW Requirements with Those of X11 Applications**
Glenn Faden, Sun Microsystems, Inc.
- 480 **Restating the Foundations of Information Security**
Donn Parker, SRI International
- 494 **The Role Of Network Security In A Methodology For Information Security Design And Implementation**
Professor J.H.P. Eloff, Mr. A.J. Nel, Rand Afrikaans University
- 505 **A Secure European System for Applications in a Multi-vendor Environment (The SESAME Project)**
T. A. Parker, ICL Secure Systems
- 514 **A Secure Quorum Protocol**
Masaaki Mizuno, Mitchell L. Nielsen, Kansas State University
- 524 **Security Guidance for VAX/VMS Systems**
Debra L. Banning, SPARTA, Inc.
- 533 **Sneakernet: Getting a Grip on the World's Largest Network**
Captain James B. Hiller, USAF, Space and Warning Systems Center
- 543 **A Socio-Technical Analysis of a USA National Computer Security Conference**
Stewart Kowalski, Stockholm University & Royal Institute of Technology
- 533 **Standardized Certification**
Captain Charles R. Pierce, USAF, Air Force Cryptologic Support Center
- 563 **A Strategic Framework For Information Security Management**
Rolf Moulton, BP America
Santosh Misra, Cleveland State University
- 572 **A System Security Engineering Process**
J. D. Weiss, AT&T Bell Laboratories
- 582 **Teaching Computer Systems Security in an Undergraduate Computer Science Curriculum**
Alfred W. Arsenault, Captain Gregory B. White, USAF,
U. S. Air Force Academy
- 598 **Toward Certification, A Survey of Three Methodologies**
Captain Charles R. Pierce, USAF, Air Force Cryptologic Support Center

- 608 **Trusted Distributed Computing: Using Untrusted Network Software**
E. John Sebes, Richard J. Feiertag, Trusted Information Systems
- 619 **Trusting X: Issues in Building Trusted X Window Systems or What's not Trusted About X?**
Jeremy Epstein, TRW Systems Division
Jeffrey Picciotto, MITRE Corporation
- 630 **Using Existing Management Processes to Effectively Meet the Security Plan Requirement of the Computer Security Act: The IRS Experience**
Richard A. Stone, Joseph Scherer, Internal Revenue Service
- 634 **Viruses in an OS/2 Environment: Remembrances of Things Past and a Harbinger of Things to Come**
Kevin P. Haney, National Institutes of Health
- 644 **Why Does Trusted Computing Cost So Much?**
Susan Heath, Phillip Swanson, Daniel Gambel, Grumman Data Systems

PANEL Executive Summaries (unrefereed)

- 654 **PANEL: Acquiring Computer Security Services and Integrating Computer Security and ADP Procurement**
Dennis Gilbert, National Institute of Science and Technology
Barbara Guttman, National Institute of Science and Technology
- 655 **PANEL: Compartmented Mode Workstation(CMW) Program Overview**
Steven Schanzer, Moderator, Defense Intelligence Agency
- 658 **PANEL: The Computer Emergency Response Team System (CERT System)**
E. Eugene Schultz, Lawrence Livermore Laboratory
Richard Pethia, Software Engineering Institute, Carnegie Mellon University
- 663 **PANEL: Computer Security Management and Planning**
Christopher Bythewood, National Computer Security Center
- 664 **PANEL: Cracking the Cracker Problem**
Dorothy E. Denning, Moderator, Georgetown University
- 665 **The Role of Technology**
Matt Bishop, Dartmouth College
- 666 **PANEL: Electronic Dissemination of Computer Security Information Executive Summary**
Marianne Swanson, National Institute of Science and Technology
- 667 **What Can Dockmaster Offer You?**
Cindy Hash, Department of Defense
- Session: Guidelines & Evaluations**
- 669 **Towards Mutual Recognition of Security Evaluations**
Andrea Arnold, Digital Equipment Corp
Cornelia Persy, SIEMENS
Gottfried Sedlak, IBM

- 674 **PANEL:** Fielding COTS Multilevel Security Solutions: The Next Step
James Litchko, Trusted Information Systems Inc.
- 675 **PANEL:** Inference and Aggregation in Multilevel Databases: Research Directions
Teresa F. Lunt, Moderator, SRI International
- 676 Detecting and Evaluating Inference Channels
Thomas D. Garvey, SRI International
- 679 Inference Prevention in Databases: Data Design vs. Query Processing
Catherine Meadows, Naval Research Laboratory
- 680 Challenges in Addressing Inference and Aggregation
LTC. Gary Smith, USA, National Defense University
- 681 Approaches to Handling the Inference Problem
Bhavani Thuraisingham, The MITRE Corporation
- 684 **PANEL:** Military and Telecommunications Security: Specialized Methods
Richard Lefkon, Moderator, New York University
- 685 Malicious Code Prevention for Embedded Computer Weapon Systems
Debra L Banning, Gail M. Ellingwood, SPARTA
- 689 Computer Viruses as Electronic Warfare
Myron Cramer, Booz-Allen & Hamilton
- 690 Preventing Virus Insertion Through Switches
Ed Fulford, Northern Telecom
- 693 Nuclear Disaster and The Millennium Horse
Richard Lefkon, New York University
- Session:** National Issues
- 695 Reduced Defense Spending Increases the Need for Trusted Systems
Carole S. Jordan, Defense Investigative Service
- 696 **PANEL:** 1991: A Year of Progress in Trusted Database Systems
John R. Campbell, Moderator, National Security Agency
- 698 Recent Developments in Some Trusted Database Management Systems
Bhavani Thuraisingham, The MITRE Corporation
- 701 Oracle and Security: Year in Review 1990-91
Linda L. Vetter, Oracle Secure Systems
- 704 1991 SYBASE Secure Products: Executive Summary
Helena B. Winkler-Parenty, SYBASE
- 706 **PANEL:** Requirements and Experiences
Dennis Gilbert, National Institute of Science and Technology
- 708 **PANEL:** Risk Management
Irene Gilbert, National Institute of Science and Technology
- 709 **PANEL:** Specifying, Procuring, and Accrediting MLS System Solutions
Joel E. Sachs, Arca Systems, Inc.
- 714 **PANEL:** Trusted Applications in the Real World
Stephen Walker, Trusted Information Systems Inc.
- 715 **PANEL:** Winning Strategies in Information Systems Security Education, Training, and Awareness
W. V. Maconachy, Moderator, Department of Defense

Authors Cross Index

Abrams, Marshall D.	257	Franklin, C.	59
Adra, Hicham N.	226	Fulford, E.	690
Arsenault, Alfred W.	582	Futcher, Deborah A.	78
Arvin, Chuck	137	Gambel, Daniel	644
Arnold, Andrea	669	Garvey, T. D.	372, 676
Badenhorst, K.P.	313	Gilbert, Dennis	654, 706
Badger, Lee	120	Gilbert, Irene	708
Banning, D. -	59, 524, 685	Goan, Terrance L.	167
Bauer, R. K.	246	Gosselin, Michelle J.	157
Bell, D. Elliott	456	Grance, Tim, Lt.	167
Bishop, M.	665	Grogan, William	101
Boone, John M.	237	Guttman, Barbara	654
Brady, Kevin	338	Haney, Kevin P.	634
Branstad, Martha	25, 45	Hash, Cindy	667
Brentano, James	167	Heaney, Jody	257
Brewer, David	45	Heath, Susan	644
Brown, Dawn	357	Heberlein, L. T.	167, 362
Brunner, Ronald E.	110	Hiller, J. B., Capt	532
Bythewood, C	663	Ho, Che-Lin	167
Campbell, John R.	1, 696	Hoffman, Lance J.	88
Cramer, Myron	689	Hubbard, Brian	25
Crawford, David S.	69	Hunteman, William	347
Davis, Russell	417	Jackson, Kathleen A.	215
Denning, Dorothy E.	664	Jahl, Christian	45
Dennison, Mark W.L.	427	Jajodia, Sushil	267
Deswarte, Yves	295	Jenkinson, Earl H.	137
Dias, Gihan V.	167	Johnson, Howard	137
DuBois, David H.	215	Jordan, Carole	695
Ellingwood, G. M.	59, 685	Kilpatrick, John A.	305
Eloff, J.H.P.	313, 494	King, Maria M.	283
Epstein, Jeremy	619	King, Osborne	257
Fabre, Jean-Charles	295	Kowalski, Stewart	543
Faden, Glenn	472	Kuhn, D. Richard	436
Fagan, Peter	195	Kurth, Helmut	45
Feiertag, Richard J.	608	LaPadula, Leonard	257
Ferraiolo, David	37	Lazear, Manette	257
Ferraiolo, Karen	37	Lefkon, Richard	684, 693
Fitch, John A., III,	88	Levitt, Karl N.	167, 362

Authors Cross Index

Litchko, James	674	Schultz, E. Eugene	658
Lunt, Teresa F.	372, 675	Sebes, E. John	608
Maconachy, W. V.	715	Sedlak, Gottfried	669
Mansur, Douglass L.	167	Sharp, Ron L.	78
Mayer, Barbara	25	Siil, Karl A.	205
Mayfield, Terry	237	Smaha, Stephen E.	167
McCumber, John R.	328	Smith, Gary	680
McDonald, C. W.	237	Snapp, Steven R.	167
Meadows, Catherine	679	Spafford, Eugene H.	446
Misra, Santosh	563	Squire, M. B.	347
Mizuno, Masaaki	514	Stallings, Cathy A.	215
Moulton, Rolf	563	Sterne, Daniel F.	25
Muckenhirn, C.	559	Stone, Richard A.	630
Mukherjee, B	167, 362	Straw, Julian	195
Nel, A. J.	494	Suri, Gurpreet S.	177
Nielsen, Mitchell L.	514	Swanson, Marianne	666
Obal, James	101	Swanson, Phillip	644
O'Brien, Richard	147	Teal, Daniel M., Lt.	167
Olson, Ingrid	257	Thuraisingham, B.	681, 698
O'Neill, Michael	15	Toth, Kalman C.	427
ParkerDonn B.	480	Trouessin, Gilles	295
Parker, T. A.	505	Unger, Elizabeth A.	397
Pethia, Richard	658	Varadarajan, R.	407
Persy, Cornelia	669	Varadharajan, Vijay	386
Pfleeger, Charles	45	Vemulapalli, K. C.	397
Picciotto, Jeffrey	619	Vetter, Linda	701
Pierce, C. R. Capt	137, 533, 598	Walker, Stephen	714
Price, D.	59	Weidner, M. L.	246
Price, William, Lt Col	15	Weiss, J. D.	572
Randell, Brian	295	Welke, Stephen	237
Rogers, Clyde	147	White, Frank	15
Sachs, Joel E.	13, 246, 709	White, G. B., Capt	582
Sandberg-Maitland, W.	226	Wilson, W. F., Dr.	13, 246
Sandhu, Ravi S.	177, 267	Winkler-Parenty, H.	704
Schaefer, Lorrayne J.	188	Wolcott, Dawn	25
Schanzer, Steven	655	Yasaki, Brian K.	78
Scherer, Joseph	630		
Schou, Corey D.	305		

A Method to Detect Intrusive Activity in a Networked Environment¹

L.T. Heberlein, K.N. Levitt, B. Mukherjee

Computer Security Laboratory
Division of Computer Science
University of California
Davis, Ca. 95616

ABSTRACT

Intrusive activity is occurring on our computer systems, and the need for intrusion detection has been demonstrated. This paper discusses some of the benefits and drawbacks of trying to detect the intrusive activity by analyzing network traffic. A general solution, based on detecting and analyzing abstract objects, is formulated. Finally, results from applying the solution are presented.

1. Introduction

Computers are the targets of attacks [3]. Reports appear in the media almost weekly about outsiders breaking into computers, employees misusing computers, and rogue viruses and worms penetrating computer systems. Incidents such as the internet worm of 1988 [3], the Wank worm [3], and the Netherland hackers have gained international recognition, and they serve to emphasize the vulnerability of computer systems around the world.

These reported incidents are cases of intrusive activity in our computer systems. Intrusive activity can be defined as any attempt which, if successful, will result in one of the following:

- disclosure of information against the wishes of the owner of the information
- modification of information against the wishes of the owner of the information
- denial of the use of services by legitimate users of the system
- use of resources against the wishes of the system's owner (e.g. disk or CPU)

The first three bulleted items are discussed in [4]. The last bulleted item, the stealing of resources, covers actual observed activity which did not fit easily into the three previous categories. For example, using our network security monitor (NSM) [8], we have observed an intruder use a system to crack password files. The intruder was not interested in either looking at existing information on the system, modifying information on the system, or denying resources to legitimate user. The intruder simply used the CPU, when it was idle, to crack passwords.

Authentication and access control mechanisms are designed to guard against intrusive activity; however, these mechanisms have not been wholly successful. Failure of these mechanisms is due in part to the ease by which passwords can be compromised, failure by system administrators and users to properly use the access control mechanisms, poor operating system designs, and flawed operating system implementations (i.e., bugs).

The failures of authentication and access control mechanisms are compounded by the decentralization of computer systems and the increased access to a computer system by computer networks. The decentralization of computer systems is the movement away from a single mainframe computer to multiple workstations and personal computers. The movement is fueled by the increasing power and decreasing costs of workstations and personal computers. The result of decentralization is a type of computer system which is administered by people, usually the user community, with little or no formal training in system administration or computer security. This in turn results in a greater chance for poorly configured authentication and access control mechanisms.

Connecting a computer to a network also increases the chances of intrusive activity occurring on that computer since this process increases the number of people who can potentially access it. Connecting a computer to a network provides a path to that computer for every user with access to the network. If the

¹ This work is supported in part by Lawrence Livermore National Laboratory

network is part of the internet, essentially everyone with access to a telephone has a path to that computer.

With the realization that current authentication and access control mechanisms have not provided adequate security against intrusive behavior, institutions which use computers and computer networks have become interested in detecting the intrusive activity which is occurring. If an intrusion can be detected, an institution can at least know from where intrusive activity is coming, how the activity is being perpetrated (and therefore, hopefully how to stop it), and what data have been compromised.

In the summer of 1988, University of California at Davis and Lawrence Livermore National Laboratory began an effort to detect intrusive activity on a network of heterogeneous computer systems. A brief overview of this effort is presented in section two. Sections three and four present the mechanisms by which our monitor detects intrusive activity. And section five presents some of the results of our efforts as well as directions for future research.

2. Network Monitor

Intrusion detection systems examine available sources of information about the various operations in a computing system to determine if intrusive activity is occurring. The main source of information for most intrusion detection system is the audit trails generated by the operating system. Although the audit-trail-based analysis has provided a measure of success, a number of limitations exist with this method. First, audit trails traditionally do not provide much of the information necessary to perform security analysis. This is due in part to the historical purpose of audit trail collection - the billing of customers. Second, audit trails tend to be system specific. Each operating system provides a different set of information in a different format. An intrusion detection system designed to work on the Multics operating system's audit trails would need a great deal of restructuring to operate on another operating system's audit trails. Third, the collection of audit trails is expensive in terms of CPU usage and storage utilization. Many organizations, even those working in the field of computer security, turn off auditing on their machines to avoid the resource penalty. Fourth, the audit trails themselves can be the target of an intruder. Intruders have been known to turn off auditing on machines in order to hide their tracks. Fifth, and last, the delay in the actual recording and analysis of the audit information can allow an intruder to do damage and exit the machine before the intrusion is noticed [17]. So, although there exists a strong desire for immediate notification of intrusive activity, audit mechanisms can introduce a delay factor.

By exploiting the broadcast property of a local area network (LAN) and network protocol standards, the analysis of network traffic can solve a number of the drawbacks associated with audit-trail-based analysis. First, network standards exist by which a variety of hosts can communicate. An intrusion detection system based on network traffic can therefore simultaneously monitor a number of hosts consisting of different hardware and operating system platforms. Second, the collection of network traffic does not create any performance degradation on the machines being monitored, so network monitoring is more attractive to a user community which places importance in the performance and responsiveness of their machines. Third, since a network monitor can be logically isolated from the computing environment, its analysis cannot be compromised by an intruder. Typically, the intruder has absolutely no way of knowing that the network is being monitored. And fourth, since a network monitor draws its information directly from the network, no delay occurs from the instant an intrusion occurs until the instant the evidence is available. Instead, intrusive activity can be observed as it occurs.

The original work on this type of network monitoring was based on simple traffic analysis: modelling the flow of data among the different machines [9,10]. In [9,10], network traffic is modelled with a concept called a data path. A data path is a method by which one machine can communicate with a second machine. A data path is defined by the three-tuple `<src_host, dst_host, network_service>`. If the traffic flow shifted (e.g., a new data path is observed) at any point, this information would be reported as a possible intrusion. For example, a particular host initiating a login to a host to which it has never logged into before would be considered suspicious. This work was based on Denning's hypothesis that intrusive activity would manifest itself as anomalous behavior [2].

Although this method showed early promise, a major drawback quickly became apparent: the information available from simple network packet analysis was at a level much too low to detect subtle intrusive activity. For example, an intrusion over a commonly used data path would not be detected. Unfortunately, this is often the case when the intrusion is being perpetrated by an insider.

To provide for a more effective intrusion detection system, our monitor needed the capability to detect

and analyze higher-level objects which are not directly observed (i.e., individual network connections and hosts). Also, to perform the analysis information about each object-attributes for the object-needed to be known.

The logical architecture of our system is shown in figure 1, and the components which provide for the additional complexity of analysis, viz. object detector and object analyzer, are shown in the dashed box. The functionality provided by these components have greatly enhanced our efforts to detect intrusive activity. Results from actual use of our monitor can be found in [7,8]. We have attempted to both generalize and formalize the methods by which our monitor detects and analyzes objects, and this work is presented in sections three and four.

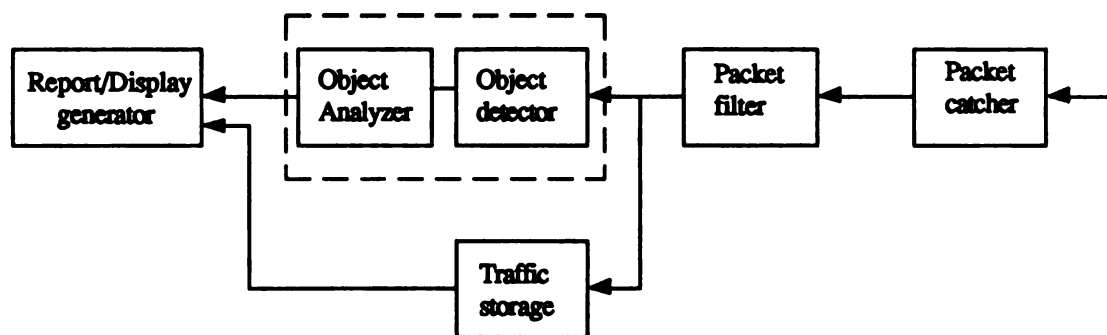


Figure 1

3. System Description Language

The problem of detecting intrusive activity in a heterogeneous network of computers through the observation of network packets can be generalized to the detection of a behavior in a complex system (e.g., networked system) from the analysis of low level information (e.g., network packets). The complex system is composed of a variety of components (i.e. hosts, connection, and packets) each of which in turn may be composed of other components, but only the simplest of components, the lowest levels of information, are directly visible to a monitor. Unfortunately, to detect the behavior of interest (i.e. intrusive activity), the complex components which are not directly observed, as well as the low level components, must be examined for the manifestation of the behavior.

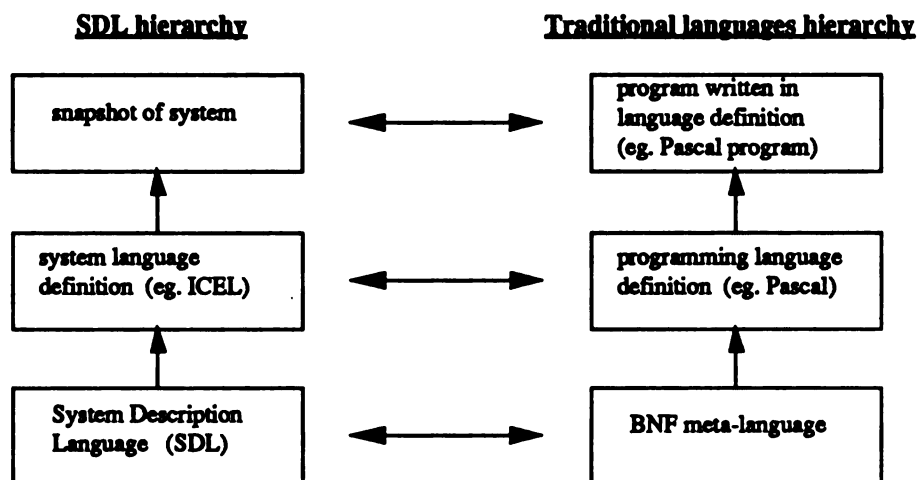


Figure 2

To provide for a formal mechanism to infer the complex components of a system, we have defined a meta-language, called the system description language (SDL), to describe the relationships among components of a system. The description of a system with this language is called its system language definition. As the

low level information is observed, the system language definition is used to infer the existence of the complex objects and the relationships between them. A snapshot of all the low level objects and the inferred complex objects and their relationships to one another represent a model of the actual system at a particular time instant. It is this model which will be examined for the manifestation of the behavior of interest.

The SDL, the system language definition, and the snapshot of an actual system have a direct resemblance to the definition of a traditional programming language. The SDL provides a functionality similar to that of the BNF meta-language. The system language definition is similar to a traditional program language definition (e.g., Pascal). And the snapshot of a system is similar to a program defined by a traditional programming language. This relationship is shown in figure 2.

The system description language is the focus of this section. Section 3.1 introduces the issues which must be addressed by the system description language. Section 3.2 presents a review of attribute grammars, the ancestor of the system description language. And section 3.3 discusses the actual system description language.

3.1 Issues to be Addressed by the SDL

To design a meta-language which can be used to describe and model complex systems from the observation of low level information, a number of issues must be addressed. First, how are the low level, simple components of the system detected, and how are the attributes of each low level object determined? We have chosen to not address this issue in this paper, and it is not part of the language definition. The low level components are detected, and their associated attributes are determined by a preprocessor. This is not unlike the design of conventional programming languages which assume the presence of a lexical analyzer to detect tokens, and, if necessary, determine their attributes.

The second issue is the identification and representation of components of the system which are not observed directly. In fact, a complex object which does not have a real world counterpart may be desired. For example, our model for the computer network environment includes an object called a "service-set." The service-set object does not exist in the actual system, but its presence is helpful in analyzing other components such as network connections. The system description language must provide a mechanism for inferring the existence of these unobserved, perhaps nonexistent, objects. Furthermore, the language must provide mechanisms to determine enough information about these abstract objects so they can be analyzed for the behavior of interest.

The third issue is concerned with the transitory nature of many of the objects in a system. Systems such as a heterogeneous network have a number of components which exist for a time, and then disappear. For example, network connections are created and destroyed continuously. The system description language must be able to handle the creation and destruction of components, and the system description language must provide information to determine when a component should be created or destroyed. Thus the model of an actual system, as determined by a system language definition, can change over time.

In summary, the system description language assumes that the low level, simple components and their attributes are provided to it. From these simple components, the systems description language must provide a mechanism to infer the existence of, and the relationships between, complex objects. The system description language must provide mechanisms to determine enough information about the complex objects to analyze the objects for the presence of the behavior of interest. Finally, the system description language must provide a means both to determine when a component to the system is created or destroyed and to modify the model of the system due to the creation or destruction of a component.

3.2 Attribute Grammars

The system description language which satisfies the above requirements is built upon the concept of attribute grammars. A quick introduction to attribute grammars is provided below. Readers already familiar with this subject may want to skip to section 3.3.

An attribute grammar describes both the strings accepted by a language (e.g., the syntax of the language) and a method to determine the "meaning" of those strings (e.g., the semantics of the language). An attribute grammar consists of a context-free grammar, a set of attributes for each symbol in the grammar, and a set of functions defined within the scope of a production rule in the grammar to determine the values for the attributes of each symbol in that production [1]. The following example of an attribute grammar for the definition and interpretation of binary numbers² will be used to clarify the relationships between these three

² This example is taken from [12].

$N \rightarrow LL$	$N \rightarrow L_1 L_2$	$v(N) = v(L_1) + v(L_2)/2^{l(L_2)}$
$N \rightarrow L$	$N \rightarrow L$	$v(N) = v(L)$
$L \rightarrow LB$	$L_1 \rightarrow L_2 B$	$v(L_1) = 2v(L_2) + v(B), l(L_1) = l(L_2) + 1$
$L \rightarrow B$	$L \rightarrow B$	$v(L) = v(B), l(L) = 1$
$B \rightarrow 1$	$B \rightarrow 1$	$v(B) = 1$
$B \rightarrow 0$	$B \rightarrow 0$	$v(B) = 0$

A
B

Figure 3

components of an attribute grammar.

The context-free grammar for our language of binary numerals is defined by $G = (V, N, P, S)$ where V is the set of symbols, N is the set of nonterminal symbols, P is the set of production rules, and S , an element of N , is the start symbol. The set of terminal symbols, a subset of V , is $\{1, 0, .\}$. These are the ASCII characters one, zero, and period. The set of nonterminal symbols, N , is $\{B, L, N\}$. They represent the abstract objects bit, list of bits, and number. The start symbol for our attribute grammar for binary numbers is N , the abstract number. The set of production rules relating these symbols and providing the definition of acceptable strings is given in figure 3A.

By this context free grammar, we can see that the string 11.01 is an acceptable binary number. The parse tree for this string is given in figure 4A.

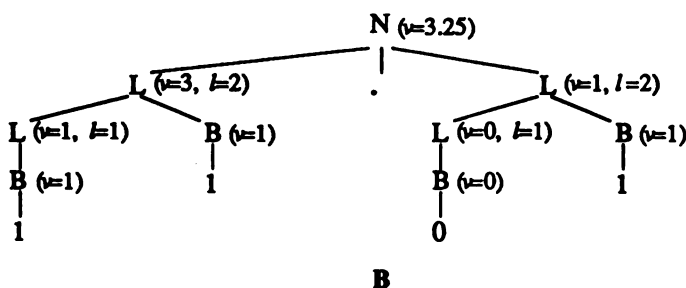
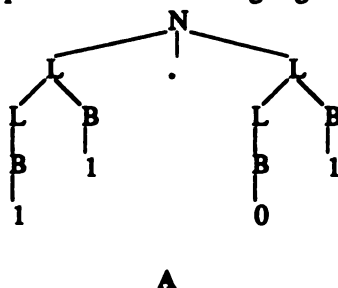


Figure 4

The context-free grammar can be used to build a parse tree of a string and determine whether the string is valid in the language; however, the context-free grammar cannot be used to determine the meaning of the string. The addition of attributes and attribute functions are necessary to determine the meaning of the string.

The set of attributes, A , for each nonterminal are given as follows: $A(B) = \{v\}$, $A(L) = \{v, l\}$, and $A(N) = \{v\}$. The attribute v is the value of a symbol, and the attribute l is the length of a symbol.

The set of functions defined within the scope of each production rule is given in figure 3B.

By using the attributes for each symbol and the attribute functions, we can now assign meaning to each symbol in the parse tree (see figure 4B). For our language of binary numbers, the most important meaning is that of the start symbol N . Our string 11.01 now has the meaning of 3.25.

3.3 System Description Language

This section introduces the system description language, an extension of attribute grammars. This system description language provides a structure by which a system's components and relationships between components can be described. The description, or system language definition, of a system can be used to both infer the existence of complex objects (e.g., determine the syntactic structure of the system) and assign "meaning" to these objects (e.g., the semantic information about the system). The meaning of an object, the values of its attributes, will be used to determine if the behavior of interest is present in any of the components of the system.

Similar to an attribute grammar, a system language definition written in the SDL consists of a structural grammar, a set of attributes for each object, or symbol, in the structural grammar, and a set of

functions defined within the scope of a production rule of the structural grammar which determine the attribute values for each object in that production.

3.3.1 Objects

Objects are the components of the system which will be modelled. These objects may or may not have real world counterparts. Two varieties of objects exist: basic objects and complex objects. Basic objects are the low-level components which are directly observed. These are similar to terminal symbols in traditional programming languages. Complex objects, on the other hand, are not observed and must be inferred from the observation of basic objects. Complex objects are similar to non-terminal symbols in traditional programming languages. These two objects are discussed further below.

3.3.1.1 Basic Objects

Basic objects are simple, indivisible components of the complex system being modelled; they are detected and their attributes determined by a preprocessor. This preprocessor performs the job of a lexical analyzer in traditional programming languages. Basic objects are treated as events; they only exist for the moment at which they are observed. For example, in the networked system, packets are basic objects. Basic objects for other systems may be an audit record from an operating system, a message to a spacecraft component, or a sampled data point from some measuring instrument.

A basic object type is defined by a name and a list of attributes. The name format for our system is the same as the standard C identifier. Attributes will be discussed in section 3.3.2. An example basic object representing a possible network packet is:

basic: packet { *attribute list* }

The keyword **basic** states that the following object type is a basic object, and the object type's name is **packet**. Attributes for this object will be discussed later in section 3.3.3.

3.3.1.2 Complex Objects

As mentioned previously, complex objects are components of a system which are not directly observed by the monitor, so they must be inferred from the observation of the basic objects. A complex object is composed of basic objects and/or other complex objects. For example, a complex object type called process may be defined for an audit-trail-based monitor. Although processes are not directly observed by the monitor, information about them can be inferred from the audit records. Therefore, in our model, processes are composed of audit records. Compositions will be discussed further in section 3.3.3.

A major difference between complex objects and basic objects is that complex objects have persistence. Whereas basic objects are treated as events, complex objects are treated as persistent elements which are created and possibly destroyed. The creation of a complex object occurs as soon as it can be inferred. The destruction of an object is considerably more difficult and depends on both the definition of the complex object and the existence of objects which compose the complex object. The two rules which govern the possible destruction of an object are described below.

First, if any object A exists and is part of an object B's composition, then object B should continue to exist. Second, if the last object which is part of object B's composition is destroyed, then object B will be destroyed after a specified time delay, Δt , unless another object which is part of B's composition is created or observed. This specified Δt is the value of a function associated with the object, and it may depend on the object's attributes.

Complex objects can be composed of only basic objects, only complex objects, or a combination of basic and complex objects. Complex object types are defined in my system by one of the following forms depending on their composition:

complex type *i*: *name* { *attribute list* }

where *i* varies from 1 to 3 depending on the makeup of the composition objects.

3.3.2 Attributes

As mentioned previously, each object has a set of attributes associated with it. These attributes provide a "meaning" to each object. It is the attributes which will be used to determine if the object is associated with a particular behavior. These attributes are also used, along with the production rules described in section 3.3.3, to determine if an object A is part of object B's composition.

Each attribute consists of a name and a type. The name is used to reference the value, and the type determines the value type which can be assigned or retrieved from the attribute. For example, "int value" would

describe an attribute of type "int" which is referenced by the name "value." Attribute types may be complex structures defined in the same format as complex types are described in the C language [11].

Many of the attribute values of an object will be assigned by the monitor. For example, when the existence of a new host is inferred, a host object is created and its internet address is immediately assigned by the monitor. The values of other attributes, however, are determined by attribute functions. Attribute functions, described in section 3.3.4, take as input attribute values associated with the object and possibly attribute values of other objects associated with it by the production rules (see section 3.3.3).

A complex object type to represent a stream (a unidirectional flow of data from one process to another process) composed of packets can now be defined as follows:

```

complex type 1:      stream{
                        inet_addr      src_addr
                        inet_addr      dst_addr
                        int             src_port
                        int             dst_port
                        int             creation_time
                        int             num_of_packets
                        int             num_of_bytes
                      }

```

This simple definition of a process has a simple identifier, *stream*, addresses for the source and destination hosts, source and destination ports to specify the processes on the two machines, a time of creation, the number of packets exchanged between the two processes, and the number of bytes in all the packets exchanged.

The set of attributes for an object *O* can be defined as $A(O) = \{a_1, a_2, \dots, a_n\}$. For example, $A(\text{process}) = \{\text{src_addr}, \text{dst_addr}, \text{src_port}, \text{dst_port}, \text{creation_time}, \text{num_of_packets}, \text{num_of_bytes}\}$.

3.3.3 Productions

Productions define the relationship between the different object types of a system. They define which types of objects compose a complex object, and they indicate how to determine which set of objects from an object type compose the complex object. A production rule has the form:

complex_object_type -> list of *object_composition*

The *complex_object_type* is simply the name of a complex object type (e.g., *stream*). An *object_composition* is a set defined by a tuple of the form $\langle \text{object_type}, \text{restrictions} \rangle$. The *object_type* is simply the name of one of the defined object types (basic or complex), and the restrictions determine which of all possible objects of type *object_type* are actually used to compose the complex object.

For example, let the complex object type called *stream* be defined as above, and let the object type called *packet* be defined as follows:

```

basic:      packet {
                        inet_addr      src_addr
                        inet_addr      dst_addr
                        int             src_port
                        int             dst_port
                        int             num_of_bytes
                        int             time
                      }

```

A production rule for the *stream* object can now be defined as follows:

```

stream -> packet
  where for all e ∈ packet
    e.src_addr = stream.src_addr
    & e.dst_addr = stream.dst_addr
    & e.src_port = stream.src_port
    & e.dst_port = stream.dst_port

```

Finally, each element of *packet* which composes a particular *stream* object is called a sub-component of the *stream* object, and the *stream* object is called a super-component of the *packet* objects. The concepts of sub-components and super-components will be used in section 4.2 to define integrated object analysis functions.

3.3.4 Attribute Functions

The attributes of a complex object which are not defined by the monitor when the object is inferred are defined by attribute functions. The attribute functions for a structural language are defined as they are for attribute grammars; however, special attention must be given to the format of the production and the restriction for the production. For example, an attribute function to determine the value for the attribute "num_of_bytes" of a stream object could be as follows:

$$\text{stream.num_of_bytes} = \sum_{i=1}^n e_i.\text{num_of_bytes}$$

Where $n = |\text{packet}|$, and each $e \in \text{packet}$ is assumed to be a sub-component of the stream object as defined by the restrictions in the production rule for stream objects.

4. Detecting Behaviors in Systems

Once the structural grammar, attributes, and attribute functions have been defined, a second set of functions, called behavior-detection functions, must be defined for each object in the structural grammar. Behavior-detection functions determine whether an object is associated with the particular behavior of interest. Because a behavior may manifest itself differently or more clearly in different object types, each object in a system parse tree (the snapshot of the system) must be examined for the behavior by particular behavior-detection functions designed for that object type. For each type of object, there will be two behavior-detection functions: the isolated behavior-detection function and the integrated behavior-detection functions. These two function types are discussed below.

4.1 Isolated Object Analysis

An isolated behavior-detection function for an object uses the attributes of that object to calculate the probability that the object is associated with the behavior of interest. In short, an isolated behavior-detection function is a classifier. With some preprocessing to transform the attribute types, a large number of classifiers can be used.

Unfortunately, classifiers generally have to be trained with sample data, and the behavior of interest is often quite rare. There are at least two possible solutions to the problem of lack of sample data: expert systems and single behavior classifiers. An expert system, designed by people knowledgeable about the problem domain, can use heuristics to determine how close an object's behavior is to the behavior of interest. A single behavior classifier is built around the assumption that a rare behavior will be significantly different than normal behavior [2]. If this is true, a single classifier can profile normal behavior, and then it could report any behavior which does not strongly resemble normal behavior. Work on such single behavior classifiers have been performed by SRI for IDES [13] and Los Alamos National Laboratory for Wisdom and Sense [17]. For our particular problem environment, we combined the efforts of both an expert system and a single behavior classifier.

4.2 Integrated Objects Analysis

An integrated behavior-detection function for an object modifies the result of the isolated behavior-detection function for the object by including the analysis of the isolated behavior-detection functions for sub-components and super-components of that object. The modification by an integrated behavior-detection function allows the inclusion of both the results of aggregated analysis (those from super-components) and the results of more detailed levels of analysis (those from sub-components). The integrated behavior-detection function can be implemented by a weighted average function such as:

$$\frac{W_1 * \text{Object_if} + W_2 * \text{Super_if} + W_3 * \text{Sub_if}}{W_1 + W_2 + W_3}$$

Where Object_if is the value calculated by the object's isolated behavior-detection function, Super_if is the average isolated behavior-detection function value for all the super-components, Sub_if is the average isolated behavior-detection function value for all the sub-components, and W_1 , W_2 , and W_3 are the weights.

The relationship between an object's attributes, isolated behavior-detection functions, and integrated behavior-detection functions can be seen in figure 5. In this example, we are interested in analyzing the object B_1 for a particular behavior. The object B_1 is composed of objects C_1 and C_2 , and it is part of the object A_1 . Result B_{1v} is the analysis of object B_1 in isolation, and result $B_{1v'}$ is the result after combining the result of B_{1v} with the results from objects C_1 , C_2 , and A_1 .

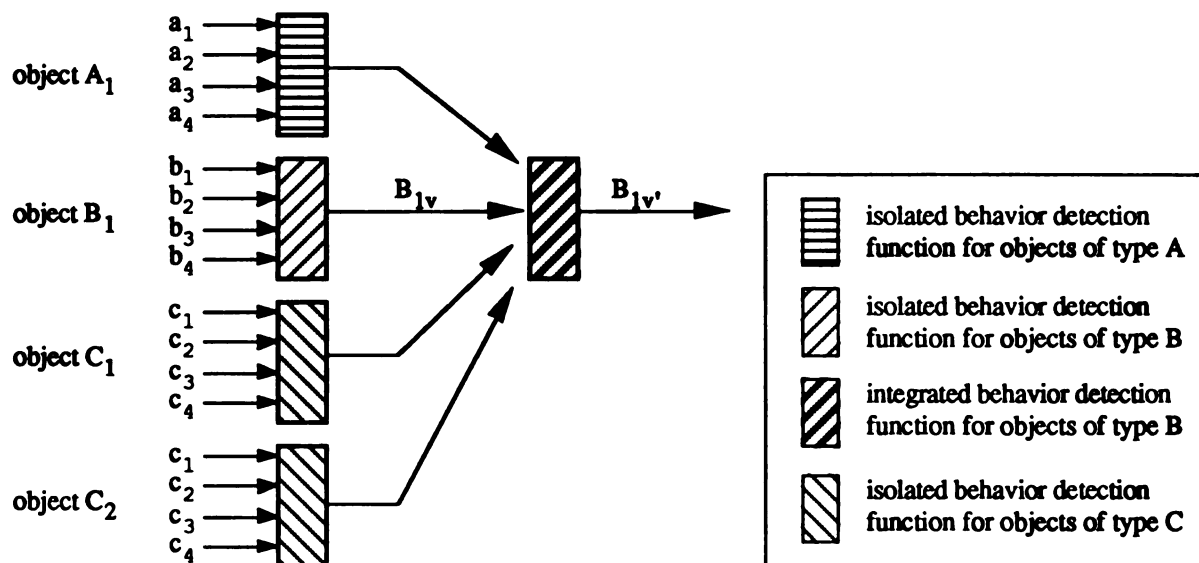


Figure 5

5. Results and Future Research

By using the system language definition for the networked environment described in [7], one programmer was able to code both the object detector and object analyzer modules in less than two weeks. Since the coding of these modules is a straight forward implementation of the system language definition, we hope to provide automatic development tools in the future which will automatically generate object detector and object analyzer modules from a system language definition.

We have concentrated our analysis efforts on an isolated behavior-detection function for connections. This function combines a simple anomaly detector, an attack model, and an expert system to arrive at a single suspicion value. The higher the suspicion value is, the more likely our monitor believes the connection is associated with intrusive activity.

We monitored the Electrical Engineering and Computer Science LAN at UCD for a period of approximately three months. During this time over 400,000 connections were detected and analyzed, and among these connections, over 400 were identified as being associated with intrusive behavior.

Our future work includes continual improvement of the isolated behavior-detection function for connections as well as other objects in the model (i.e. service-sets, hosts, and streams). We would like to take advantage of semantic knowledge about known system vulnerabilities, and we would also like to develop profiles of intrusive activity as well as normal activity.

As mentioned previously, we are also moving towards automatic code generation for the object detector and object analyzer components of the monitor. We are currently developing a system language definition for a stand alone host based monitor too, and if we can develop automatic code generators for object detector and analyzer modules, then porting the monitor to a different operating system should be greatly simplified.

Finally, we are incorporating our network monitor into a distributed intrusion detection system called DIDS [15]. DIDS combines both host based as well as network based monitors to take advantage of the benefits of both systems.

References

1. G.V. Bochmann, "Semantic Evaluation from Left to Right," *Communications of the ACM*, vol. 19, no. 2, pp. 55-62, Feb. 1976.
2. D.E. Denning, "An Intrusion Detection Model," *IEEE Trans. on Software Engineering*, vol. SE-13, no. 2, pp. 222-232, Feb. 1987.
3. P.J Denning, ed. Computers Under Attack: Intruders, Worms, and Viruses. New York: ACM Press, 1990.
4. Department of Defense Trusted Computer System Evaluation Criteria, Dept. of Defense, National Computer Security Center, DOD 5200.28-STD, Dec. 1985.
5. G.V. Dias, K.N. Levitt, B. Mukherjee., "Modeling Attacks on Computer Systems: Evaluating Vulnerabilities and Forming a Basis for Attack Detection," Technical Report CSE-90-41, University of California, Davis.
6. C. Dowell and P. Ramstedt, "The COMPUTERWATCH Data Reduction Tool," *Proc. 13th National Computer Security Conference*, pp. 99-108, Washington, D.C., Oct 1990.
7. L.T. Heberlein, "Towards Detecting Intrusions in a networked Environment," Technical Report CSE-91-23, University of California, Davis.
8. L.T. Heberlein, B. Mukherjee, K.N. Levitt, D. Mansur., "Towards Detecting Intrusions in a Networked Environment," *Proc. 14th Department of Energy Computer Security Group Conference*, May 1991.
9. L.T. Heberlein, G.V. Dias, K.N. Levitt, B. Mukherjee, J. Wood., "Network Attacks and an Ethernet-based Network Security Monitor," *Proc. 13th Department of Energy Computer Security Group Conference*, pp. 14.1-14.13, May 1990.
10. L.T. Heberlein, G.V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, D. Wolber., "A Network Security Monitor," *Proc. 1990 Symposium on Research in Security and Privacy*, pp. 296-304, May 1990.
11. B.W. Kernigan, D.M. Ritchie., The C Programming Language, 2nd ed. Englewood Cliffs, New Jersey: Prentice Hall, 1988.
12. D.E. Knuth, "Semantics of Context-Free Languages," *Math Systems Th.* 2 (1968), 127-145. Correction appears in *Math Systems Th.* 5 (1971),95.
13. T.F. Lunt, et al., "A Real Time Intrusion Detection Expert System (IDES)," Interim Progress Report, Project 6784, SRI International, May 1990.
14. S.E. Smaha, "Haystack: An Intrusion Detection System," *Proc. IEEE Fourth Aerospace Computer Security Applications Conference*, Orlando, FL, Dec. 1988.
15. S.R. Snapp, J. Brentano, G.V. Dias, T.L. Goan, L.T. Heberlein, C. Ho, K.N. Levitt, B. Mukherjee, S.E. Smaha, T. Grance, D.M. Teal, D.L. Mansur., "DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and an Early Prototype," to be published in *Proc. 14th National Computer Security Conference*, Oct. 1991.
16. W.T. Tener, "Discovery: an expert system in the commercial data security environment," *Security and Protection in Informations Systems: Proc. Fourth IFIO TC11 International Conference on Computer Security*, North-Holland, May 1988.
17. H.S. Vaccaro and G.E. Liepins, "Detection of Anomalous Computer Session Activity," *Proc. 1990 Symposium on Research in Security and Privacy*, pp. 280-289, Oakland, CA, May 1989.
18. J.R. Winkler, "A Unix Prototype for Intrusion and Anomaly detection in Secure Networks," *Proc. 13th National Computer Security Conference*, pp. 115-124, Washington, D.C., Oct. 1990.

MODEL-BASED INTRUSION DETECTION

Thomas D. Garvey*
Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

Teresa F. Lunt
Computer Science Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

Abstract

This paper introduces model-based reasoning and discusses how model-based reasoning capabilities can be applied to intrusion detection. We discuss the benefits of the approach and have shown its advantages over those currently in use. The use of model-based reasoning technology allows intrusion models to be specified much more easily and naturally than is the case using other technologies. Most importantly, the use of model-based reasoning technology will allow IDES to be a much better detector of intrusions.

1 Introduction

Timely detection of unauthorized intruders into computers and computer networks is a problem of increasing concern. Intruders might be characterized as “joy riders” with no malicious intent, as thieves aiming to appropriate resources of the computer system or those controlled by the system, or as terrorists aiming to destroy or incapacitate the system. Intruders often use specific, known procedures to breach a system’s security. Examples include programmed password attacks, access to privileged files, or exploitation of known system vulnerabilities.

IDES is an intrusion detection system built on the concept of detecting anomalous behavior of users with respect to observed behavioral norms. This approach may be likened to an unsupervised learning scheme for behavioral patterns with a subsequent pattern recognition approach to determining whether observed behavior falls inside or outside the pattern. In effect, a model of a user’s behavior is generated based on observations, but it is difficult to relate the model to specific (and specifically proscribed) activities. Thus, validation of the behavior of IDES’ statistical algorithms may prove to be difficult.

IDES also includes an expert system component that attempts to encode known system vulnerabilities and attack scenarios in its rule base. IDES raises an alarm if observed activity matches any of its encoded rules. However, expert system technology provides no support for developing models of intrusive behavior and encourages the development of *ad hoc* rules.

Here, we discuss how we are extending the IDES paradigm to include specific models of proscribed activities. These models would imply certain activities with certain observables which could then be monitored. This would allow us to actively search for intruders by looking for activities which would be consistent with a hypothesized intrusion scenario. A determination of the likelihood

*copyright 1991 Thomas D. Garvey and Teresa F. Lunt

of a hypothesized intrusion would be made based on the combination of evidence for and against it. The security properties of such an explicit model should be easier to validate.

The primary objectives of this work are to enhance the IDIES intrusion-detection system to include top-down, model based intrusion detection as one of its capabilities. We expect that intrusion scenarios will vary for different types of intruders and for different systems. Here, we are specifically interested in representing models for intrusion into systems such as commercial banking and financial systems, military computer networks, and systems for controlling communication and power distribution networks.

1.1 Background

Existing security mechanisms protect computers and networks from unauthorized use through access controls, such as passwords. However, if these access controls are compromised or can be bypassed, an abuser may gain unauthorized access and thus can cause great damage and disruption to system operation. Most computer systems have security susceptibilities that leave them vulnerable to attack and abuse. It is plain from numerous newspaper accounts of break-ins and computerized thefts that access control mechanisms cannot be relied upon in most cases to safeguard against a penetration or insider attack. Even the most secure systems are vulnerable to abuse by insiders who misuse their privileges. Audit trails can establish accountability of users for their actions, and have been viewed as the final defense, not only because of their deterrent value, but because in theory they can be perused for suspicious events and then to provide evidence to establish the guilt or innocence of suspected individuals. Moreover, audit trails may be the only means of detecting authorized but abusive user activity.

One of the key problems in detecting intrusions is that huge amounts of data are collected and must be sorted through. This data is not necessarily relevant to detecting intrusions, and may omit many items that would be of interest for intrusion detection. Furthermore, single events in the audit trail may not themselves be indicators of an attempted or successful intrusion, but their interrelationships with other events may be important indicators. Also, such audit trails may omit information that is relevant to detecting intrusions. These factors make it difficult to analyze audit trails for possible security breaches using conventional techniques. What is needed is a basis for understanding which data out of the huge volume of data available should be examined. This would allow one to maximize the utility of the data collected while minimizing the extraneous information.

The earliest work on intrusion detection [1] categorized the threats that could be addressed by audit trail analysis as external penetrators (who are not authorized the use of the computer); internal penetrators (who are authorized use of the computer but are not authorized for the data, program, or resource accessed), including masqueraders (who operate under another user's id and password) and clandestine users (who evade auditing and access controls); and misfeasors (authorized users of the computer and resources accessed who misuse their privileges). This study suggested that external penetrators can be detected by auditing failed login attempts; that some would-be internal penetrators can be detected by observing failed access attempts to files, programs, and other resources; and that masqueraders can be detected by observing departures from established patterns of use for individual users. Nothing was offered for detecting clandestine users and the legitimate user who abuses his or her privileges. In the decade since that first study was published, several research groups have built prototype intrusion-detection systems using these recommendations, but little or no further guidance has emerged on how to recognize intrusive behavior, beyond these simple guidelines.

Subsequent early work focused on developing procedures and algorithms for automating the offline security analysis of audit trails. One such project used existing audit trails and studied

possible approaches for building automated tools for their security analysis [2]. Another such project considered building special security audit trails and studied possible approaches for their automated analysis [3]. These projects provided the first experimental evidence that users could be distinguished from one another based on their patterns of usage of the computer system [2], and that user behavior characteristics could be found that were capable of discriminating between normal user behavior and a variety of simulated intrusions [3].

Based on this early evidence, work was begun on a real-time intrusion-detection system, that is, a system that would continuously monitor user behavior and be capable of detecting suspicious behavior as it occurs. This system, called IDES (Intrusion-Detection Expert System), takes the approach that intrusions, whether successful or attempted, could be detected by flagging departures from historically established norms of behavior for individual users [4, 5, 6].

SRI's real-time intrusion-detection expert system (IDES) is an independent system processes audit data characterizing user activity received from a target system. Its goal is to provide a system-independent mechanism for real-time detection of security violations. IDES is independent of any particular target system, application environment, system vulnerability, or type of intrusion, thereby providing a framework for a general-purpose intrusion-detection system using real-time analysis of audit data.

IDES currently has two detection components. Its statistical component keeps statistical profiles of past user behavior, and compares current behavior with historical behavior to determine whether the current behavior is anomalous. IDES' expert system component contains rules that characterize types of intrusions, system vulnerabilities, and security policies, and raises an alarm if observed activity matches any of its encoded rules.

The IDES prototype is currently running at SRI and monitoring an internal Sun network there. A version of IDES is also installed and working with live data at the FBI.

There are obvious difficulties with attempting to detect intrusions solely on the basis of departures from observed norms for individual users. Although some users may have well-established patterns of behavior, logging on and off at close to the same times every day and having a characteristic level and type of activity, others may have erratic work hours, may differ radically from day to day in the amount and type of their activity, and may use the computer in several different locations and even time zones (in the office, at home, and on travel). Thus, for the latter type of user, almost anything is "normal," and a masquerader might easily go undetected. Thus, the ability to discriminate between a user's normal behavior and suspicious behavior depends on how widely that user's behavior fluctuates and on the range of "normal" behavior encompassed by that user. And although this approach might be successful for penetrators and masqueraders, it may not have the same success with legitimate users who abuse their privileges, especially if such abuse is "normal" for those users. Moreover, the approach is vulnerable to defeat by an insider who knows that his or her behavior is being compared with his or her previously established behavior pattern and who slowly varies their behavior over time, until they have established a new behavior pattern within which they can safely mount an attack.

Because the task of discriminating between normal and intrusive behavior is so difficult, another study has taken the straightforward approach of automating the security officer's job. Such an approach lends itself to traditional expert system technology, in which the special knowledge of the "experts" in intrusion-detection, namely the system security officers, is codified as rules used to analyze the audit data for suspicious activity. The obvious drawback to this approach is that the security officers, in practice, have obtained only limited expertise because of the large amount of audit data produced and the tedium and length of time required to perform their checks. Thus, while automating these rules provides the useful function of freeing up the security officer to perform further analysis than they would otherwise have been capable of, such rules cannot be expected to

be comprehensive. This approach would be more aptly called a security officer's assistant.

Several intrusion-detection systems, including IDES, also include a rule-based system containing rules designed to describe known system vulnerabilities and reported attack scenarios, as well as intuition about suspicious behavior [7, 8], and some intrusion-detection systems rely exclusively on such expert systems. The rules are fixed in that they do not depend on past user or system behavior. An example of such a rule might be that more than three consecutive unsuccessful login attempts for the same userid within five minutes is a penetration attempt. Audit data from the monitored system is matched against these rules to determine whether the behavior is suspicious.

2 Model-Based Reasoning for Intrusion Detection

We have recently embarked on a study to explore the application of model-based reasoning technology to intrusion-detection. The eventual result will be an additional intrusion-detection component for IDES. This component will enable IDES to go beyond what any existing intrusion-detection system is capable of.

The model-based reasoning approach extends the IDES paradigm to include specific models of proscribed activities. These models imply certain activities with certain observables which could then be monitored. This will allow IDES to actively search for intruders by looking for activities which would be consistent with a hypothesized intrusion. A determination of the likelihood of a hypothesized intrusion is made based on the combination of evidence for and against it. The intrusion scenarios are expected to vary for different types of intruders and for different systems.

Figure 1 shows how model-based reasoning can be used for intrusion detection. The box labeled "scenario models" represents a knowledge base containing specifications of various scenarios or models of intrusion. These models are specified in terms of the sequences of user behavior that constitute the scenario. For example, one scenario could represent a programmed password attack. This scenario would contain the steps needed to carry out the attack, expressed in terms of the specific user behavior involved (and not in terms of the audit data).

The box labeled "active models" includes those models for which the system has discovered some evidence for their occurrence. The system is currently seeking additional evidence to confirm or refute these models. As evidence is discovered that would support one of the other scenario models, that model would be added to the active set. For example, the system may have hypothesized that user *A* is carrying out a programmed password attack, because user *A* was observed to have scanned the directory in which the password file resides.

The box labeled "anticipator" represents the part of the system that uses the active models to hypothesize the next step in the scenario that is expected to occur. For example, the hypothesized next step might be that user *A* will copy the password file. The "planner" then translates this hypothesized behavior into the specific attributes and values of the audit data that would indicate that behavior. In other words, the planner figures out how the hypothesized behavior would show up in the audit data. To do the translation, the planner uses a database of tables or matrices that map aspects of user behavior to particular elements and values in the audit data, indicated by the box labeled "behavior/data mapping." For example, the hypothesis that user *A* will copy the password file might be translated into the following things to look for in the audit data: user *A* uses the 'copy' command, user *A* opens the password file, and user *A* writes a new file.

This mapping of aspects of user behavior to how the behavior will show up in the audit data must exhibit properties that differentiate the particular behavior of concern from everything else that might be occurring. These distinguishing properties must have the following characteristics.

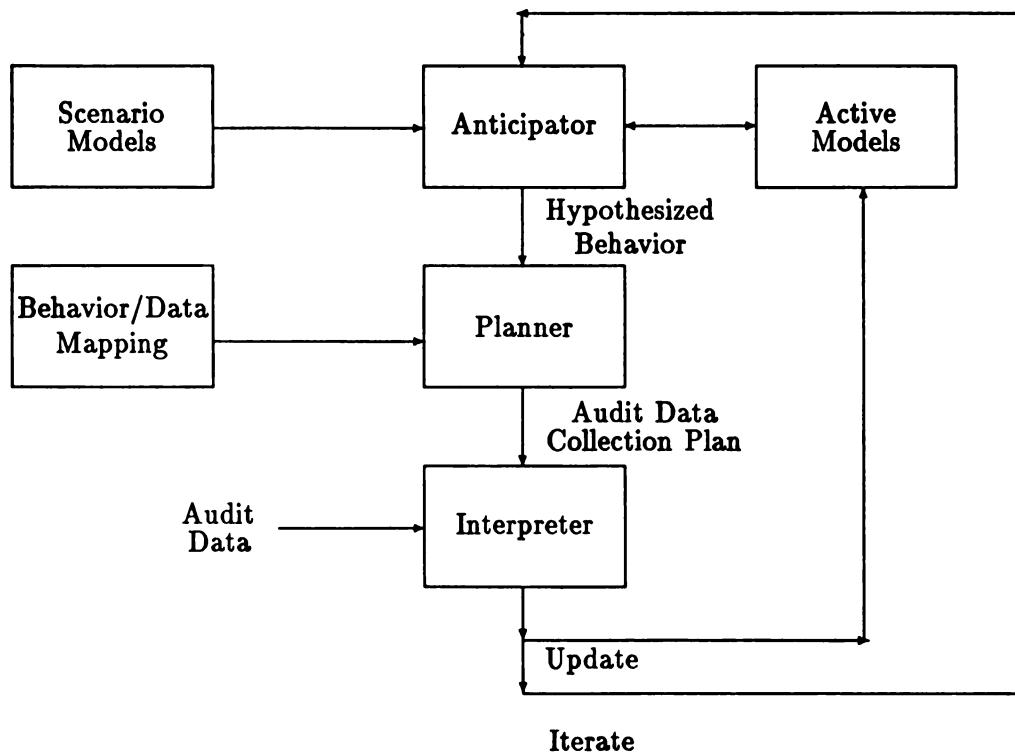


Figure 1: Model-Based Reasoning Approach

- They must be easily recognized, so that they can be readily detected.
- They must be clearly associated with the behavior in question. These are called *critical features*, because they always occur in the behavior you are looking for.
- They must not be associated with other ‘normal’ behavior. These are called *distinguishing features*, because they generally do not occur in behavior that is considered normal.

Thus, in addition to the descriptions of how the intrusive behavior will show up in the audit data, there also must be included descriptions of other, or normal, behavior. However, normal behavior may be defined simply as anything other than the particular behavior the system is looking for. In this case, the models of intrusion must be specified so as to include only aspects of behavior not exhibited unless the intrusion scenario is being enacted.

The planner then uses this information, that is, the particular items in the audit trail that are indicative of the behavior in question, to develop a plan for the specific audit data to examine next.

Next the “interpreter” compares the values in the plan to the actual values of the data observed, in an attempt to confirm or refute the hypothesized scenario. The results are used to update the active models, and then the process begins again with the anticipator. This process progresses until enough evidence is obtained to put the likelihood for a particular intrusion scenario over some predetermined threshold. At this point, the system announces that a potential intrusion has been detected.

We plan to examine the feasibility of using SRI’s Gister¹ evidential reasoning system for the fusion and interpretation of evidence for hypothesized intrusions. This process takes place in the

¹Gister is a trademark of SRI International [11].

interpreter described above. In Section 3, we describe how this fusion and interpretation of evidence is done, and we include an example in Section 3.2 to help the reader understand the steps involved. We will also develop a specification for a model-based intrusion detection capability, based on Gister, for inclusion within IDES. Previous applications of Gister include intelligence processing, military situation assessment, medical diagnosis, and acoustic and electronic signal processing.

2.1 Benefits of Model-Based Reasoning

The benefits of using model-based reasoning technology in intrusion detection applications are manifold, including the following.

- Much more data can be processed, because the technology allows you to selectively narrow the focus of the relevant data. Thus, at any given time, only a small part of the data collected need be examined.
- More intuitive explanations of what is being detected can be generated, because the events flagged can be related to the defined intrusion scenarios.
- The system can predict what the intruder's next action will be, based on the defined intrusion models. Such predictions can be used to verify an intrusion hypothesis, to take preventive action, or to determine which data to look for next.

In this section, we discuss these benefits.

A tremendous amount of audit data is generated in the monitored computer systems, and this enormous amount of data collected contains relatively little real information. With the model-based reasoning approach, the models of intrusion scenarios allow the intrusion-detection system to focus its attention on the data likely to be of most utility at the moment. The models can be used to examine only the data most relevant to detecting intrusions. In effect, we can narrow the field of view to optimize the data that has to be analyzed. This is analogous to pointing and tuning a sensor to optimize performance.

If the stream of audit data contains a significant number of intrusions in comparison with the total volume of audit data (i.e., there is a large signal-to-noise ratio), then an approach in which all the incoming data is examined and analyzed can be successful. However, if the number of intrusions is very small in comparison with the total volume of audit data (a small signal-to-noise ratio), then the amount of data to be examined can quickly overwhelm the intrusion-detection system. The system will be drawing very many conclusions, most of which will be dead ends. In this case, a more efficient approach would be to examine only the specific data in the audit data stream that are relevant at the moment. Thus, we can, in effect, increase the signal-to-noise ratio in particular areas by looking only in those areas. This top-down approach to data analysis will be more efficient in the intrusion-detection domain, where the signal-to-noise ratio is extremely small.

With the top-down model-based reasoning approach, the models of intrusion can be used to decide what specific data should be examined next. These models allow the system to predict the action an intruder would take who is following a particular scenario. This in turn allows the system to determine specifically which audit data to be concerned with. If the relevant data does not occur in the audit trail, then the scenario under consideration is probably not occurring. If the system does detect what it was looking for, then it predicts the next step and will then examine only data specifically relevant to confirming the hypothesis of the posited intrusion, and so on until a conclusion is reached. Thus, a model-based system reacts to the situation, using only that data most appropriate to the given situation and context.

In contrast with this approach, in which a set of intrusion models allows you to look for only a few things at any given point in time, an expert system's rules are always being used and evaluated against all the incoming audit data.

As is the case with expert systems, the approach is limited, in that it looks for known intrusion scenarios, whereas the greatest threat may be unknown vulnerabilities and the attacks that have not yet been tried. IDES' statistical intrusion-detection component takes a more global approach. Thus, the model-based and statistical approaches are each strong where the other is weak. The combination of a statistical with a model-based approach would allow IDES to benefit from the strengths of each.

A model-based component in IDES could also make use of the information generated by the statistical component, because the statistical anomalies detected could be used as evidence by the model-based component. Moreover, the model-based component could be used to adaptively add or delete rules in the expert system rule base, as the situation requires.

2.2 Comparison with Expert Systems

Although an expert system can also be used to build models of intrusions, the model-based reasoning technology allows these models to be specified much more easily and directly. The technology allows one to specify intrusion scenarios, and then the intrusion-detection system can generate the specific rules needed for identifying supporting evidence for these scenarios from the audit data.

With the model-based reasoning technology, the models of intrusion can be modified by a security officer much more easily than can expert system rules. This is because with the model-based reasoning technology, the models can be constructed using a graphical menu-and-mouse interface that clearly shows how the information is interrelated. The user does not have to deal with the knowledge base in text form. Thus, maintaining the knowledge base does not require as much care as when maintaining an expert system rule base. This is because the interrelationships among the various model components can be displayed visually, so that it is evident to the user what the effects of any given modification will be. In contrast, in expert systems, the interrelationships among rules are not represented or even defined. Thus, it is difficult for the user to predict the overall change in behavior of the system that will result from any particular rule modification.

A model-based reasoning system is better at detecting intrusions than is an expert system. The models can more accurately represent the undesirable behavior for which evidence is being looked for in the audit data. This is because the models can be expressed naturally in terms of the sequences of events that define the intrusion scenarios. By contrast, in an expert system, the rules are generally specified in the language of the audit data. With a model-based reasoning system, it is not necessary to identify the distinguishing features of the model, as you do with rules, because these can be determined by the system itself.

Model-based reasoning supports a sound theory for reasoning under uncertainty. This technology allows uncertainty in the rules — whether the behavior implies something illegitimate — and uncertainty in the significance of the data. Such a capability cannot easily be added to an expert system. And although some rule-based expert systems allow the handling of approximate information, they are based on an *ad hoc* theory, so that it is difficult to know what the results mean.

In the next section, we give an overview of evidential reasoning, which is the theoretical foundation for the model-based reasoning technology we have been discussing.

3 Evidential Reasoning

A key problem in intrusion detection is the interpretation of audit data whose relationship to the intrusive behavior you are looking for may be uncertain. A requirement, then, is to be able to reason about the likelihood of an intrusion scenario, given evidence in the form of audit data. Evidential reasoning provides a methodology for this type of reasoning.

3.1 Overview of Evidential Reasoning

The goal of developing knowledge-based systems that can reason with information that is uncertain or inexact in one way or another has long been a part of artificial intelligence research. Several technologies have been proposed for representing knowledge and deriving consequences from imperfect data: MYCIN's certainty factors [15], Prospector's inference nets [13], fuzzy sets [16], Bayesian nets [12], and Dempster-Shafer belief functions [10] are prominent examples.

The theory of belief functions, as originally conceived by Dempster [9] and further developed by Shafer [14], has received considerable attention as a basis for representing uncertainty within expert systems. The theory is a generalization of classical probability theory and provides a representation of degrees of precision as well as degrees of uncertainty. Its ability to express partial ignorance is of great value in the design of knowledge-based systems for real-world domains.

Currently, one of the most highly developed knowledge-based systems that incorporates Shafer's theory of belief functions for a wide range of application domains is Gister [11]. In this section we give a brief review of the evidential reasoning technology employed by Gister.

The goal of evidential reasoning is to assess the effect of all available pieces of evidence upon a hypothesis, by making use of domain-specific knowledge. The first step in applying evidential reasoning to a given problem is to delimit a propositional space of possible situations. Within the theory of belief functions, this propositional space is called the *frame of discernment*. A frame of discernment delimits a set of possible situations, exactly one of which is true at any one time. Once a frame of discernment has been established, propositional statements can be represented by subsets of elements from the frame corresponding to those situations for which the statements are true. Bodies of evidence are expressed as probabilistic opinions about the partial truth or falsity of propositional statements relative to a frame. Belief assigned to a nonatomic subset explicitly represents a lack of information sufficient to enable more precise distribution. This allows belief to be attributed to statements whose granularity is appropriate to the available evidence.

The distribution of a unit of belief over a frame of discernment is called a *mass distribution*. A mass distribution, m_Θ , is a mapping from subsets of a frame of discernment, Θ , into the unit interval:

$$m_\Theta : 2^\Theta \mapsto [0, 1],$$

such that

$$m_\Theta(\phi) = 0 \quad \text{and} \quad \sum_{X \subseteq \Theta} m_\Theta(X) = 1.$$

Any proposition that has been attributed nonzero mass is called a *focal element*. One of the ramifications of this representation of belief is that the belief in a hypothesis X is constrained to lie within an interval $[Spt(X), Pls(X)]$, where

$$Spt(X) = \sum_{Y \subseteq X} m_\Theta(Y) \quad \text{and} \quad Pls(X) = 1 - Spt(\bar{X}). \quad (1)$$

These bounds are commonly referred to as *support* and *plausibility*. A *body of evidence* (BOE) is represented by a mass distribution together with its frame of discernment. A BOE that directly

represents one of the available pieces of evidence is called *primitive*; all other BOEs are *conclusions* or intermediate conclusions.

In evidential reasoning, domain-specific knowledge is defined in terms of *compatibility relations* that relate one frame of discernment to another. A compatibility relation simply describes which elements from the two frames can simultaneously be true. A compatibility relation, $\Theta_{A,B}$ between two frames Θ_A and Θ_B is a set of pairs such that

$$\Theta_{A,B} \subseteq \Theta_A \times \Theta_B,$$

where every element of Θ_A and every element of Θ_B is included in at least one pair.

Evidential reasoning provides a number of formal operations for assessing evidence, including:

1. **Fusion** — to determine a consensus from several bodies of evidence obtained from independent sources. Fusion is accomplished through Dempster's rule of combination:

$$m_{\Theta}^3(A_h) = \frac{1}{1-k} \sum_{A_i \cap A_j = A_h} m_{\Theta}^1(A_i) m_{\Theta}^2(A_j), \quad (2)$$

$$k = \sum_{A_i \cap A_j = \emptyset} m_{\Theta}^1(A_i) m_{\Theta}^2(A_j).$$

Dempster's Rule is both commutative and associative (meaning evidence can be fused in any order) and has the effect of focusing belief on those propositions that are held in common.

2. **Translation** — to determine the impact of a body of evidence upon elements of a related frame of discernment. The *translation* of a BOE from frame Θ_A to frame Θ_B using the compatibility relation $\Theta_{A,B}$ is defined by:

$$m_{\Theta_B}(B_j) = \sum_{\substack{C_{A \mapsto B}(A_k) = B_j \\ A_k \subseteq \Theta_A, B_j \subseteq \Theta_B}} m_{\Theta_A}(A_k), \quad (3)$$

where $C_{A \mapsto B}(A_k) = \{b_j | (a_i, b_j) \in \Theta_{A,B}, a_i \in A_k\}$.

3. **Projection** — to determine the impact of a body of evidence at some future (or past) point in time. The *projection* operation is defined exactly as translation, where the frames are taken to be one time-unit apart.
4. **Discounting** — to adjust a body of evidence to account for the credibility of its source. Discounting is defined as

$$m_{\Theta}^{discounted}(A_j) = \begin{cases} \alpha \cdot m_{\Theta}(A_j), & A_j \neq \Theta \\ 1 - \alpha + \alpha \cdot m_{\Theta}(\Theta), & \text{otherwise} \end{cases} \quad (4)$$

where α is the assessed credibility of the original BOE ($0 \leq \alpha \leq 1$).

Several other evidential operations have been defined and are described elsewhere [11].

Independent opinions are expressed by multiple bodies of evidence. Dependent opinions can be represented either as a single body of evidence, or as a network structure that shows the interrelationships of several BOEs. The evidential reasoning approach focuses on a body of evidence, which describes a meaningful collection of interrelated beliefs, as the primitive representation. In contrast, all other such technologies focus on individual propositions.

3.2 The Analysis of Evidence

To illustrate the reasoning methods described above, we use the following example.

A user logs in from a remote host after trying several bad passwords and usernames. The user makes several errors in entering command names and arguments and tries to look at some directories and files for which permission is denied. The user also several times uses commands such as 'who' to find out about other system activity. After a few minutes, the user logs out. Was this an intruder?

In evidential reasoning the first step is to construct the sets of possibilities (the frames of discernment) of each unknown. For example, the user could either be an intruder or not:

{Yes, No}

Other frames could also be constructed; we would probably want one for user location

{Present, Remote}.

We distinguish two types of location for a user — present (i.e., physically at the keyboard) and remote. Because the majority of intruders do not have direct physical access to the target machine, a keyboard location is considered to indicate normal use and not an intruder. Most intrusions originate from remote internet sites. However, because an intruder can jump from host to host, intrusive behavior is also likely to appear from local hosts. Thus, activity originating from any location other than the keyboard is considered equally indicative of intrusive behavior, so we use only the single category 'remote' for this. For remote use, we cannot distinguish whether the user is an intruder based on this dimension of behavior alone.

We expect that an intruder may be somewhat paranoid and we will also want to include a frame to capture paranoia level

{Paranoid, Cool}.

A paranoid intruder (one who is afraid of being caught) will probably have very short sessions (lasting under two minutes), because the longer the session the greater the risk of discovery. A paranoid intruder will also commonly check to see who is logged in and what they are doing. Thus, for example, in Unix we can expect an inordinate number of 'who,' 'ps,' and 'finger' commands to indicate a paranoid intruder. We can characterize user sessions as having a high degree of this sort of activity two or more such commands are used. Thus, we consider short sessions and two or more "surveillance" commands to be strong indicators of paranoia.

An intruder may also be unfamiliar with the system, so we will include a frame for familiarity

{Familiar, Unfamiliar}.

A person who is unfamiliar with the computer system is likely to have a relatively large number of invalid commands, resulting from attempts to execute commands that are not recognized by the system. Such a person is also likely to have a relatively large number of errors resulting from invalid command usage, for example, from too few arguments or invalid parameters. A relatively large number of file permission errors, resulting from attempting to read, write, or execute files or directories when permission is denied, is also indicative of a person unfamiliar with the computer system. Thus, we consider relatively large numbers of errors of these several types to be strong indicators of unfamiliarity with the system. Conversely, low error rates for all of these categories of error strongly suggest a normal, nonintrusive, user.

Authentication errors result from the use of an invalid username or password during login. We consider a high rate of authentication errors (greater than three failed login attempts for a given username in one minute) to be strongly suggestive of an intrusion attempt.

The second step in evidential reasoning is to construct the compatibility relations that define the domain-specific relationships between the frames. A connection between two propositions A_1 and B_1 indicates that they may co-occur (in other words, $(A_1, B_1) \in \Theta_{A,B}$).

Figure 2 shows the frames and compatibility relations used in determining whether the user is an intruder.

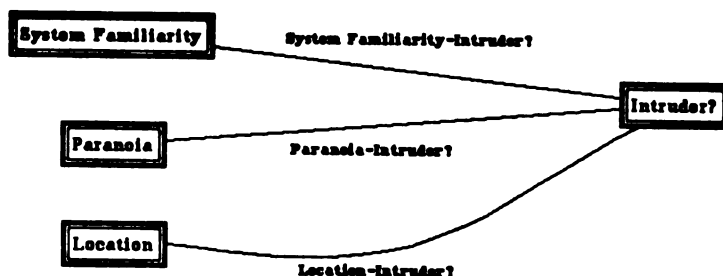


Figure 2: Frames and compatibility relations.

Once the frames and compatibility relations have been established, we can analyze the evidence. The goal of the analysis is to establish a line of reasoning from the evidence to determine belief in a hypothesis, in this case that the user is an intruder. Figure 3 shows the analysis within the Gister framework.

The first step is to assess each piece of evidence relative to an appropriate frame of discernment. Each piece of evidence is represented as a mass distribution, which distributes a unit of belief over subsets of the frame. For example, the fact that the user logged in from a remote host is pertinent to the *Location* frame, and we attribute 1.0 to *Remote* to indicate our complete certainty on this point.

The fact that the user had a high number of authentication errors leads us to believe that the user may be an intruder. Based on this, we assign a likelihood of 0.75 to the possibility that the user is an intruder.

The high number of command usage and file permission errors gives information about *Familiarity*. Based on the number and types of errors, we assign a belief of 0.7 to the possibility, *Unfamiliar*; the remaining 0.3 is assigned to *Familiar*.

The last piece of evidence, that the user used several “surveillance” commands and had a short session, give information about *Paranoia* and, might be assessed as giving 0.75 support that the user is paranoid and 0.25 that the user is *Cool* and that this is usual behavior for that user (perhaps the user is a system administrator).

Evidence from these sources will provide the inputs to our analysis and are depicted in Figure 3. Many of these determinations are judgments that may not be of equal validity. In order to be able to weight them differently, we will provide a means for discounting the impact of the evidence through the discounting operation. This will allow us to change their relative weights.

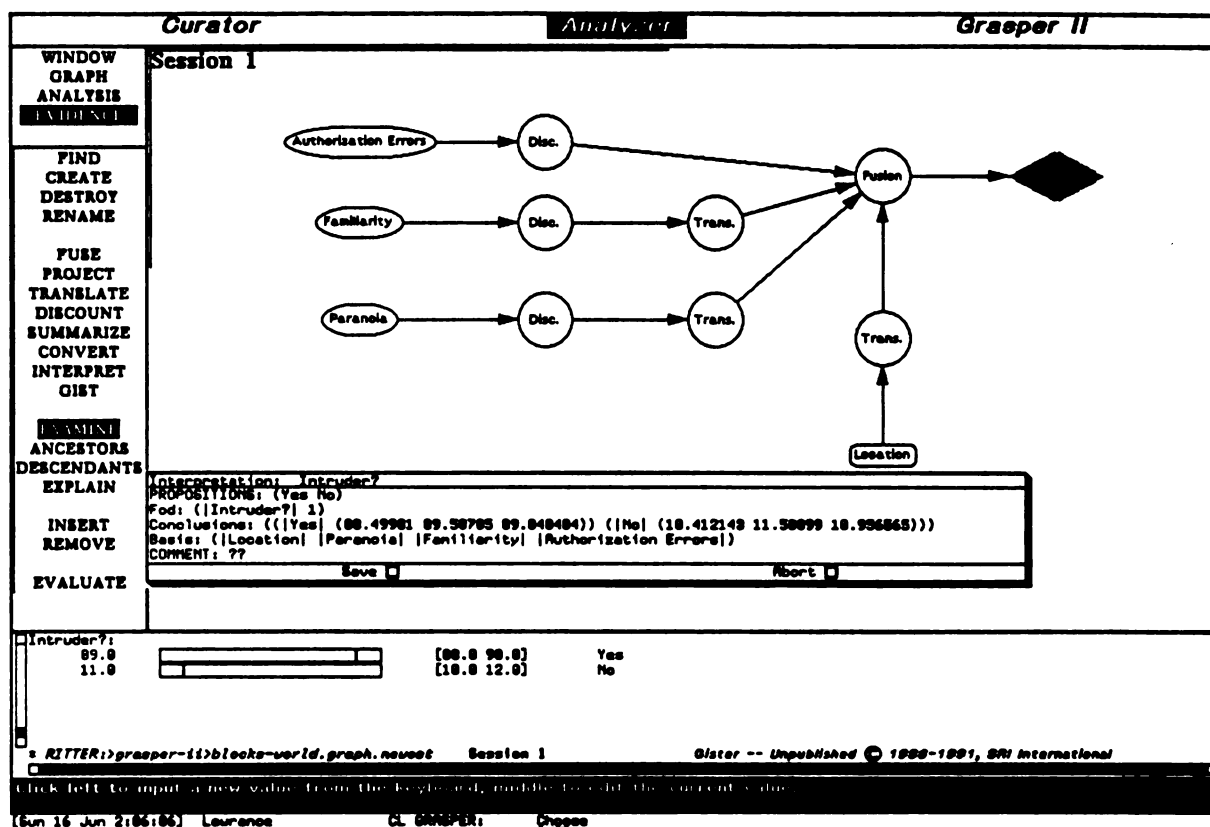


Figure 3: An intrusion analysis within Gister.

Our final step is to construct the actual analysis of the evidence as shown in Figure 3 to determine its impact upon the question at hand. In this case the question of whether our user is an intruder can be answered by an assessment of belief over elements in the *Intruder?* frame. Evidential operations are used to derive a body of evidence providing beliefs about whether the user is an intruder.

In the analysis in Figure 3, all sources except the *Location* source are discounted. The *AuthorizationErrors* source is already providing information about the likelihood of an intruder, but the others must all be translated to the *Intruder?* frame. These independent BOEs are now represented relative to a common frame and can be combined using the *fusion* operation (i.e., Dempster's Rule). Fusing the mass distributions yields a mass distribution relative to the *Intruder?* frame, from which conclusions as to whether the user is an intruder can be drawn.

Specifically,

$$m_{Intruder?}(x) = \begin{cases} 0.88, & x = \{Yes\} \\ 0.10, & x = \{No\} \end{cases}$$

We use an interpretation node to assess the support and plausibility for the answers *Yes* and *No* to the question of whether the user is an intruder. The associated evidential intervals for the atomic propositions in this mass distribution (shown in the lower window pane of Figure 3) are:

$$\begin{aligned} [Spt(\{Yes\}), Pls(\{Yes\})] &= [0.88, 0.90] \\ [Spt(\{No\}), Pls(\{No\})] &= [0.10, 0.12] \end{aligned}$$

The hypothesis $\{Yes\}$ is clearly the most likely, and we conclude that the user is an intruder.

All the operations discussed above have been implemented within Gister. Frames and compatibility relations are represented as graphs, which can be constructed, examined, and modified interactively. Having an automated means to compute a conclusion is necessary.

The completed analysis graph can be seen to be the counterpart of the proof tree of logical deduction. Each node represents an opinion, and the arcs trace the derivation of one opinion from other opinions and the knowledge contained in the compatibility relations. The complete graph shows the derivation of an ultimate conclusion from the primitive bodies of evidence.

The use of evidential reasoning provides a richer vocabulary for expressing belief about uncertain events than is available in most other technologies.

4 Future Work

In future work, we plan to acquire various intrusion scenarios from law enforcement agencies and elsewhere and to represent these scenarios as models within SRI's Gister system. Gister provides capabilities for fusion and interpretation of evidence from audit trails and statistical profiles in order to determine the likelihood that specific hypothesized intrusion scenarios are being enacted. Finally, we plan to specify a model-based intrusion detection capability for inclusion within IDES.

We believe that the importance of this area will continue to increase as more and more key systems become vulnerable to disruption or destruction by unauthorized intruders.

5 Summary and Conclusions

We have described model based reasoning and discussed how it can be applied to the intrusion detection domain. We have discussed the benefits of the approach and have shown its advantages over those currently in use, in particular expert systems. Finally, we identify our plan for incorporating this technology into the IDES intrusion-detection system. Using model-based reasoning technology will allow us to process much more audit data, because only the data most relevant to the current context need be examined. The technology will allow for more intuitive explanations of what is being detected can be generated, because the events flagged can be related to the defined intrusion scenarios. The technology also allows intrusion models to be specified much more easily and naturally than is the case using other technologies. Most importantly, the use of model-based reasoning technology will allow IDES to be a much better detector of intrusions.

References

- [1] J. P. Anderson. *Computer Security Threat Monitoring and Surveillance*. Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, April 1980.
- [2] H. S. Javitz, A. Valdes, D. E. Denning, and P. G. Neumann. *Analytical Techniques Development for a Statistical Intrusion Detection System (SIDS) based on Accounting Records*. Technical report, SRI International, Menlo Park, California, July 1986. not available for distribution.
- [3] J. van Horne and L. Halme. *Analysis of Computer System Audit Trails — Final Report*. Technical Report TR-85007, Sytek, Mountain View, California, May 1986.
- [4] T. F. Lunt. IDES: An intelligent system for detecting intruders. In *Proceedings of the Symposium: Computer Security, Threat and Countermeasures, Rome, Italy*, November 1990.
- [5] T. F. Lunt, Ann Tamaru, Fred Gilham, R. Jagannathan, Caveh Jalali, H. S. Javitz, A. Valdes, and P. G. Neumann. *A Real-Time Intrusion-Detection Expert System*. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, 1990.

- [6] T. F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. G. Neumann, and C. Jalali. IDES: A progress report. In *Proceedings of the Sixth Annual Computer Security Applications Conference*, December 1990.
- [7] T. F. Lunt, R. Jagannathan, R. Lee, A. Whitehurst, and S. Listgarten. Knowledge-based intrusion detection. In *Proceedings of the 1989 AI Systems in Government Conference*, March 1989.
- [8] R. A. Whitehurst. *Expert Systems in Intrusion-Detection: A Case Study*. Computer Science Laboratory, SRI International, Menlo Park, CA, November 1987.
- [9] Dempster, Arthur P., "A Generalization of Bayesian Inference," *Journal of the Royal Statistical Society* 30(Series B), 1968, pp. 205-247.
- [10] Lowrance, John D., and Garvey, Thomas D., "Evidential Reasoning: A Developing Concept," *Proceedings of the IEEE International Conference on Cybernetics and Society*, October 1982, pp. 6-9.
- [11] Lowrance, John D., Garvey, Thomas D., and Strat, Thomas M., "A Framework for Evidential-Reasoning Systems," *Proceedings AAAI-86*, Philadelphia, Pennsylvania, August 1986.
- [12] Pearl, Judea, "Fusion, Propagation, and Structuring in Bayesian Networks," Tech. Report CSD-850022, Cognitive Systems Laboratory, Computer Science Department, University of California, Los Angeles, June 1985.
- [13] Reboh, Rene, "Knowledge Engineering Techniques and Tools in the Prospector Environment," Technical Note 243, Artificial Intelligence Center, SRI International, Menlo Park, California, June 1981.
- [14] Shafer, Glenn A., *A Mathematical Theory of Evidence*, Princeton University Press, New Jersey, 1976.
- [15] Shortliffe, Edward H., *Computer-Based Medical Consultations: MYCIN*, American Elsevier, New York, 1976.
- [16] Zadeh, Lotfi A., "Fuzzy Sets as a Basis for a Theory of Possibility," *Fuzzy Sets and Systems*, Vol. 1, 1978, pp. 3-28.

NOTIFICATION : A PRATICAL SECURITY PROBLEM IN DISTRIBUTED SYSTEMS

Vijay Varadharajan

Hewlett-Packard Laboratories,
Filton Road, Stoke Gifford, Bristol BS12 6QZ, U.K.

Abstract

This paper considers a practical problem that arises when considering scenarios in distributed computing environments. In a distributed object system, the dependency between objects is a common phenomenon. This in turn implies that changes occurring in an object's state will affect the behaviour of other objects. For correct operation of the system these changes need to be properly notified to the other cooperating objects. Such situations are very common in many office applications and we consider one such typical case, namely the producer-consumer scenario and examine its security implications. We consider a solution to this problem, referred to as the *Notification Problem*. The solution is based on the increasingly publicized Kerberos authentication system. Note that Kerberos forms part of the OSF's¹ Distributed Computing Environment (DCE). We then consider the trust implications of the proposed solution which leads us to the more general problem of proxy or delegation in distributed systems. We conclude the paper by proposing an extension of the Kerberos system to handle proxy situations.

¹Open Software Foundation

1 Introduction

Several models of distributed systems are based on systems of objects, such as the ISO ODP model ([2]) and the CCITT DAF model ([1]). Although it is virtually impossible to get total agreement as to what constitutes an object system (look at the various systems proposed in [7]), the intrinsic concepts are very similar. We shall use the terms 'object system' and 'distributed system' interchangeably, to mean distributed object systems as in [8].

In distributed object systems we have a number of objects that exist independently of one another, but may use each others functionality to provide certain services. That is, although many objects can exist as fully functional entities in isolation, many other objects need the existence and functionality of other objects.

Suppose that one object is dependent on another for data so that it may perform some task, and that the data on which it relies is often changing. Thus the dependent object has to know when there has been a change, and access the other object to receive the updated data. The situation of dependency, notification of change, and access, is what we call the Notification problem. Such situations commonly arise in office system applications.

It is the dependency between objects, and the need for some objects to access other objects that is of interest from the security point of view. We need to establish how the dependency is set up, and what conditions need to be satisfied for a dependent object to access another object. The security issues are of access control, and of course authentication (which is a basic premise for access control). So we want to check whether an object can access another object, and that they are who they claim to be.

In this paper, we will restrict the discussion to the Kerberos authentication (and access control) system. Kerberos provides an authentication mechanism, based on a private key scheme, which was originally developed in Project Athena at MIT ([4]). It is being proposed as the underlying authentication mechanism in a number of distributed systems architectures and forms part of the OSF's Distributed Computing Environment (DCE). One of the nice characteristics of the Kerberos scheme is that it is transparent to the user. Initial secure communication is established using keys based on the users' passwords, but these initial keys are stored only long enough to provide a means of key distribution for session keys. However, Kerberos is not entirely suited to delegation. We will consider the Kerberos scheme in section 3.

The paper is organised as follows: section 2 describes in more detail the kinds of notification scenarios that occur in distributed object systems. In section 3 we describe the particular authentication and access control mechanism we are dealing with, namely Kerberos. This is followed by a section on how notification can be interpreted in Kerberos. This will involve declaring our assumptions, proposing our solution, and considering the trust implications. This in turn leads to the problem of delegation in distributed systems. We conclude the paper by outlining an extension of the Kerberos system to cope with this delegation (proxy) problem.

2 Notification Scenarios

In this section we will describe some scenarios where we have object dependencies in distributed systems. These scenarios are just examples of a more general scheme of dependencies where we have Producer and Consumer objects.

A typical situation which already exists in the PC office world is that of spreadsheets and charts (as in Microsoft Excel² and Lotus 1-2-3³). The spreadsheet consists of tables of information, where the contents of the table are called elements. Some elements in the table stand in a direct relation to other elements, such as being the sum of values in a column. A chart is a graphical representation of some section of the table, such as a bar chart. The chart is constructed from the selected section of the table. A "hot link" is established between the spreadsheet and the chart, so that if there is any alteration to the selected area of the spreadsheet, then this results in a change in the chart.

In this case, the spreadsheet is a Producer object, in that it "produces" data that is used by the chart. The chart is a Consumer object, in that it consumes the data of the Producer. The Producer can be thought of as an active object, whilst the Consumer is more of a passive object. Of course there could be many charts co-existing and using the same spreadsheet data. This can be generalised to say that for each Producer there can be more than one Consumer.

However, not every Consumer may want to know, or be allowed to know, of all the changes that may occur in the Producer. Consumers may only be allowed access to restricted parts of the Producer's data or output ports. The decision whether a Consumer is allowed access depends on the access control rules (the access control policy) of the distributed system.

Another example in distributed computing is that of electronic mail handlers. Each user in the distributed system can be considered to have an electronic mailbox, or in-tray. If mail arrives for them in the distributed system, then they may want to be immediately informed (in order to request their mail and maybe change the display of their mail icon). However, if they are not currently logged on, then they may wish to defer collection of their mail until the next time they logon. Here it is clear that the mailer is the Producer, and the mailbox the Consumer.

In many cases the Producer should not have to know about the Consumers. This could be handled by a Notification Server. The Notification Server is a register of all Consumers that have an interest (want to be notified of changes) in the Producer. Every time there is a change in the contents (data) of the Producer, the Producer informs the Notification Server (including information on the change), and it in turn informs all parties that have registered an interest. It is then up to the Consumers to directly request (interrogate) the Producer to obtain the any remaining (e.g. updated) information.

When a Consumer registers an interest in a Producer, it does so via the Notification Server. The Consumer must be authenticated (to establish that it is who it claims to be), and there must be a check to see whether this Consumer has the necessary rights for it to be informed of a change in the Producer, and also to request access to the Producer once it has been notified of change.

²Microsoft Excel is a trademark of Microsoft Corporation

³Lotus 1-2-3 is a trademark of Lotus Corporation

Once the Consumer has established an interest in the Producer, it may not need to go through the authentication process each time it wishes to access the Producer. However, it will need some form of secret known to itself and the Producer for authentication and to demonstrate that it has in fact been granted permission. This could take the form of a capability, or maybe some shared secret key, with freshness.

Notice that the Notification Server is not a critical *independent* component of the system, as it could be incorporated into the Producer. However, the function of the Notification Server is fundamental in the system, as it establishes the register of "legal" Consumers (according to the access control policy), which are all the entities that will be notified of changes to the Producer. It should now be clear why the Notification problem is so dependent on the authentication and access control mechanisms in use.

3 Kerberos

The Kerberos authentication scheme was developed by Project Athena at MIT. It is based on the client/server model of distributed computing, where clients access the resources of servers using remote procedure calls (RPC). One of the intentions of Kerberos is to make the authentication transparent to the users (Principals) of the system. So, each user does not have to decide whether it wishes to be authenticated or not by the use of its secret keys.

The Kerberos system has two basic components: the authentication server (AS), and a ticket granting server (TGS). The authentication server is used when a Client "logs in" (see 1 below). The AS communicates with the Client using a secret key known only to the AS and the Client. This key is generated from the Client's password using a one-way function, so that the Client does not have to provide any information other than its password (making authentication transparent). The AS supplies a key to the Client which is to be used for communication with the TGS, along with a Kerberos "ticket" for the TGS (see 2 below).

The Client requests from the TGS a ticket for a named Server, *s*. It also sends the ticket it received from the AS to the TGS, along with an authentication certificate (as in 3). The authentication certificate contains some information about the Client that is encrypted (signed) using the new key of the Client. The ticket contains the information on the Client, and also the secret key that is shared with the Client. The information in the ticket is encrypted (for both integrity and secrecy) using the secret key known only to the TGS and the AS, so that only the TGS can obtain the key from the ticket, and so it may use the key and the information about the Client to check the authentication certificate sent from the Client.

Once the TGS has authenticated the Client, it may provide a session key and token to be used by the Client for direct communication with the Server, encrypted using their shared key (see 4). Access control may be enforced at the TGS, in determining which Clients it may give tokens for the Server. Thus the TGS would effectively be generating capability tokens, that are accepted (without question) by the Server. However, access control may be enforced at the Server, using access control lists (ACLs). In this case the TGS is not discriminatory as to who it grants tickets for, as the ticket in itself does not determine rights, but can only be used for authentication.

So, the Client now has a ticket that it may present to the Server, and also a key (the session

key) which it uses to create an authentication certificate (so the Server can then authenticate the Client). In the same way the Client established itself with the TGS, so it does so with the Server, except that now uses the ticket for the Server (and not the ticket for the TGS), and also the session key for authenticating itself with the Server (and not the key from the AS to authenticate itself with the TGS). The similarity between messages 3 and 5 can easily be seen.

In the case where the Client requires authentication of the Server (mutual authentication), the Server sends some return message to the Client, signed using the session key, demonstrating that it has received the key, and that this returned message is indeed “fresh” and not a replay of an earlier response (as in 6).

1. Client \rightarrow AS : c, tgs
2. AS \rightarrow Client : $\langle K_{c,tgs}, \langle T_{c,tgs} \rangle_{K_{tgs}} \rangle_{K_c}$
3. Client \rightarrow TGS : s, $\langle T_{c,tgs} \rangle_{K_{tgs}}, \langle A_c \rangle_{K_{c,tgs}}$
4. TGS \rightarrow Client : $\langle K_{c,s}, \langle T_{c,s} \rangle_{K_s} \rangle_{K_{c,tgs}}$
5. Client \rightarrow Server : $\langle A_c \rangle_{K_{c,s}}, \langle T_{c,s} \rangle_{K_s}$
6. Server \rightarrow Client : $\langle \text{currenttime}+1 \rangle_{K_{c,s}}$

where:

$A_c = \langle \text{client-name, client-IP-addr, currenttime} \rangle$

$T_{c,tgs} = \langle \text{client-name, tgs-name, current-time, lifetime, client-IP-addr, } K_{c,tgs} \rangle$

$T_{c,s} = \langle \text{client-name, server-name, current-time, lifetime, client-IP-addr, } K_{c,s} \rangle$

4 Notification and Kerberos

In this section we consider some of the properties of Kerberos, taken as given components of the system. We look at how we may implement notification using the existing Kerberos system, and the problems that it presents. We then propose a solution to those problems. Finally we remark on the need for trusted entities in the system. Our aim is to use the existing Kerberos protocols and message formats as far as possible. Papers have been published recently (e.g. [6]) describing several weaknesses and limitations of Kerberos. In this paper, it is not our intention to consider these issues.

4.1 Solution

Let us start by first considering some of the properties of Kerberos.

- For a Principal (Client) to use a Server, it must first authenticate itself with the Kerberos Authentication Server. It then receives a token that it may take to a Ticket Granting Server (TGS) to receive a ticket for a particular server.
- A Principal need only present its password when it logs in. Subsequent authentication can be performed by presenting the token to the TGS. It may then receive a session key for communication with the server, along with a token to authenticate itself with the server.
- Therefore, for a Principal to gain access to a specific service, it need only have a valid token for that service (obtained from the TGS), and also the session key for communicating with the server.

Coming to the problem of Principal/Consumer/Producer, we can consider two scenarios :

In the first scenario, we have the following :

- The Producer is in Kerberos terms a Server. It is registered with Kerberos, and can be authenticated.
- The Consumer can be regarded as a Client. It is registered with Kerberos, and may obtain a ticket-granting ticket from the Kerberos Authentication Server, and further tickets for specific services from the TGS.

This is a direct mapping of Kerberos to the Producer-Consumer problem. Here, one could have the TGS play the role of the notification server informing the Consumers of any changes in the Producers' data. The Consumers need to be registered with the Authentication Server to begin with.

In the second scenario, we have

- The Producer is in Kerberos terms a Server. It is registered with Kerberos, and can be authenticated.
- The Principal that created the Consumer can be regarded as a Client. This may be a possibility when it is not feasible to register each of the Consumer as a Principal. In this case, the Principal is registered with Kerberos, and may obtain a ticket-granting ticket from the Kerberos Authentication Server, and further tickets for specific services from the TGS.

Note that in the second scenario, as the Consumer is not regarded as a Kerberos Principal, and therefore cannot in itself gain tickets for services, or be authenticated. If the Principal were to give this token for a service and the session key for that service to the Consumer (an object created by the Principal), then that Consumer may act on behalf of the Principal, but strictly for use of the specified server. The Consumer will be acting on behalf of the

Principal as long as the token for the service is valid. It cannot act on behalf of the Principal for any other services, or to communicate with the TGS or Authentication Server, as it has neither of the required keys for such communication.

With this second scenario, we need the following changes to Kerberos:

- The ticket from the TGS given to the Principal for a specific service should have a lifetime longer than the lifetime of the Principal (and set by the Principal). Currently it has a time which is the minimum of the remaining lifetime of Principal's ticket-granting ticket, and the remaining lifetime of the server. This is so that the Consumer may access the Producer after the Principal has logged off (but controlled by how long the Principal wants it to last).
- Tickets may remain after the Principal logs off. In particular the Principal should be able to define which tickets should exist. This does not affect the general security of the Principal, as the Consumer knows nothing of the Principal's password or the session key for the TGS.
- When a Consumer is requesting access to a Producer, the Producer can only recognize the Principal involved and not the Consumer. The Consumer will be given access only to the data that it is authorized for, which is dependent on the Principal on whose behalf the Consumer is acting. This is important because a Consumer may acquire authority to access Producers on behalf of several Principals.

4.2 Issues of Trust

Problems with the second solution above may occur if there is an untrusted medium between the Principal and the Consumer. There will be a need to protect the channel between the Principal and its Consumer. This problem could arise if the Consumer object "moves" to another machine or domain. Neither the Consumer nor the Principal/Consumer communication can be trusted.

The above problem suggests that there may be a need for a session key between the Principal and the Consumer that it owns. This presents a problem unless the Consumer can somehow register itself as a Principal. However, if the Consumer can register itself as a Principal, then this leads to our first scenario. We can therefore consider a solution in which the Consumer acts as any other Principal, in authenticating itself, communicating with the TGS, and gaining tokens for servers.

An important issue in these solutions is the need for a proxy feature where one object is authorised to act on behalf of another object. We consider this in the next section.

5 Proxy in Kerberos

We have considered the proxy or the delegation problem in distributed systems in detail in [9]. Kerberos (Vers.4) is not entirely suitable for handling such situations. We have just

received documentation on Kerberos (Vers.5) ([5]) which has some support for proxy. In this paper, we briefly outline an extension to Kerberos which can be used for managing proxy. In fact, the schemes we describe here are different from the one being considered in Kerberos Version 5. As stated earlier, our intention here is to use as far as possible the existing Kerberos framework and provide the extra delegation feature. Hence the delegations will suffer from the general weaknesses of the Kerberos system described elsewhere (e.g. in [6]).

For our purposes we shall consider the delegation to be between Clients (Kerberos Principals) for access to Servers. Thus we make the following observations:

1. We must assume the delegating client (originator) has been authenticated with the AS, and has also gained a valid ticket and session key for the particular server from the TGS (i.e. $\langle T_{a,s} \rangle_K$, and $K_{a,s}$).
2. As Principals do not maintain their secret keys (based on their passwords), they can only authenticate themselves using their shared keys with the TGS. Thus only authenticated objects can act as intermediaries.
3. The ticket and the session key for the server could be passed during proxy. If the key is passed it cannot be used as an authenticator by the server to authenticate each component in the chain of delegation (as it is shared by all objects in the path). Having the key no longer guarantees uniqueness of the Client/Server pair. Furthermore, unless the delegator and the delegate share a mutually secret key, then the session key must be transferred in plain. This is obviously undesirable, making this session key virtually worthless. An alternative would be to use the AS as a communications server, but this would involve considerable overheads.
4. If the Principals have not yet requested tickets for services from the TGS, then the TGS may not know their existence, and certainly not their shared key. Thus the AS must act as an authenticator for the server, given a chain of delegations, as it is the only object that can decrypt each of the signed components in the chain.
5. The authenticating token should contain information on the delegation of authority. This should be sufficient to identify the chain of authorisation and also to validate the actual ticket for the server.

The above leads us to conclude that an extension to the Kerberos mechanism is needed for managing proxy. We shall use a scheme based on the above secret key methods. We shall assume that the AS acts as an authentication server for the end points, and that the secret key of each object is the key generated by the AS for use between the Client and the TGS (and therefore known by the AS for authenticated objects).

Firstly let us assume that the Client (delegator), A, has obtained an authorised ticket from the TGS for a Server, S. This is obtained from the TGS in the standard manner, including the session key. Client A may then choose to communicate directly, in the normal Kerberos manner, with S. However, A may choose instead to delegate this request to another Client, B (where B must be registered with the AS).

We consider two delegation scenarios using Kerberos. First consider the following one, with the syntax as before:

A delegates to B

1) $A \rightarrow B : \langle A, B, S, t_a, \text{duration}_a, DT_a, T_a \rangle$

where

$DT_a = \langle A, B, S, t_a, \text{duration}_a \rangle_{K_{a,tg}}$ is the delegation token

$T_a = \langle T_{a,s} \rangle_{K_s}$ is the token proving A's right to the service.

B is given the details of its delegation.

B (as a delegate) then requests use of server S.

2) $B \rightarrow S : \langle DR, A, DT_a, T_a \rangle$

where

DR is the delegate's service request, containing the information of T_a in the Kerberos protocol described above.

$DR = \langle B, S, t_b, \text{duration}_b, \langle B, S, t_b, \text{duration}_b \rangle_{K_{b,sg}} \rangle$

The time stamp t_b subsequently allows B to verify the freshness of message (5). S reads token T_a and checks whether A has the delegated right. In the process, it also gets hold of $K_{a,s}$.

3) $S \rightarrow AS : \langle S, AS, \langle S, AS, t_s, DR, A, DT_a \rangle_{K_s} \rangle$

Here, S asks AS for authentication of the delegation token as having come from A, and the delegate's service request as having come from B. The time stamp t_s subsequently allows S to verify the freshness of AS's message (4).

4) $AS \rightarrow S : \langle AS, S, \langle t_s+1, K_{b,s}, \langle K_{b,s}, t_b \rangle_{K_{b,sg}} \rangle_{K_s} \rangle$

AS authenticates the delegation chain by checking whether A has given the right to B and B is in fact making the request. It also provides a session key $K_{b,s}$ between delegate and server. Furthermore AS can pass this session key to B via S.

5) $S \rightarrow B : \langle S, B, \langle t_b+1 \rangle_{K_{b,s}}, \langle K_{b,s}, t_b \rangle_{K_{b,sg}} \rangle$

To complete the process, S sends the session key to B, extracted from (4). Its encryption under $K_{b,sg}$ preserves its secrecy. Having obtained $K_{b,s}$, B is able to verify using t_b that S (and AS) have replied to a fresh message (3), so that the session key is indeed fresh.

Alternatively, instead of routing the key $K_{b,s}$ to B via S, one can make the AS reply directly to B. However in this case it will not allow B to know whether S has in fact received the key $K_{b,s}$ from AS.

Let us now consider an alternative delegation protocol for the Kerberos system. The difference between this one and the one presented above is that in this case, we use the Ticket

Granting Server to perform the checking of the delegation chain instead of the Authentication Server (see point 4 above).

1) The first step is exactly the same as the one given above, where A delegates to B.

B then follows the standard Kerberos procedure of requesting TGS for a ticket to the server, except now it sends the message that it received from A to indicate that it is acting on behalf of A.

2) $B \rightarrow TGS : \langle S, \langle T_{b,tgs} \rangle_{K_{tgs}}, \langle A_b \rangle_{K_{b,tgs}}, DT_a, T_a \rangle$

TGS can now authenticate the delegation chain by checking whether A has the right to use the server, whether A has delegated the rights to B, and whether B is making the request. Then TGS can send the following to B

3) $TGS \rightarrow B : \langle S, K_{b,s}, t_b + 1, MT_b \rangle_{K_{b,tgs}}$
 where
 $MT_b = \langle T_{b,s}, T_{a,s}, \langle A, B, S, t_a, duration_a \rangle \rangle_K$

B can now pass the modified delegated token MT_b to the server and authenticate itself using the session key $K_{b,s}$. The final messages follow original Kerberos, using MT_b .

4) $B \rightarrow S : \langle A_b \rangle_{K_{b,s}}, \langle MT_b \rangle$

A_b is a service request of the same form as A_c in the original Kerberos protocol above, containing time stamp t_b . MT_b allows the server to check for itself the delegation from A to B, and A's rights to S.

5) $S \rightarrow B : \langle t_b + 1 \rangle_{K_{b,s}}$

B is able to verify the freshness of S's response using the time stamp.

Notice that in each of the above protocols the session key given to the original delegating client A and the server S, namely $K_{a,s}$, is still maintained. A received this session key from TGS, and the server received the key in the ticket $T_{a,s}$, generated by TGS. We may have not used this key, yet it is still secret to this client/server pair.

A big advantage of retaining the session key in these protocols is that it could be used in the revocation process, as follows. A may send a message directly to S, instructing S to deny any delegate's request. The server does not have to use AS to authenticate this request, as it may do so itself using $K_{a,s}$. It may then promptly take appropriate action, and will not have to rely on the performance and availability of AS.

6 Discussion

In this paper, we have considered a practical problem that arises in distributed object systems. In such systems, the dependency between objects is a common phenomenon, which implies that changes occurring in an object's state are required to be properly notified to other objects to ensure correct operation. We considered a typical scenario, namely the producer-consumer case, which occurs in many office applications. We considered its security implications, solutions to which require suitable authentication and access control mechanisms. We described a solution based on the increasingly publicized Kerberos authentication system. Trust implications of the proposed solution led to the more general problem of proxy or delegation in distributed systems. We have considered the proxy problem in detail in [9]. We concluded this paper by proposing an extension of the Kerberos system to handle proxy situations. The extension proposed provides an added advantage for revocation, in providing a secure channel between the originator and the end point.

7 References

- [1] CCITT *Distributed Applications Framework*, CCITT SG VII/Q19 - DAF.
- [2] ISO *Open Distributed Processing*, ISO/IEC JTC1/SC21/WG7.
- [3] ISO/IEC JTC1/SC21 N5045. *Working Draft Access Control Framework*. SC21/WG1 Security Ad Hoc Group, Seoul Meeting, July 16, 1990.
- [4] Steiner, Jennifer G., Neumann, Clifford, and Schiller, Jeffrey I., *Kerberos: An Authentication Service for Open Network Systems*, Version 4, Project Athena, Massachusetts Institute of Technology. Presented at USENIX 1988, Dallas, Texas.
- [5] Kohl John, Neumann Clifford, Steiner Jennifer., *Kerberos Version 5, Draft RFC*, Project Athena, MIT, Dec.1990.
- [6] Bellovin Steven, Merrit Michael., *Limitations of the Kerberos Authentication System*, Computer Communications Review, Vol.20 No.5, Oct.1990, pp119-132.
- [7] Meyer, B. *Object-Oriented Software Construction*, Prentice-Hall International, 1988.
- [8] V.Varadharajan, S.Black, *Multilevel Security in a Distributed Object Oriented System*. Proceedings of the Annual Computer Security Applications Conference, 1990, Tucson, Arizona.
- [9] V.Varadharajan, P.Allen and S.Black, *An Analysis of the Proxy Problem in Distributed Systems*, To be Published in the Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy, 1991.

Output Perturbation Techniques for the Security of Statistical Databases*

Kasinath C. Vemulapalli

Elizabeth A. Unger[†]

Department of Computing and Information Sciences
Kansas State University

Abstract

In the past a number of techniques have been proposed to avoid inferential security breaches in statistical databases. The best methods qualitatively as well as quantitatively were based on output perturbation. In this paper we present four output perturbation techniques as deterrents to compromise in statistical databases. We analyze the techniques for the deterrent value against compromise and the statistical consequences such as the amount of bias they introduce in the values of the statistics released. In particular, we analyze the techniques for *sum* queries and also compare the bias and deterrence of these techniques. The analysis is done for exact compromise. Compromise accomplished by averaging, a common problem with output perturbation technique, is avoided by releasing same answer for identical query sets.

1 Introduction

A database consists of a model of some part of the real world. Such a model is made up of entities (elements of the part of the real world modeled), attributes (characteristics of the entities), and relationships among different entities. Entities with identical attributes constitute a particular entity type (e.g., *Patient* in a Hospital Database.) A database system that enables its users to retrieve only aggregate statistics (e.g., sample mean and count) for a subset of the entities represented in the database is called a Statistical Database System (SDB.) Some examples of SDBs are test data for manufacturing processes and data released by the Census Bureau. These examples are special-purpose databases, since providing aggregate statistics is their only purpose. In other situations, a single database may serve multiple purposes. A hospital database, for instance, might be used by physicians to support their medical work as well as the statistical researchers of the National Health Council. In this case, statistical researchers are authorized to retrieve only aggregate statistics; the physicians, on the other hand can retrieve microdata from the database.

The problem of providing security in both types of the databases described above has attracted much attention in the recent years. This problem is greatly complicated by the possibility that a legitimate user could ask many different "legal" queries and infer confidential information from them. The inference is the deduction of confidential data from non-sensitive data objects. In addition, user might process either "public" (age, sex, marital status, etc.) or confidential (salary, Grade Point Average, etc.) information about certain individuals, and use this knowledge in framing queries to obtain information for other attributes on those individuals.

*Copyright © 1991 Elizabeth A. Unger

[†]Partially funded by CCRCA under Contract 91E014

2 Statistical Database Model

We describe a statistical database in terms of an abstract model. Although the model does not accurately describe either the logical or physical organization of most databases, its simplicity allows us to focus on the disclosure problem and facilitate analysis.

The *information state* of a statistical database system has two components: the data stored in the database and external knowledge. The database contains information about the attributes of N individuals or entities (organizations, companies, persons, etc.). There are M *attributes*, where each attribute $A_j (1 \leq j \leq M)$ has $|A_j|$ possible values. An example of an attribute is *Sex*, whose two possible values are *Male* and *Female*. We let x_{ij} denote the value of attribute j for individual i . When the subscript j is not important to the discussion, we shall write simply x_i to denote the value of an attribute A for individual i .

It is convenient to view a statistical database as a collection of N records, where each record contains M fields, and x_{ij} is stored in record i , field j . Note that this is equivalent to a relation (table) in a relational database, where the records are M -tuples of the relation. If the information stored in the database is scattered throughout several relations, then the natural join of these relations would be the database we would be looking at as equivalent to SDB.

A disclosure may be either exact or approximate. *Exact disclosure* occurs when q is determined exactly. In this paper we use *compromise* and *exact compromise* interchangeably. *Approximate disclosure* occurs when q is not determined exactly. Dalenius describes three types of approximate disclosure [5]. First, a disclosure may reveal *bounds* L and U such that $L \leq q \leq U$. Second, a disclosure may be *negative* in the sense of revealing that $q \neq y$, for some value y . For example, a user may learn that $\text{sum}(\text{Dept} = EE * \text{Sex} = \text{Female}, \text{GPA}) \neq 3.5$. Third, a disclosure may be *probabilistic* in the sense of disclosing information that is true only with some probability.

Complete compromise is said to occur when one deduces everything in the database. *Partial disclosure* is said to occur if deductions regarding some individuals can be made but the entire database is not deduced. Finally, if no positive or negative disclosure can occur in a database, then the database is *strongly secure*. If only negative disclosure can occur in a database, then the database is *weakly secure*.

3 Previous work

Several techniques are proposed to deter inferential attacks on SDBs. These methods have been classified under four approaches: conceptual, query restriction, data perturbation, and output perturbation[2]. Two models are based on conceptual approach: the conceptual model [4] and the lattice model [9]. Each of these models present a framework for better understanding and investigating the security problem of SDBs. Neither presents a specific implementation procedure. Some query restriction techniques are query set size control[6], query set overlap control[7], and partitioning[12]. Most of these techniques are ineffective against inferential attacks such as Trackers. Data perturbation introduce noise in the data stored. The problem with this control is that it cannot be used in general purpose databases. Output perturbation techniques introduces noise in the data released whereas the data stored is untouched. So in general output perturbation techniques are effective for both static and dynamic databases. This is because the snoopers in general do not have the ability to insert and delete records.

Output perturbation techniques can be classified into two categories: record based, and result based. The record based output perturbation techniques introduce noise in each record values before the statistic is calculated, whereas, result based output perturbation techniques add noise in the result or in only one record randomly before the statistic is released. Random Sample Queries by Denning[7],

Varying-Output Perturbation by Beck[3] are record based techniques, whereas Rounding by Achugbue and Chin[1], Dalenius[5], and Duplication/Deletion by Kaushik[10] are result based techniques. We will briefly describe the mechanisms proposed by Denning[7], Beck[3] and Kaushik[10].

The method introduced by Kaushik[10,14] is based on introducing uncertainty in the released statistic by perturbing one record in the query set. This perturbation is accomplished by duplicating, deleting or returning the true value of one of the records in query set. The attractive feature of this method is that since only one record is perturbed and as the size of the query set increases the bias introduced will diminish and yet the deterrent value is not reduced.

Formally, the scheme is as follows:

1. If a query, $q(C)$, is answerable, then one of the following three options is chosen in order to report the results to the user,
 - The query response is calculated from the set of records obtained after duplicating a record in the query set.
 - The query response is calculated from the set of records formed by deleting a record from the query set.
 - The query response is the true query set.
2. The decision to choose one of the three options is random. However, it is necessary that the two conditions below be satisfied:
 - The same query option must be chosen for any query resulting in the same query set.
 - If two queries result in the same query set, and if the option chosen is to duplicate/delete a record, the same record must be duplicated/deleted from the two query sets regardless of the order of the records in the query sets or the formulation of the queries.

These restrictions are necessary because compromise could occur if different options, (e.g., delete) and different records are chosen as the query is repeatedly posed to the database, an accurate estimate of the true response can be deduced by averaging.

Kaushik's work includes the evaluation of the method's effectiveness against individual, general and double trackers for exact disclosure. The bias introduced by the method is zero for statistics like mean and relative frequency. The variance of mean is given by,

$$Var(\bar{\mu}) = \sigma^2 p \left(\frac{(n+3)}{(n+1)^2} + \frac{1}{(n-1)} - \frac{2}{n} \right) + \frac{\sigma^2}{n}$$

where σ^2 is the variance of the true values of the query set, n is the query set size, and p is the probability of duplication or deletion (assuming equal probability). It is clear that variance of the mean is a function of variance of the original query set and decreases with the increase in the size of the query set. For more discussion on compromise and bias of this technique refer to Kaushik[10].

4 Our Methods

Beck[3] describes that perturbation based on the statement of the query, the membership of the response set, the variance of the individual values in the response set, and many other factors which can

be defeated by the well known averaging. Our technique however is not based on any one of the above factors. The possibility of averaging is averted by providing identical answers to identical query sets regardless of the formulation the query[11].

Method-1: Every record Addition/Deletion

This method reports the results from a query set by perturbing each record in the query set. The scheme is as follows:

1. For each query, $q(C)$ ($Sum(C, Y)$), mean of the attribute Y for the records in the query set is calculated and multiplied by a constant fraction h determined by the database administrator. Let the product be $h\bar{y}$. Each record is perturbed by one of the options given below before reporting the results to the user:

- Add the product $h\bar{y}$ to the value of the record.
- Subtract the product $h\bar{y}$ from the value of the record.
- Retain the true value of the record.

2. The decision to choose one of the three options is random. However, it is necessary that the following conditions be satisfied:

- For identical query sets the number of additions, subtractions and true values must be constant regardless of the composition of the query.
- In addition, the additions should be over the same records (also subtractions and true values) for the same query set.

The reason for the the above restriction is to disallow compromise by averaging.

Advantages of this method are that, it is suitable for all statistics, it does not introduce any bias in the expected values, and the variance has attractive properties. The probability of exact compromise is very low in this technique and increases with query set size. Exact compromise would be possible only if the means \bar{y} of the series of queries which are involved in compromise are equal. This method is simple and has been analyzed statistically in sections 4.1 and 4.2.

Method-2: Every record Addition/Subtraction within a Range

This method is exactly same as the previous one except for the value of h . h takes a value in some range between 0 and 1. Formally,

$$L \leq h \leq U$$

where,

$$0 \leq L \leq 1 \text{ and } 0 \leq U \leq i$$

The values of L and U can be selected by the Database Administrator and h is randomly generated in the range. This method results in higher bias but provides better security.

Method-3: Every record Rounding

In this method the effectiveness of a rounding technique at record level is investigated. Rounding at record level eliminates the compromise as reported in Achugbue and Chin[1] for systematic rounding. The scheme is as follows:

Each record is rounded up or down to the nearest multiple of some base b . Let $b' = \lfloor (b+1)/2 \rfloor$ and $d = q \bmod b$. Let x' be the perturbed value of record value x . Then,

$$x' = \begin{cases} x & \text{if } d = 0 \\ x - d & \text{if } d < b' \text{ (round down)} \\ x + d & \text{if } d \geq b' \text{ (round up)} \end{cases}$$

Simulation results of this method have been very encouraging with zero compromise and bias introduced being marginally lesser than Method-1 and Method-2.

Method-4: Addition/Subtraction to single record

This method is similar to Kaushik's method, but instead of duplicating or deleting a record, we add or subtract a product of the mean \bar{y} i.e., $h\bar{y}$ from one of the record value which is selected at random using some random distribution. Care is taken to return same response for identical queries. The method is as follows:

1. If a query, $q(C)$, is answerable, then one of the following three options is chosen in order to report the results to the user,
 - The query response is calculated from the set of records obtained after adding to a record in the query set, the product $h\bar{y}$.
 - The query response is calculated from the set of records formed by subtracting from a record in the query set, the product $h\bar{y}$.
 - The query response is the true value.
2. The decision to choose one of the three options is random. However, it is necessary that the two conditions below be satisfied:
 - The same option must be chosen for any query with the same query set.
 - If two queries result in the same query set, and if the option chosen is to add/subtract from a record, the same record must be chosen from the two query sets regardless of the order in which records are put together in the query sets.

The main disadvantage Kaushik's method had was that, if the database has extreme values and if one of those values were selected for duplication/deletion the bias would be high. The modification proposed in method-4 eliminates high bias problem by adding/subtracting some fraction of the mean of the query set. The fraction can be decided by the DBA depending on the confidentiality of the data.

4.1 Bias for Total(sum) queries

Suppose that a query requests $sum(C, Y)$ where Y represents an attribute. Let y_i be the true value for the i^{th} record in the response set R , and let n be the number of records in R . The mean value of the response set is denoted by \bar{y} .

As a response to the query, we return the value of

$$T = \sum_{i=1}^n x_i$$

where

$$x_i = y_i + Xh\bar{y}$$

where X is the random variable and the distribution function $f(x)$ is given by:

$$f(x) = \begin{cases} p_1 & \text{when } X = -1 \\ p_2 & \text{when } X = 1 \\ 1 - p_1 - p_2 & \text{when } X = 0 \end{cases}$$

where p_1 , p_2 and $(1 - p_1 - p_2)$ are the probabilities of subtracting the fraction $h\bar{y}$, adding it, and returning the true value respectively of an attribute value.

The expected value of T ,

$$\begin{aligned} E(T) &= E(\sum x_i) \\ &= E(\sum (y_i + Xh\bar{y})) \\ &= \sum E(y_i + Xh\bar{y}) \\ &= \sum E(y_i) + \sum E(Xh\bar{y}) \\ &= n\bar{y} + nh\bar{y}E(X) \end{aligned}$$

when $p_1 = p_2$, $E(X) = 0$, hence,

$$E(T) = n\bar{y}$$

So the bias introduced is zero if the probability of addition is equal to the probability of subtraction.

The variance of the total T ,

$$\begin{aligned} Var(T) &= Var(\sum (y_i + Xh\bar{y})) \\ &= Var(\sum y_i + \sum Xh\bar{y}) \\ &= Var(\sum Xh\bar{y}) \\ &= n^2 h^2 \bar{y}^2 Var(X) \\ &= n^2 h^2 \bar{y}^2 (E(X^2) - (E(X))^2) \end{aligned}$$

when $p_1 = p_2 = p$, $E(X) = 0$, and $E(X^2) = 2p$, hence,

$$Var(T) = 2n^2 h^2 \bar{y}^2 p$$

The variance calculated above has some attractive properties. It depends on the size of the query set n as a square, thus, the standard deviation increases linearly with respect to n . Also more important, the variance is dependent on the fraction h which can be decided by the database administrator. As the value of h decreases, standard deviation decreases linearly.

4.2 Compromise

Let q_1 and q_2 be two queries such that $n_1 = |q_1(C_1)| = n + 1$ and $n_2 = |q_2(C_2)| = n$. Also let the overlap of q_1 and q_2 be 'n'. Let X_1 and X_2 be the random variables used in the calculating the perturbed results of q_1 and q_2 . Let $P(x)$ denote the probability of 'x' being true. Let p_1 and p_2 denote probability of addition and subtraction respectively. The probability of compromise of $(n + 1)^{th}$ record value y_{n+1} is,

$$\begin{aligned} P_c &= P(y_{n+1} = \sum_{i=1}^{n+1} (y_i + h\bar{y}_1 X_1) - \sum_{i=1}^n (y_i + h\bar{y}_2 X_2)) \\ &= P(\sum_{i=1}^n (y_i + h\bar{y}_1 X_1) = \sum_{i=1}^n (y_i + h\bar{y}_2 X_2)) (1 - p_1 - p_2) \\ &\quad + P(\sum_{i=1}^n (y_i + h\bar{y}_1 X_1) = h\bar{y}_2 + \sum_{i=1}^n (y_i + h\bar{y}_2 X_2)) p_2 \\ &\quad + P(\sum_{i=1}^n (y_i + h\bar{y}_1 X_1) = -h\bar{y}_2 + \sum_{i=1}^n (y_i + h\bar{y}_2 X_2)) p_1 \end{aligned}$$

Let $Y_1 = \sum_{i=1}^n \bar{y}_1 X_1$ and $Y_2 = \sum_{i=1}^n \bar{y}_2 X_2$, Now,

$$P_c = (1 - p_1 - p_2)(P(Y_1 = Y_2)) + p_2(Y_1 = 1 + Y_2) + p_1(Y_1 = -1 + Y_2)$$

When $\bar{y}_1 = \bar{y}_2$, the probability distribution of Y_i is given by,

$$f_{Y_i}(j) = \begin{cases} \sum_{m=j}^{\lfloor (n+j)/2 \rfloor} p_2^m p_1^{m-j} (1 - p_1 - p_2)^{n-2m+j} \binom{n}{m} \binom{n-m}{m-j} & \text{for } j \geq 0 \\ \sum_{m=-j}^{\lfloor (n-j)/2 \rfloor} p_2^m p_1^{m+j} (1 - p_1 - p_2)^{n-2m-j} \binom{n}{m} \binom{n-m}{m-j} & \text{for } j \leq 0 \end{cases}$$

Now,

$$\begin{aligned} P(Y_1 = Y_2) &= \sum_{i=-n}^n f_{Y_1}(i) f_{Y_2}(i) \\ P(Y_1 = 1 + Y_2) &= \sum_{i=-n}^{n-1} f_{Y_2}(i) f_{Y_1}(i+1) \\ P(Y_1 = -1 + Y_2) &= \sum_{i=-n}^{n-1} f_{Y_1}(i) f_{Y_2}(i+1) \end{aligned}$$

When $p_1 = p_2 = p$, latter equations become equal, therefore,

$$P_c = (1 - 2p) \sum_{i=-n}^n f_{Y_1}(i) f_{Y_2}(i) + 2p \sum_{i=-n}^{n-1} f_{Y_1}(i) f_{Y_2}(i+1)$$

As the probability of compromise is higher for small query set sizes, our method would yield best results when used with query set size control. Also, if the number of queries needed for compromise increases (for example, general trackers[6] need 4 queries for compromise as against 2 by individual tracker and double tracker needs at least 4 queries) the probability of compromise decreases exponentially. This is in addition to the exponential decrease of probability with the increase of query set size.

5 Simulation and Results

In this section a description of the simulation and comparison of the four methods is presented. It is encouraging to note that the simulation results agree with the analytical results of sections 4.1 and 4.2 for Method-1.

We again consider the two query compromise situation described in section 4.2, for simulation. Let q_1 and q_2 be two *sum* queries such that $n_1 = |q_1(C_1)| = n+1$ and $n_2 = |q_2(C_2)| = n$. Also let the overlap of q_1 and q_2 be n . Let Y be the attribute being compromised. Let y_i be the value of Y for i^{th} record. Let \bar{y}_1 and \bar{y}_2 be the mean of q_1 and q_2 respectively. Let x_i be the perturbed value of the record i released after applying one of the methods discussed above.

Let δ be defined as follows:

$$\delta = \sqrt{\sum_{i=1}^N (y_i - x_i)^2 / N}$$

where N is the size of the query. δ gives the amount of bias introduced by the methods.

The simulated database contains random values as follows. It has 80% values in a narrow range, and 10% very large and 10% very small values. This gives a true picture of confidential attributes such as salary, GPA etc. In the following tables we present a comparison of bias introduced and deterrence against exact compromise of the methods presented in section 4.

Method	Query Size	True.o/p (Qry1)	Compromise	Delta(bias) (Qry1)	Sum (Qry1)	% bias (Qry1)
1	1-5	415	105	260	14402	1.823
2		10	15	168	14137	1.191
3		5	0	105	13008	0.805
4		325	78	93	16426	0.509
1	5-25	195	45	334	63891	0.514
2		2	2	358	52391	0.565
3		3	0	245	62337	0.408
4		324	76	93	65291	0.148
1	10-100	100	26	575	227496	0.248
2		2	1	708	223337	0.298
3		5	0	503	221063	0.226
4		325	84	93	225590	0.045

Table-1: Bias and Compromise for 1600 trials.

Some explanation is needed about the terms used in the table. For each query size per method 1600 trials are performed. The methods are tested with three different query sizes (Column-2.) Column-3 gives the number of instances (out of 1600) when perturbed value equals the true value. Column-4 gives the number of instances of compromise in 1600 trials. Column-5 and 6 give the average bias from the true value, and sum respectively from 1600 trials. Column-7 gives the bias as a percent of sum.

From the above table it is apparent that method-1 improves in terms of compromise as well as bias (from size of the query increases. Method-2 has better deterrence properties but bias is somewhat higher. The performance improves with size. Method-3 seems best it has zero compromise and less bias. Method-4 has compromise almost constant but bias is considerably reduced as the query size increases.

Database Administrator (DBA) has control over the amount of bias introduced in the answers. In

Methods-1,2 and 4, value of the fraction h can be adjusted where as the value of base b can be changed in Method-3. The following table illustrates the effect of change of the fraction h for Method-1.

Query Size	h -value	Compromise	Delta (Qry1)	Sum (Qry1)	% bias
1-5	0.1	103	1315	14402	9.132
	0.04	103	526	14402	3.650
	0.02	106	263	14402	1.823
	0.01	108	131	14402	1.213
5-25	0.1	32	1623	62826	2.583
	0.04	37	670	63891	1.03
	0.02	50	334	63891	0.515
	0.01	50	166	63891	0.342
10-100	0.1	9	2786	223177	1.248
	0.04	21	1130	227496	0.498
	0.02	33	564	227496	0.248
	0.01	44	280	227496	0.165

Table-2. Effect of h on Bias and compromise, Method-1

It can be observed that bias decreases as h value decrease. This allows the DBA to set the fraction h of the database to the required value, to obtain the required degree of security. A note about the compromise in the above table would be necessary. Integer arithmetic has been used in the calculation of mean and the average, hence the the number of exactly compromised instances got slightly inflated.

6 Conclusions and Future work

Release of confidential information of an individual using inference control has been of interest of many researchers for quite a long time. A number of methods were proposed which were either very expensive or not very effective. In this paper we presented four effective output perturbation method to deter compromise which are computationally inexpensive. Method-4 which involves perturbing the result is very inexpensive. Statistical analysis is done for Method-1 and simulation studies are done for others and a comparison is presented. It has been observed that these methods are increasingly effective as the size of the query set increases.

Other methods can be statistically analyzed as an extension to this work. These methods can be tested on real database which would give a better feel of their performance. An improvement of the implementation is possible. Right now the averaging problem is averted by seeding the random number generation on the size of the query set. This might not work if the order of the records is not maintained. Ordering can be achieved by sorting the records, but sorting would be expensive. Literature suggests that transforming the query into a canonical form before it is submitted to the database would be effective in many cases[11]. These issues are still open for research community to resolve.

References

- [1] Achugbue, J.O., and Chin, F.Y., "Effectiveness of output modification by rounding for protection of statistical databases", *INFOR* Vol.17, No.3, pp.209-218 (Aug 1979).
- [2] Adam, N. R., and Wortmann, J. C., "Security-Control Methods for Statistical Databases: A Comparative Study", *ACM Computing Surveys*, Vol.21, No.4, pp. 515-556 (Dec. 1989).

- [3] Beck, L. L., "A Security Mechanism for Statistical Databases", *ACM Transactions on Database Systems*, Vol. 5, No. 3, pp. 316-338 (Sep. 1980).
- [4] Chin, F. Y., and Ozsoyoglu, G., "Statistical Database Design", *ACM Transactions on Database Systems*, Vol. 6, No. 1, pp. 113-139 (Mar. 81).
- [5] Dalenius, T., "A Simple Procedure for Controlled Rounding", *Statistik Tidskrift*, Vol. 3, pp. 202-208, 1981.
- [6] Denning, D. E., Denning, P. J., and Schwartz, M. D., "The Tracker: A Threat to Statistical Database Security", *ACM Transactions on Database Systems*, Vol. 4, No. 1, pp. 76-96 (Mar. 1979).
- [7] Denning, D. E., "Secure Statistical Databases with Random Sampling Queries", *ACM transactions on Database Systems*, Vol. 5, No. 3, pp. 291-315 (Sep. 1980).
- [8] Denning, D. E., *Cryptology and Data Security*, Addison-Wesley Publishing Company, Inc., USA, 1982.
- [9] Denning, D.E., "A Security Model for Statistical Database Problem", *Proceedings of the Second International Workshop on Management*, pp. 1-16, 1983.
- [10] Kaushik, N., *A New Deterrent to Compromise of Confidential Information from Statistical Databases*, M. S. Thesis, Kansas State University, 1988.
- [11] Maxwell, R., *Output Perturbation Deterrent to Trackers*, M.S. Report, Kansas State University, 1990.
- [12] McLeish, M., "Further Results on the Security of Partitioned Dynamic Statistical Databases", *ACM Transactions on Database Systems*, Vol. 14, No. 1, pp. 98-113 (Mar. 1989).
- [13] Schlörer, J., "Disclosure from Statistical Databases: Quantitative aspects of Trackers", *ACM Transactions on Database Systems*, Vol. 5, No. 4, pp. 467-492 (Dec. 1980).
- [14] Unger, E. A., McNulty, S. K., "Natural Change in Dynamic Database as a Deterrent to Compromise by Tracker", *Proceedings of Sixth Annual Computer Security Applications Conference*, pp. 116-124 (Dec 1990).
- [15] Vernulapalli, K.C., and Unger, E.A., "Investigations of output perturbation techniques", Technical Report, Dept. of Comp. and Info. Sciences, Kansas State University, 1991.

AN OVERVIEW OF INFORMIX-ONLINE/SECURE*

Rammohan Varadarajan
Informix Software, Inc.
4100 Bohannon Drive
Menlo Park, CA 94025
ramm@informix.com

This paper discusses the architecture of the Informix trusted DBMS product, INFORMIX-OnLine/Secure, a solution for open system environments based on the *Trusted Database Interpretation's* [3] trusted subject architecture description. INFORMIX-OnLine/Secure runs as a trusted application on secure UNIXTM platforms. It provides row level labeling, supports a large label space, provides multimedia capability, and supports on-line transaction processing. It has a small trusted computing base (TCB) that adheres to the "least privilege" doctrine.

INTRODUCTION

INFORMIX-OnLine/Secure is targeted to meet all requirements for the Orange Book [1] B1 class. It has its origins in the INFORMIX-OnLine product. In building INFORMIX-OnLine/Secure, we have gone beyond traditional retrofit exercises as exemplified by many B1 UNIX systems. We have elected to re-architect the system to provide high assurance. We have adhered to well-endorsed techniques of software reuse and layering to ensure that retrofitting security does not degrade the quality or functionality of an already proven product.

The major goals for INFORMIX-OnLine/Secure are:

- Multiple configurations. In addition to the primary B1 configuration, a C2 compliant configuration, and a B1/EP "enhanced performance" configuration will be available.
- High assurance. INFORMIX-OnLine/Secure's TCB is small, well structured, and adheres to the principal of least privileged. Covert channel analysis and penetration tests are part of our development process.
- Quality. The system is being developed following DOD-STD-2167A [4] methodology, modified to account for retrofit activity. DOD-STD-2168 [5] governs the project's quality control aspects.
- Security Functionality. INFORMIX-OnLine/Secure supports as many labels as the operating system it runs on will support.
- Performance and DBMS Functionality. The hallmarks of the INFORMIX-OnLine product — high performance OLTP, multimedia capability, etc., — are retained.
- Portability. INFORMIX-OnLine/Secure executes on all secure UNIX platforms. The architecture does not rely on features specific to any particular secure UNIX platform.

The major tenets influencing the architecture and design decisions are:

- Do not re-invent technology.
- Keep technology where it belongs.

As a result, there is no need to develop new login protocols, network security components, canonical label formats, or modifying secure UNIX device drivers.

* © Copyright 1991, Informix Software, Inc. Informix is a registered trademark of Informix Software, Inc. Other names indicated by ® or TM are registered trademarks or trademarks of their respective manufacturers.

OPERATIONAL ENVIRONMENT

INFORMIX-OnLine/Secure does not place any restrictions of its own on the operational environment. It does, however, expect to be executing on a B1 secure UNIX operating system. In addition, the configuration of the operating system must provide for the infrastructure for INFORMIX-OnLine/Secure to support various user roles, data isolation, and an audit mechanism.

SUPPORT FOR IDENTIFICATION

INFORMIX-OnLine/Secure relies entirely on the secure UNIX TCB calls to identify and authenticate the security attributes of a user session. There is no login to the DBMS.

SUPPORT FOR USER ROLES

INFORMIX-OnLine/Secure supports three user roles: Database System Administrator, Database System Security Officer, and the regular DBMS user. Some user roles possess more privilege than others. Adequate controls on proliferation of privileges are critical. INFORMIX-OnLine/Secure achieves this control by procedural methods which require operational environment support.

INFORMIX-OnLine/Secure User

INFORMIX-OnLine/Secure expects the operational environment to provide a special UNIX group ("ix_users") to which all users of INFORMIX-OnLine/Secure must belong.

Database System Administrator

The Database System Administrator (DBSA) is charged with maintaining INFORMIX-OnLine/Secure. A special category ("IX_DBSA") and group ("ix_dbsa") must be set aside for the exclusive use of the DBSA. The DBSA must be permitted to log in only at the security level DataHigh* + IX_DBSA and all levels that this label dominates. There can be multiple DBSA login accounts on each system.

Database System Security Officer

The Database System Security Officer (DBSSO) is entrusted with maintaining the security of an INFORMIX-OnLine/Secure system through such tasks as re-labelling data, reassigning privileges and configuring audit granularity. A special category ("IX_DBSSO") and group ("ix_dbssso") must be set aside for the exclusive use of the DBSSO. The DBSSO must be permitted to log in only at the security label DataHigh + IX_DBSSO. There can be multiple DBSSO login accounts on each system.

ISOLATION OF DATA STORES

INFORMIX-OnLine/Secure has to isolate its data stores from operating system processes (other than the DBMS TCB). A special category ("IX_DATA") and group ("ix_data") must be set aside for use in labeling the device containing the data stores in INFORMIX-OnLine/Secure. A process must possess the IX_DATA category and be a member of the ix_data group to access the DBMS data stores through the MAC and DAC mechanisms of the secure operating system. No users should possess the category IX_DATA or be a member of ix_data.

SUPPORT FOR AUDIT

Audit records generated by INFORMIX-OnLine/Secure are stored in the secure UNIX audit trail. They are marked and distinguishable from other audit records generated by the operating system.

SYSTEM ARCHITECTURE

Figure 1 is a schematic of the INFORMIX-OnLine/Secure architecture. It operates on a client/server paradigm. The main components that make up INFORMIX-OnLine/Secure are the User Front End, the Administrator Front End (for exclusive use by the DBSA), the SQL Engine, the Kernel, and the Secure Administrative Front End (used exclusively by the DBSSO). Each of the components executes as a separate

* "DataHigh" refers to the highest security level at which data can be in the DBMS.

UNIX process or a collection of UNIX processes. Process isolation features of UNIX are used to maintain separate address spaces for each component. Secure UNIX interprocess communication (IPC) primitives are used for communication between the main components. As shown in Figure 1, devices under the control of secure UNIX are used as data stores. A section of the memory is used as a disk cache.

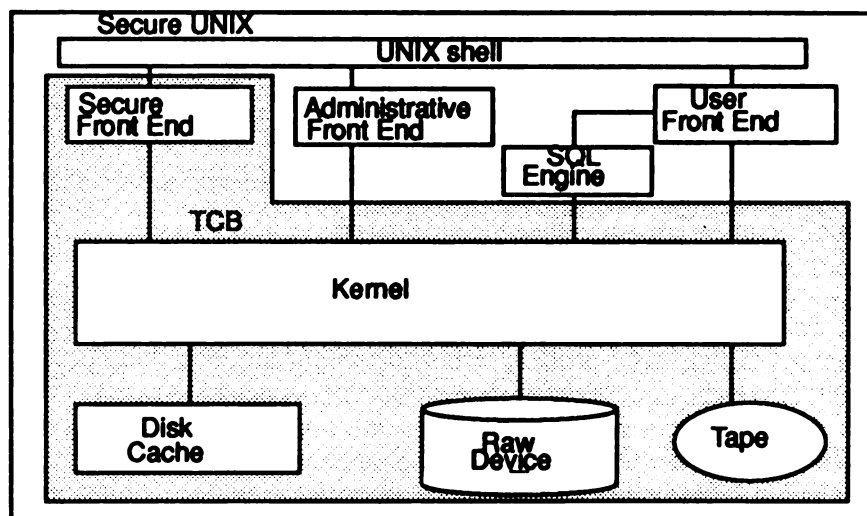


Figure 1 INFORMIX-OnLine/Secure System Architecture

The architecture could be classified as a trusted subject with respect to the operating system. However, the principle of least privilege is enforced by making large portions of the TCB a proper TCB subset by making them single level processes.

Synchronous protocols are used between the processes for communication. The DBMS components are examined in more detail in the following subsections.

FRONT ENDS

Three kinds of front ends can be used to interact with INFORMIX-OnLine/Secure.

User Front End

The User Front End (UFE) is the entity that interacts with the ordinary user.

The User Front End can be written using a wide repertoire of tools available: interactive SQL, embedded SQL in languages like C, Ada, Fortran, or Cobol, 4GL programs, or 3GL programs with direct calls into the TCB. If the user programs are themselves multilevel secure programs, then they have to be 3GL programs written to the trusted application protocol. INFORMIX-OnLine/Secure allows for application programs to be multilevel secure with respect to the DBMS while being single level with respect to the operating system. This is achieved by having the multilevel secure applications run with the special category, IX_DATA.

Administrator Front End

The Administrator Front End (AFE) is a special kind of User Front End, tailored for use by the DBSA. It is a full screen menu driven interface which allows (and restricts) the DBSA to perform only DBMS maintenance related tasks. Access to the AFE is restricted by the category IX_DBSA and group ix_dbsa.

The AFE is an application built using the tools mentioned above. It is untrusted and single threaded. The DBSA role involves system start-up, shut-down, tuning, monitoring, archive, restore and integrity checking actions. The AFE initiates separate processes to do each of these tasks. These processes are part of the Kernel, discussed later.

Secure Administrator Front End

The Secure Administrator Front End (SAFE) is the front end used by the Database System Security Officer (DBSSO) to do the following:

- Label maintenance
- DAC maintenance
- Audit maintenance

The SAFE is part of the TCB. Its interaction with the DBSSO is via a simple, dialogue-driven interface. It is a multilevel secure application written in C. It allows and restricts the DBSSO to perform only operations related to the DBSSO role.

Because this interface is for use only in times of pressing security need, it locks objects when the DBSSO is changing them. If a user happens to have an object locked when the DBSSO is making a change to it, the user's lock is broken, their process is forced to exit, and the transaction rolled back.

Sensitivity Label Maintenance

The MAC policy cannot be circumvented by any user in the system; hence there must be some provisions for correcting mislabeled information in the system. Using the menus provided by the SAFE, the DBSSO can examine and modify a storage object's attributes and sensitivity label to any valid sensitivity label supported by the operating system.

DAC Maintenance

The DBSSO is permitted to add and remove all discretionary access privileges from a user. Ownership attributes of an object can also be changed using the menus provided for DAC maintenance.

Audit Maintenance

The SAFE provides an Audit Maintenance interface which has a set of menus used to assist in maintenance of the audit masks. Audit masks are discussed in a later section.

To help the DBSSO maintain the audit masks, INFORMIX-OnLine/Secure provides a user audit mask report option. This option generates a report as an operating system file that the DBSSO can print showing all of the audit mask catalog.

SQL ENGINE

The SQL Engine translates SQL statements from users and conveys them to the Kernel. In addition to parsing SQL statements, it builds an execution plan, optimizes the execution plan, and executes this plan. Logically, the SQL Engine can be viewed as being made up of a parser, optimizer and plan-executor.

Parser

The parsing operation does not require any multilevel information.

Optimizer

INFORMIX-OnLine/Secure uses the OnLine optimizer. Such optimization typically requires information at levels different from that of the session. For example, information about the total number of rows in a table may be germane to estimate the cost of an operation in terms of I/O and CPU-cycles.

Modifications to the optimizer and the Kernel have been made to ensure that satisfactory optimization can be achieved with sanitized information.

Executor

No multilevel information is needed to interpret the optimized plan. The executor invokes data definition and manipulation operations implemented in the TCB. The SQL Engine, therefore need not be trusted.

KERNEL

The Kernel is the entity within INFORMIX-OnLine/Secure that directly interacts with the device where the data resides. The Kernel implements the access method for reaching data within the DBMS.

There are four types of processes within the Kernel:

- RSAM processes
- Daemon processes
- Support processes
- Transient processes

RSAM Process

The Relational Storage Access Method (RSAM) process is the workhorse of the Kernel. It is the entity within the Kernel that supports all DBMS object abstractions and services requests from outside the TCB.

The RSAM process is not multi-threaded; there is one RSAM process instance for each user session. The RSAM process has to ensure that the integrity of the database is preserved over concurrent execution of several RSAM processes. The disk and disk-cache are shared with other RSAM processes which could be servicing front ends at different security levels.

The RSAM process acts as the reference monitor between the front ends and the data. It is therefore trusted and part of the TCB. It identifies the front end (via the operating system) and performs MAC and DAC checks as expected of the reference monitor. The RSAM process(es) also perform audit activities as per guidelines in [1] and [8].

Support Processes

Support processes are specialized programs that perform infrequent non-periodic tasks. Examples of what they do are:

- starting INFORMIX-OnLine/Secure
- archiving INFORMIX-OnLine/Secure
- restoring INFORMIX-OnLine/Secure

The Support processes are small specialized programs. They are single threaded and execute in separate address spaces as per the UNIX process paradigm. Because of this, the size and complexity of the TCB is not adversely affected. There is no connection between user session instances and the number of active support processes; the support processes are started up by the DBSA when a particular special functionality is needed within the Kernel. Support process need to be trusted because they have access to the system during start-up and have access to the disk and archive tapes that have multilevel data.

Daemon Processes

The Daemon processes are specialized programs that conduct repetitive tasks in service of all RSAM process instances. For example, they:

- periodically write data from the disk cache to disk
- clean up after user processes that terminate abnormally
- signal state changes to RSAM instances

The Daemon processes are small, specialized programs. Each executes in a separate address space. Because of this, the size and complexity of the TCB is not adversely impacted. The Daemon processes are single threaded. There is no connection, however, between user session instances and the number of active daemon processes; the number of Daemon processes is configurable by the DBSA. The Daemon processes must be trusted because they have access to multilevel data on the disk and disk cache.

Transient Processes

The Transient processes are programs used to launch Support and Daemon processes from an untrusted front end.

DISK AND DISK CACHE

The Kernel uses a "raw" device as the disk store for the INFORMIX-OnLine/Secure data.* A raw device is a device without the operating system's file system on the disk.

The raw device and disk cache are labelled at DataHigh + IX_DATA. To the operating system, the disk and the disk cache appear as single-level entities. To INFORMIX-OnLine/Secure, the disk and disk cache are multilevel stores. The DBMS Kernel is the only DBMS entity that can directly access the disk and disk cache. It is trusted to keep the separation of objects at different security levels.

All access to the disk by the Kernel uses the secure UNIX read, write and seek functions. As a consequence, operating system device drivers that are used for disk access must be trusted and function to specification.

B1/EP CONFIGURATION

The difference between the B1 and the B1/EP configurations is that the SQL Engine executes in the same address space as the RSAM process of the Kernel in the B1/EP configuration. This makes the SQL Engine part of the TCB in a B1/EP system.

SECURITY ISSUES

The TCB is the totality of protection mechanisms responsible for enforcing a unified security policy over the system. The "system" is made up of INFORMIX-OnLine/Secure and the underlying operating system. The abstractions (objects) managed by the two components are different. The result is a separate TCB for each component. The system TCB is the combination of the Secure UNIX TCB and the INFORMIX-OnLine/Secure TCB.

The secure UNIX TCB is provided by the UNIX vendor. The INFORMIX-OnLine/Secure TCB is made up of the entire INFORMIX-OnLine/Secure Kernel and the SAFE, as shown by the shading in Figure 1.

Although all the processes within the Kernel are part of the INFORMIX-OnLine/Secure TCB, there is a security distinction between them. The RSAM process and the Transient processes run as multilevel processes (at all security levels) while the Daemon and Support processes run as single level, DataHigh + IX_DATA, trusted processes. In operating systems that provide least privilege, RSAM can execute at that level with the ability to write down to the session level IPC connection.

The reason the Daemon and Support Processes run as single level processes is enforcement of the "principle of least privilege." The Daemon and Support processes only need the following actions to perform their allocated functionality:

- access (read and write) to the cache and the raw device
- communicate with the RSAM process

Since the disk and cache are single level entities, DataHigh + IX_DATA, the Daemons and Support processes must be at least that level. No special security level is required for communicating to the RSAM process, because the RSAM process is multilevel secure (e.g., it runs at all levels). So, there is no need for the Daemon and Support processes to run at any level other than DataHigh+ IX_DATA. The reason to trust them is obvious — the cache and the raw device contain multilevel data.

* A UNIX file may be used as a "cooked" data store.

The RSAM process needs to be multilevel secure because it has to access the raw device and cache (which are at DataHigh + IX_DATA) and communicate with User Front End or SQL Engine processes at various levels between Data Low and DataHigh.

The Transient processes need to be multilevel secure because they have to talk to User Front End processes at various levels, and they spawn trusted children at specific levels. To accomplish this, they need to change their level because secure UNIX does not allow processes to spawn children at levels that are different than their own.

The SAFE runs as a single level DataHigh + DBSSO trusted process. Just as in the Kernel, the reason for the SAFE being single level is the principle of "least privilege."

DESIGN OVERVIEW

It is clearly beyond the scope of this article to describe the design details of INFORMIX-OnLine/Secure. However, some of the salient design features of the INFORMIX-OnLine/Secure TCB are highlighted in the sections that follow.

Changes to INFORMIX-OnLine were warranted for the following reasons:

- To reduce the size of the TCB
In INFORMIX-OnLine, the SQL Engine and RSAM executed in the same address space.
- To close covert channels
Most of the storage and retrieval mechanisms used in vanilla DBMSs provide the user of the system with information about the order and location of DBMS objects within the storage device. OnLine is no exception.
- To provide an acceptable semantics to some existing functionality that is inherently insecure

As was mentioned in the introduction, it is our goal to use good software engineering principles in this retrofit exercise. INFORMIX-OnLine, the baseline for INFORMIX-OnLine/Secure, is a stable product with plenty of field testing. When making changes to the product, we wanted to ensure that we did not introduce errors which thereby negate the valuable field tested correctness that INFORMIX-OnLine/Secure would inherit.

Therefore instead of intrusive changes to the design of OnLine, we decided to build on abstractions and implementation that currently existed. In other words INFORMIX-OnLine/Secure introduces a new software layer, leaving the existing implementation essentially intact.

A NEW TABLE ABSTRACTION

We are implementing a new abstraction within the Kernel called a *Bundle*^{*}. A Bundle is the Kernel's internal implementation of a multilevel table. In INFORMIX-OnLine, the abstraction used to implement a table is called a *tblspace*. A Bundle hides the internal structures of tables in INFORMIX-OnLine/Secure in the same way that tblspaces do in OnLine. For the SQL Engine or any process communicating with RSAM, the table abstraction is indistinguishable from OnLine. This is depicted in Figure 2. This provides an elegant way to hide data about location of single level objects, hence eliminating the covert channels mentioned above. Additionally, it provides a simple yet acceptable semantics for operations like the "serial" data type; values in serial columns are only serial within a level.

Tblspaces maintain exactly the same structure that they have in OnLine. However, they are no longer directly accessible from outside the Kernel. Tblspaces are used as the building blocks in the implementation of Bundles. A Bundle consists of a set of tblspaces, each at its own sensitivity level, sharing a common schema. Additionally, all properties associated with tblspaces in OnLine (for example, locking, logging, etc.), are shared by all the tblspaces that make up a Bundle.

When a user session accesses a table in INFORMIX-OnLine/Secure, the user can see only those tblspaces within the Bundle that are dominated by the user's session sensitivity level.

* The concept and the term are due to Aryeh Tal-Nir.

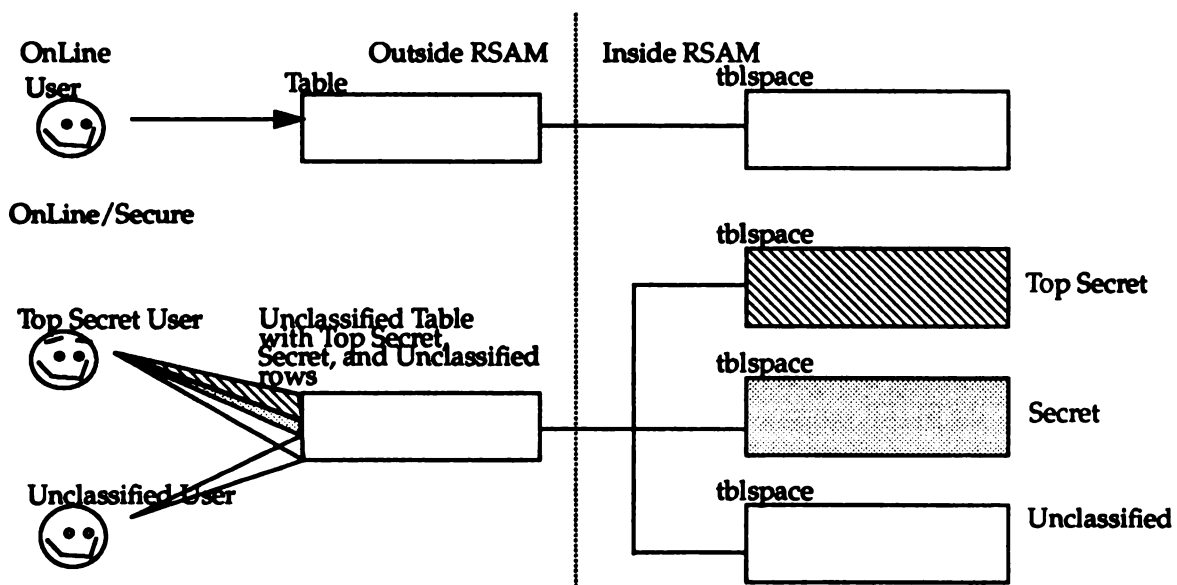


Figure 2 Tables and Table Abstractions in OnLine and INFORMIX-OnLine/Secure

AUDIT

Trusted systems must be capable of recording security-relevant events in a security audit log. In a trusted DBMS, this could imply auditing every event that takes place while the system is operational. Thus, each INFORMIX-OnLine/Secure system is able to audit all of its security relevant events and place the audit information in a security audit log. Keeping with the philosophy of not inventing any functionality that is already provided by the secure UNIX operating system, INFORMIX-OnLine/Secure audits events at the TCB boundary and sends each audit record to the operating system audit interface.

To allow the system to operate within the constraints of machine size and on-line storage capacity, INFORMIX-OnLine/Secure provides the ability to tailor which events will be audited while the system is operational.

There are five types of *audit masks* that determine which events are audited:

- **default** - The default audit mask is used if no user audit mask exists for a specific user. This implies that in the absence of any direction from the DBSSO, the events in this mask will be audited. This mask can not be removed, although it can be set to any combination of events, including removing all events from it.
- **compulsory** - The compulsory audit mask specifies events that are always audited, in addition to any other directions from the default or user audit masks. Like the default mask, this mask can not be removed, although it can be set to any combination of events, including removing all events from it.
- **user** - User audit masks are for individual users of the system who require auditing that is different from most users of the system. There can be as many user audit masks as there are users of the system, identified by login user id.
- **DBSA** - DBSA audit masks are for DBSAs. They are different from standard users of the system and apply to all DBSAs. Like the default and compulsory masks, this mask can not be removed, although it can be set to any combination of events, including removing all events from it.

- **templates** - The DBSSO may create special audit templates and store them in the audit repository. Templates may be created for specific roles or types of users in the system and then used to apply to individual users filling those roles.

The default, compulsory, and user audit masks all potentially audit the same set of events, but the DBSA audit mask covers the set of actions that are only permitted to be performed by the DBSA. The actions of the DBSSO are always audited, so there is no DBSSO audit mask.

IMPLEMENTATION

In the architecture discussion, it was noted that only the RSAM process communicates with the entities outside the TCB and implements the standard DBMS abstractions. The Support, Daemon and Transient processes are small programs designed to provide a specific functionality. Therefore, this implementation discussion focuses only on the RSAM process.

Figure 3 shows a schematic of the software layering inside the RSAM process of the Kernel. The layering, in addition to conforming with good software engineering principles, provides a convenient modularity to build the C2 and the B1/EP versions of the product.

- **SQL Engine**
The SQL Engine is used as a representative application communicating with RSAM.
- **Interface Management Layer**
This layer is the dispatcher. All entry points into the TCB are managed by this layer. For example, if a "read row" function is to be visible outside the TCB, there must be a dispatch entry for it within this layer. However, calls to "read row" made from within RSAM do not come through this layer. Instead, they go to the Audit Layer.
- **Audit Layer**
The audit layer traps every RSAM function call that can give rise to an audit event. If the audit masks indicate that the call should be audited, an audit record is generated.
- **DAC Layer**
All Discretionary Access checks are done only in this layer.
- **MAC Layer**
All Mandatory Access Checks are done only in this layer.
- **Bundle Layer**
This layer is where the translation from Bundle to tblspace is made. At this layer, all calls to a table made in the SQL Engine are translated into calls to the corresponding tblspace within the Bundle using disk and memory structures that hold state information. Once a call traverses this layer, all remaining processing is identical to what would take place within OnLine.
Not all requests processed by RSAM need to traverse this layer. For example, calls that are not related to data definition or manipulation can by-pass this layer.
- **OnLine RSAM Layers**
These are unchanged from those in OnLine.

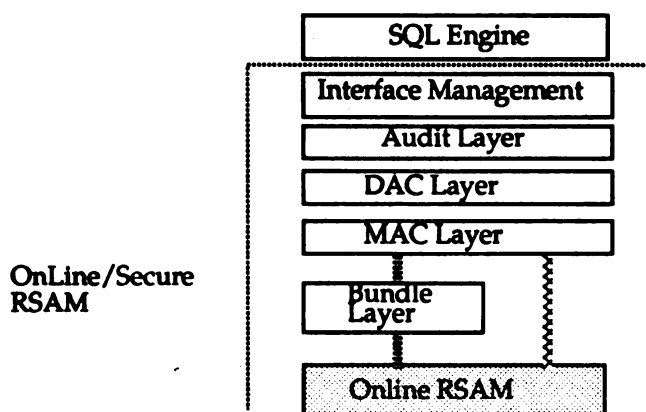


Figure 3 Interface Call Handling in INFORMIX-OnLine/Secure

CONCLUDING REMARKS

INFORMIX-OnLine has distinguished itself by high performance and advanced functionality. The architecture chosen for the INFORMIX-OnLine/Secure accounts for all requirements that arise out of the Orange Book, TDI and RAMP plan. Our architecture yields a small TCB with least privilege, thereby facilitating a speedy evaluation. The adoption of a well-thought-out development methodology and the extensive DoD standard documentation adds credibility to our high assurance claims. There is no compromise in terms of functionality; the secure product retains all the functionality of INFORMIX-OnLine. In terms of security, it supports the label space that the underlining secure operating system supports, and we believe the architecture can be migrated to B2 and B3 systems.

ACKNOWLEDGMENTS

Special thanks to Loren Anderson, Eugene Ding, Christina Fu, Leroy Lacy, Stephan Nguyen, Candace Novbakhtian, Jenny Roberson, Haiyan Song, Jack Stephens, and Aryeh Tal-Nir, without whose contributions this paper and INFORMIX-OnLine/Secure would not be successful.

REFERENCES

- [1] Department of Defense Trusted Computer System Evaluation Criteria. DOD 5200.28-STD, National Computer Security Center, December 1985.
- [2] Rating Maintenance Phase — Program Document, NCSC-TG-013, June 1989.
- [3] Trusted Database Management System Interpretation of Trusted Computer System Evaluation Criteria, NCSC-TG-021, National Computer Security Center, 1991.
- [4] DOD-STD-2167A, "Military Standard, Defense System Software Development," February 1987.
- [5] DOD-STD-2168, "Defense System Software Quality Program," Department of Defense, 1988.
- [6] DOD-STD-480A, "Configuration Control - Engineering Changes, Deviations and Waivers," U.S. Department of Defense, 1978.
- [7] A Guide to Understanding Configuration Management in Trusted Systems, NCSC-TG-006, March 1988.
- [8] A Guide to Understanding Audit in Trusted Systems, NCSC-TG-001, June 1988.

Peeling the Viral Onion

Russell Davis
PRC, Inc.
Suite 850
600 Maryland Avenue, SW
Washington, D.C. 20546

(202) 453-9021
rdavis@ames.arc.nasa.gov

Abstract

This paper boils down much of the existing virus research into a succinct set of inference rules. These rules are then expanded to include the newer self encrypting stealth viruses along with the necessary conditions for their detection. This foundation is then applied to derive additional properties of new viruses.

Forward

One problem with any virus control is in isolating the control from the virus. To overcome the issues associated with protecting the virus detector, the discussion will assume an isolated platform such as the Security Pipeline Interface (SPI) [9].

An expert system includes facts and rules which when applied together can infer new facts. Additionally, an expert system should be able to explain how a conclusion was achieved. This paper describes 12 general rules and includes predicate calculus representation. An expert system using the rules stated will most likely require external programs to calculate functions such as encryption and checksums.

Previous Work

Computer systems are exposed to a variety of security threats. Of the threats many are known while others will manifest themselves as time passes. To cope with the ever changing security environment there has been promising work done in the area of real time expert systems which have demonstrated the ability to detect computer system intrusions.

The National Computer Security Center (NCSC) has installed the 'Multics Intrusion Detection and Alerting System (MIDAS) on their DOCKMASTER network [18]. Other promising work includes the Intrusion Detection Expert System (IDES) prototype at SRI International [12]. In these examples, the expert system is located on an isolated platform (a Sun Workstation for IDES and Symbolics for MIDAS). This paper will examine possible extensions to intrusion detection systems which will identify computer viruses.

One area of interest is how to detect viruses and upon their detection, how to recover. Computer viruses became publicized [2] as a security threat in 1984. Since Cohen's paper, much research has gone into finding ways to combat viruses.

Platform Description

Detecting a virus infection, subsequent to starting with a known good product can be accomplished through the use of a weak or strong cryptographic checksum such as those described in [17] and [4]. Checksum identifiers, cryptographic or otherwise, run the risk of themselves being infected. The methods to assure checksum generator integrity include implementing the algorithm in ROM or by partitioning the function from the main system. For any rule based virus control to be effective, it must be insulated from the direct effects of a virus. One architecture which isolates the virus control software from the potentially infected host environment is SPI [9].

The SPI architecture is essentially a physical pipeline of processors configured inline with the I/O paths. The SPI pipeline processor affords an opportunity to isolate any detecting algorithm from the host or DBMS in use. In a SPI configuration, the pipeline processor will have in-line connectivity to a backup store.

This store contains a copy of the distributed software for the purpose of comparison. No assumptions are made with respect to the cleanness of the files originally placed on the backup store. The only restriction is that the backup store is not directly assessable to the host environment. Adleman [1] implies that viruses are no threat if new programs can't be introduced, old programs never change, and communications are not allowed. The isolated SPI architecture satisfies each of these three conditions.

General Rules

This section develops general computer virus inference rules. The common security threat to executables posed by viruses is loss of integrity¹. One introductory point made in [23] is that a virus carrier is usually unrelated to the program it infects.

Rule 1: An executable will change following a virus infection.

$$\text{executable}(\text{file}) \wedge \text{infection}(\text{file}) \Rightarrow \neg \text{integrity}(\text{file})$$

An executable is some set of machine readable instructions such as a program file or some binding mechanism. A virus alters a program by copying itself into programs or files [22]. The central focus of this paper is to provide an analysis of file corruption caused by computer viruses. One point made by Spafford [19] is that "viruses cannot spread by infecting pure data." Pure data in this context does not include source code nor other data which influence a computer's control execution. That is, for a virus to propagate, it must influence instructions executed by the CPU at some point. In general, data files are not executable.

Rule 2: A changed file can be identified through the use of a checksum function.

$$\begin{aligned} \text{checksum}(\text{file}) &\Rightarrow \text{integrity}(\text{file}) \\ \neg \text{checksum}(\text{file}) &\Rightarrow \neg \text{integrity}(\text{file}) \end{aligned}$$

There are many types of checksum functions. Some are based on Cyclic Redundancy Checks (CRC) or cryptographic algorithms. One example of a cryptographic checksum is described by Pfleeger [16]. Pfleeger points out that if the computed checksum matches the stored value then it is likely that the file has not been changed. That is, changes to files result in changes to the computed checksum value. As indicated in [20], a 16-bit checksum such as the CRC-16, detects 99.998% of all 18-bit and longer burst errors. It should be noted that if a CRC algorithm is known it can be defeated. To overcome a known CRC attack, an isolated platform such as SPI can be used to randomly select the CRC algorithm used and thereby immunize itself from a CRC attacker [7]. A certainty factor² based on the strength of the checksum function should be considered when using rule 2.

Rule 3: For a virus to function, it must influence machine readable instructions on the host computer.

$$\text{executable}(\text{file}) \wedge \text{infected}(\text{file}) \Rightarrow \text{virus}(\text{file}, \text{active})$$

-
1. In this paper, loss of integrity implies unauthorized modification (including destruction).
 2. The certainty factor is a measure which approaches 1.0 as the evidence for a given hypothesis increases.

Rule 3 provides the key for detecting self encrypting viruses. A self encrypting virus is designed to defeat the prefix and postfix checker controls such as those described in [24]. A virus incorporating this stealth technique introduces a different pattern for each file infected. The common denominator is that the host computer must be able to decrypt the virus as it executes the infected file. If this were not true then the infection could not execute and thereby not propagate itself.

Rule 4: Given an original file and a corrupted version of the original, there exists a function DIFF that returns the changes made to the original file which, when applied to the original file, result in the corrupted file.

$$\text{original}(\text{file}) \wedge \text{altered}(\text{file}) \Rightarrow \text{diff}(\text{pattern})$$

The confidence that the proper diff pattern has been obtained increases when the identical pattern is observed in several corrupted files.

Rule 5: Given an encrypted pattern containing an encrypting virus the decrypted code can be obtained by incrementally applying Rule 3.

$$\text{diff}(\text{pattern}) \wedge \text{applied_to}(\text{first_instruction}, \text{pattern}) \wedge \text{executable}(\text{pattern}) \Rightarrow \text{algorithm}(\text{decryption})$$

The function "applied to" uses the first executable instructions obtained from the infection to operate on the encrypted file. The initial infection instructions must provide the method for restoring the executable virus instructions. In a typical stealth virus which encrypts itself, some pattern (key stream) is usually added, using modulo 2 addition, to each byte of the virus code. By using a randomly generated pattern during the initial infection, each virus infection pattern appears different. A multi-encrypted file would also be recoverable by recursively applying Rule 5. That is, n decryption passes are required in order to obtain the decryption information necessary for the $n + 1$ th pass. In general, if there are m encryption passes used in the stealth virus, then rule 5 would have to be incrementally applied m times.

Rule 6: It is possible, through the use of a disassembler, to disassemble an executable file.

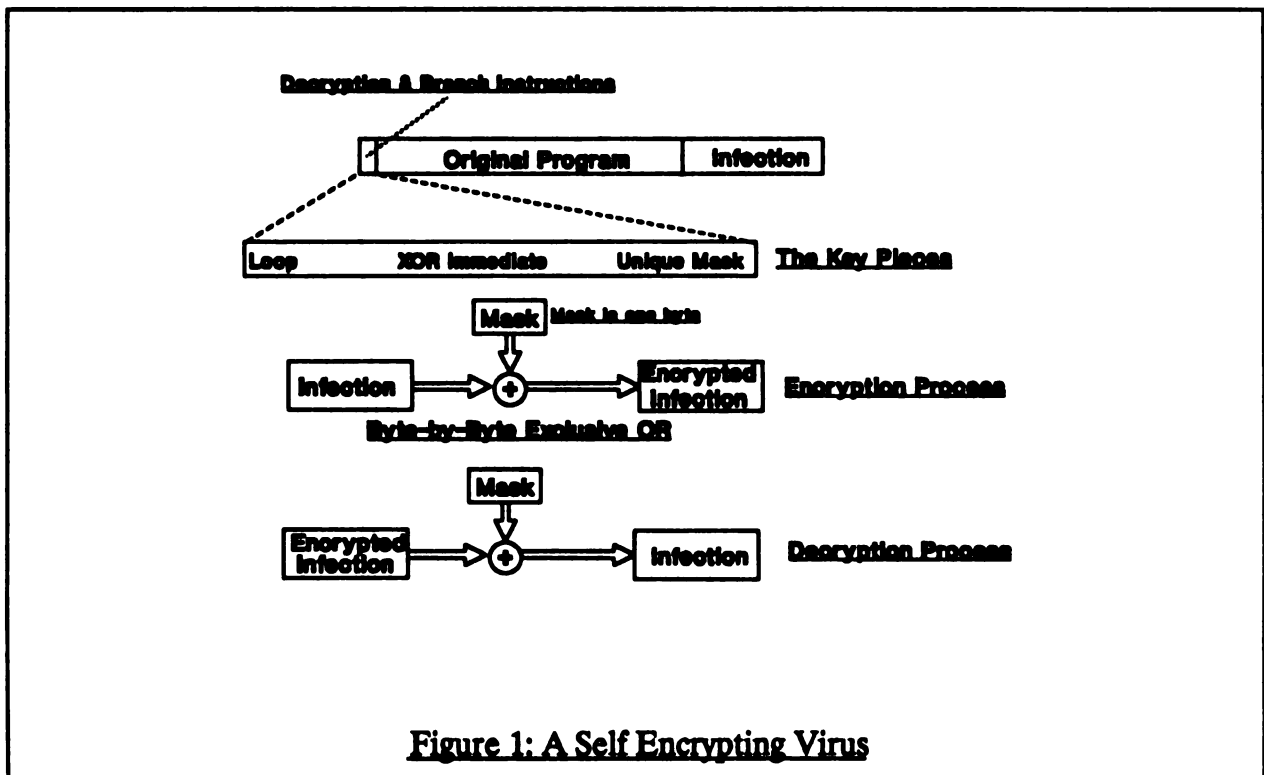
$$\text{executable}(\text{file}) \wedge \text{disassemble}(\text{file}) \Rightarrow \text{assembly}(\text{file})$$

In this discussion we use a goal-driven search for viruses. Moreover, the disassembled code has certain exploitable characteristics. We know where to begin disassembly (the start of the diff file). Additionally, a well formed executable program should be parsable into a assembly listing. Today, many debug utilities include an disassemble capability. This rule points out that if the corruption applied to a file is executable then it should be possible to disassemble.

Rule 7: An encrypted file cannot be correctly disassembled

$\text{encrypted}(\text{file}) \Rightarrow \neg \text{machine_readable}(\text{file})$
 $\neg \text{machine_readable}(\text{file}) \wedge \text{disassemble}(\text{file}) \Rightarrow \neg \text{assembly}(\text{file})$

Encrypted data is an unintelligible form called cipher [14]. If a file is encrypted then it is unintelligible and hence cannot be correctly disassembled. That is, an encrypted file must be processed (decrypted) prior to, or as part of, execution. It is possible that an encrypted file will disassemble into something syntactically acceptable but semantically meaningless. This property also applies to data files. Indeed, the file might contain data which would halt the processor.



Rule 8: An encrypted file can be disassembled after applying Rule 5.

$\text{encrypted}(\text{file}) \wedge \text{applied_to}(\text{first_instruction}, \text{file}) \wedge \text{disassemble}(\text{file}) \Rightarrow \text{assembly}(\text{file})$

The function performed in Rule 5 decrypts the cipher thereby restoring the code to a machine readable format. The resulting machine readable code can then be disassembled by applying Rule 6.

Rule 9: A known and unencrypted virus can be located if it resides in an executable.

$\neg \text{encrypted}(\text{pattern}) \wedge \text{known_as}(\text{pattern}, \text{name}) \Rightarrow \text{virus}(\text{name})$

A simple pattern matching function is sufficient to satisfy Rule 9. Many of the existing virus detecting programs search files looking for patterns representing viruses. The function "known_as" is a table look-up of known viruses.

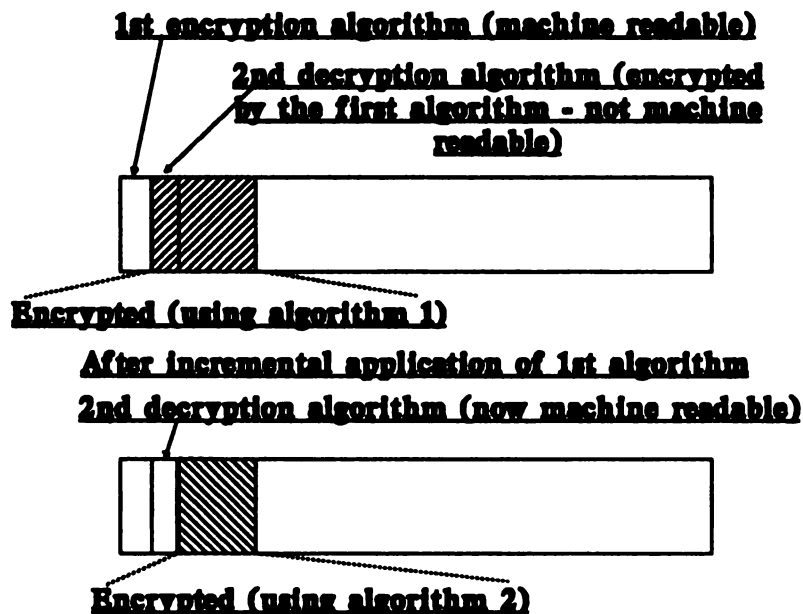


Figure 2: Double Encrypted Virus

Rule 10: If a binary difference from DIFF disassembles, the likelihood of a random error is low.

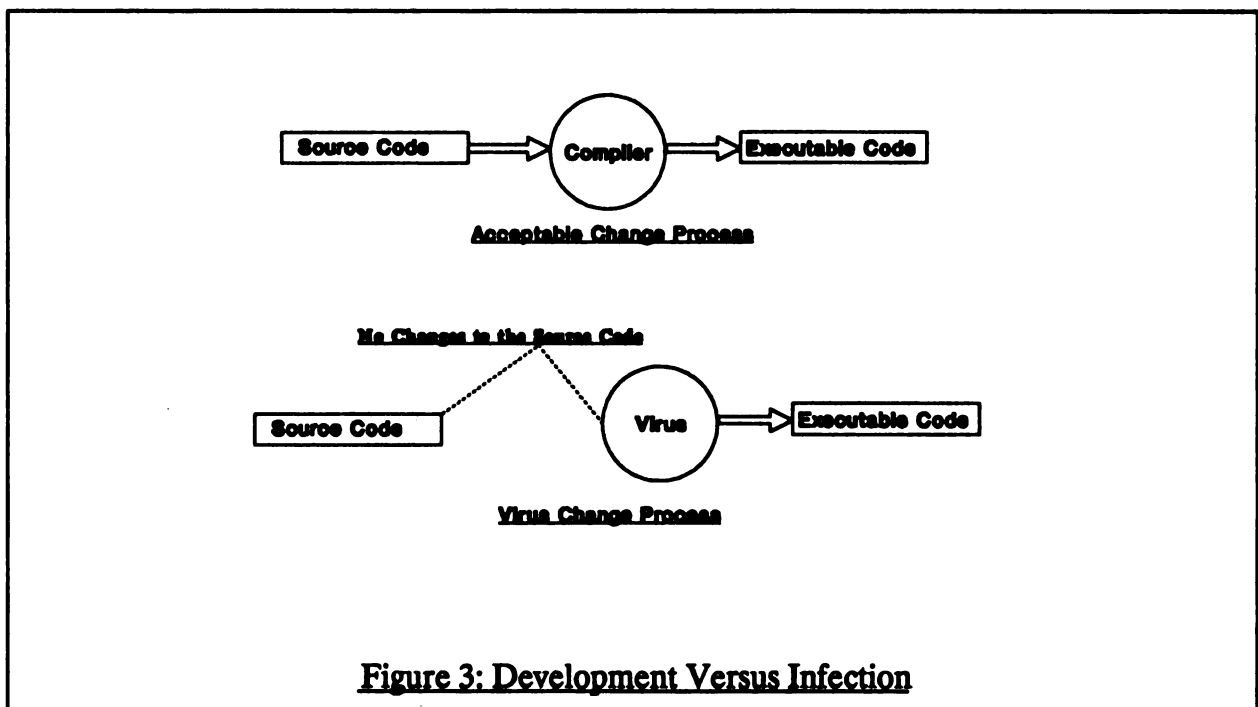
$\text{assemble}(\text{file}) \Rightarrow \text{file}(\text{executable})$

It is remotely possible to disassemble a random file and get legitimate code, however there are sufficient invalid states to make this unlikely. Rule 10 points out that in a random corruption of a file, the probability of the corrupted difference being executable is low. Within a given CPU instruction set there are many illegal states. A random corruption would most likely result in many non-valid instructions, any of which would result in an error state.

Figure 1 and 2 illustrate a stealth encryption virus. In this simple example, the circle represents a process performing the exclusive-OR function of a MASK byte to each byte of the infection. In a more sophisticated encryption scheme, the MASK could be obtained from a key stream such that each byte-wise XOR would be with a pseudo-randomly derived MASK. The random outputs would be exclusive OR'd to each byte resulting in a stronger encryption scheme.

Software Development Rules

In a development environment changes are made to source code and then recompiled. Thus legitimate changes to executable code should follow changes to source code. If the development tools and source code remain unchanged while the executable changes, then the changed executable is probably not legitimate as shown in Figure 3.



Rule 11: If an executable program changes but the source code does not then the changed executable is probably not legitimate

$$\text{configuration(unchanged)} \wedge \text{integrity(source, file)} \wedge \neg \text{integrity(executable, file)} \Rightarrow \neg \text{legitimate(executable, file)}$$

If nothing changes, then compiling the same source code should result in identical executables.

Rule 12: Compiling revised source code produces revised executable code.

$\neg \text{integrity}(\text{source}, \text{file}) \wedge \text{compile}(\text{source}, \text{file}) \Rightarrow \neg \text{integrity}(\text{executable}, \text{file})$

An interesting point made by Page [15] is the possibility of source code viruses. Given the C compiler Trojan horse example described by Thompson [21], it is not unreasonable to visualize a source code virus. To see how a source code virus might work, consider the following. By infecting only source code, it would be difficult for many of the current "executable" detectors to monitor systems. Optimizing compilers often restructure code such that the executable files might not have a discernable signature. A source infector would

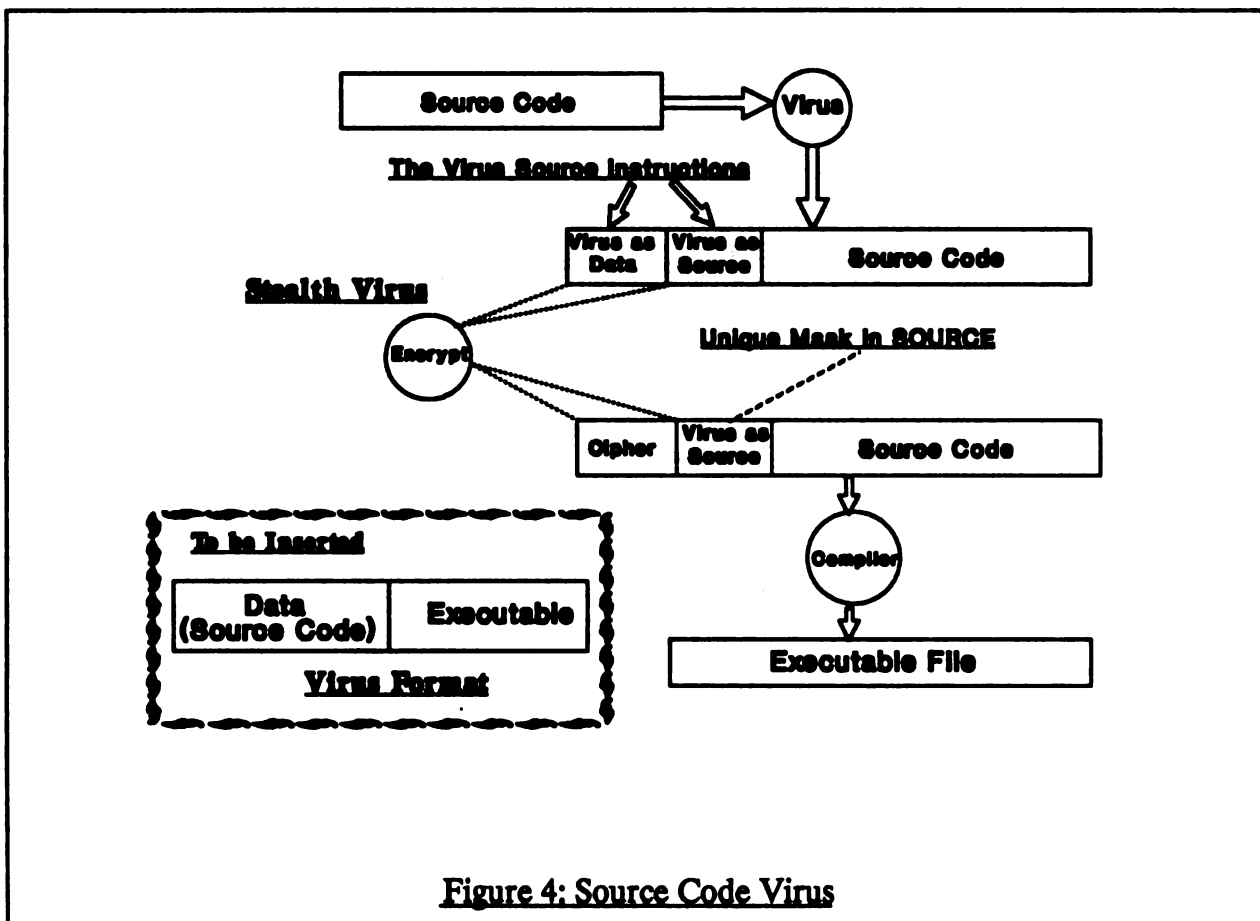


Figure 4: Source Code Virus

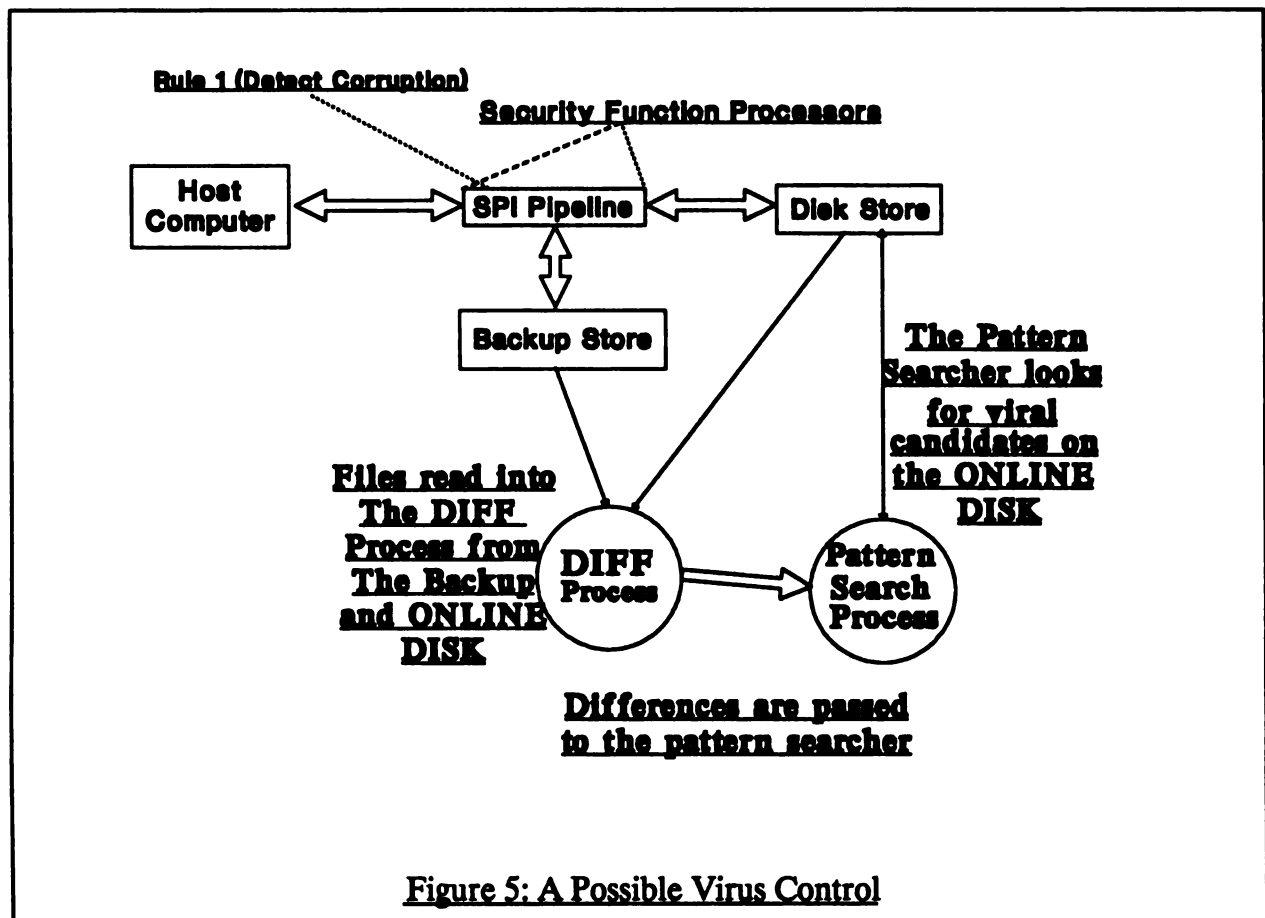
require several pieces including source code readable by a translator (compiler). The actual infection could insert two copies of itself into the source code. The first copy might be declared as a text array. The second copy would be destined for in-line insertion thereby becoming executable after compilation. Further, a stealth virus might encrypt the text array making executable pattern recognition more difficult as shown in Figure 4. A source code virus might be detected by comparing infected source code files to reveal identical in-line instructions representing the virus.

A typical source code infector might look like the virus format shown in Figure 4. In this example, the virus contains executable code and a text buffer containing compiler readable source code. Everything is self contained within the infection. After compilation, the resulting file contains both source and executable code. Clearly, a source code virus is feasible.

Applying the Rules

By applying the above stated rules a disassembled copy of the viral infection can be extracted. This section describes the procedure and then address what can be learned from the virus code. The specific rule addressed will be abbreviated. For example, Rule 1 will be denoted (R1).

Cohen has shown that in general, it is undecidable whether or not a sequence of code is a virus [3]. Furthermore, other researchers agree with Cohen's proof and have proposed refinements to his proof model [10]. By contrast, Crocker and Pozzo [5] proposed a "fail-safe" virus filter. Ducking the religious issues associated with these two extreme positions, there are some pieces of information which are decidable in polynomial time. For example, if we have a known virus such as nVIR we can conclude that a file is infected if we find nVIR in an executable. This example holds for the special case of a known virus, but not in general.



One use of assembly code is to search for illicit code as described in [8]. The assumption is that viral code has some identifiable features which differentiate it from normal instructions. A similar approach focusing on viral operating system calls was proposed in [11]. For example, in the 80x86 CPU, instructions such as IN, OUT, or INT might be cause for concern.

When a new virus hits there is time lost in figuring out what the virus does. Typical inquires request information on triggers and payloads. Much of the desired information can be obtained from a parse performed within an isolated processor. By applying a disassembly to the executable program and then searching for viral instructions, much information can be accumulated. An example environment using the SPI architecture is shown in Figure 5.

Using the SPI architecture described in [9], changes to executable files can be detected using (R2). From the altered file, the difference can be extracted by (R4). The extracted difference can be decrypted if

encrypted (R5) and then disassembled (R6). If the disassembly succeeds (R10) then there is a good indication that the file has been deliberately modified. If the unencrypted difference (R5) appears in multiple files then it is likely that a virus is at work.

Figure 5 also illustrates a possible mitigative control based on transparently restoring corrupted files from a backup disk. In this example, the backup store files are used for comparisons and are not executed on the SPI processor. Therefore, an infected file residing on the backup store could not infect the SPI processor. As a final point, the backup store could be updated using an approach similar to [13] where the user is queried.

Derivable Cases

Consider the shrink-wrap virus case. The virus would first manifest itself by executing its payload or by reproducing. If the virus is still in the reproductive stage, then it will most likely be detected in another file after that file's integrity is altered due to infection. Through the application of the rules previously stated, the newly infected file will provide a source for extracting the viral code. A pattern matcher can then explore all executable files for an occurrence of the same viral set. Should the pattern be found in an original distributed file then we can infer that the source is a shrink-wrap virus.

Summary

This paper examines inference rules involved in identifying a computer virus. The newer self encrypting stealth viruses are examined and along with the necessary conditions for their detection. The rules are then used to derive properties of new viruses.

Acknowledgements

I would like to thank Cheryl Ledbetter, Lee Rice, and the reviewers for their objective comments and recommendations.

References:

- [1] Leonard M. Adleman, "An Abstract Theory of Computer Viruses", *Lecture Notes in Computer Science* Vol. 403, *Advances in Computing - Crypto '88*, S. Goldwasser (ed.), Springer-Verlag, 1990.
- [2] Fred Cohen, "Computer Viruses", *Proceedings of the 7th DOD/NBS Computer Security Conference*, pp. 240-263.
- [3] Fred Cohen, "Computer Viruses", pp. 23-27, Copyright 1985 by Fred Cohen.
- [4] Fred Cohen, "A Cryptographic Checksum for Integrity Protection", *IFIP Computers and Security* 6(6), 1987.
- [5] Steve Crocker & Maria Pozzo, "A Proposal for a Verification Virus Filter", *Proceedings of the 1989 IEEE Computer Society Symposium on Security and Privacy*, pp. 319-324.
- [6] Russell Davis, "Exploring Computer Viruses", *Proceedings of the Fourth Aerospace Computer Security Applications Conference*, pp. 7-11.
- [7] Russell Davis, "Uncovering Viruses", *Proceedings: Fourth Annual Computer Virus & Security Conference*, pp. 796-803, March 14-15, 1991.
- [8] Garnett, "Selective Disassembly: A First Step Towards Developing a Virus Filter", *Proceedings of the Fourth Aerospace Computer Security Applications Conference*, pp. 2-6.
- [9] Lance J. Hoffman, et al., "Security Pipeline Interface (SPI)", *Proceedings of the Sixth Annual Computer Security Applications Conference*, December, 1990.
- [10] Kimmo Kauranen, et. al., "A Note on Cohen's Formal Model for Computer Viruses", *Special Interest Group - Security, Audit & Control Review*, Volume 8, Number 2, pp. 40-43, ACM Press.
- [11] Paul Kerchen, et al., "Static Analysis Virus Detection Tools For UNIX Systems", *Proceedings of the 13th National Computer Security Conference*, pp. 350-365.
- [12] Teresa F. Lunt, "Automated Audit Trail Analysis and Intrusion Detection: A Survey", *Proceedings of the 11th National Computer Security Conference*, October 1988.

- [13] James Molini and Chris Ruhl, "The Virus Intervention and Control Experiment", *Proceedings of the 13th National Computer Security Conference*, pp. 366-373.
- [14] "The Data Encryption Standard", *FIPS PUB 46*, National Bureau of Standards (Now NIST).
- [15] John Page, "An Assured Pipeline Integrity Scheme for Virus Protection", *Proceedings of the 12th National Computer Security Conference*, pp. 378-388.
- [16] Charles P. Pfleeger, "Security in Computing", copyright 1989 by Printice-Hall, Inc., pp. 160-161.
- [17] Pozzo and Gray, "An Approach to Containing Computer Viruses", *IFIP Computers and Security*, 6(4), 1987.
- [18] Michael M. Sebrint et. al., "Expert Systems in Intrusion Detection: A Case Study", *Proceedings of the 11th National Computer Security Conference*, October 1988.
- [19] Eugene H. Spafford, et al., "Computer Viruses: Dealing With Electronic Vandalism and Programmed Threats", *ADAPSO*.
- [20] Andrew S. Tanenbaum, "Computer Networks", Copyright 1981 by Prentice Hall, pp. 128-132.
- [21] Ken Thompson, "Reflections on Trusting Trust", *Communications of the ACM*, Vol. 27, No. 8.
- [22] Steve R. White, et al., "Coping with Computer Viruses and Related Problems", copyright 1989 by International Business Machines Corporation.
- [23] David R. Wichers, et. al., "PACL's: An Access Control List Approach to Anti-Viral Security", *Proceedings of the 13th National Computer Security Conference*, pp. 340-349.
- [24] Catherine L. Young, "Taxonomy of Computer Virus Defense Mechanisms", *Proceedings of the 10th National Computer Security Conference*, pp. 220-225.

PRACTICAL MODELS FOR THREAT/RISK ANALYSIS

Mark W.L. Dennison
Kalman C. Toth

CGI Information Systems & Management Consultants Inc.
275 Slater Street, 19th Floor
Ottawa, Ontario, Canada, K1P 5H9
Tel: (613) 234-2155
Fax: (613) 234-6934

ABSTRACT

This paper describes practical models used to conduct threat and risk analyses for large information systems or networks. The process of analyzing and relating threat agents, assets, and safeguards within a static threat model is described. In addition, a dynamic risk model is described that consists of threat events, security failures, and damaging outcomes. This approach permits the incorporation of known baseline security requirements such as policies and standards, as well as hardware and software security features that could be distributed throughout the information system.

INTRODUCTION

The aim of this paper is to describe new threat and risk models, together with a methodology that can be employed to support a practical risk assessment of computer systems and networks. The model and methodology are applicable in the government environment, while retaining essential compliance with previous work done in the threat/risk arena. This paper expands upon the previous work of Toth, Dennison, and Clayton [Toth 91] by adding more details pertaining to the structure of the threat and risk models.

A threat and risk analysis can be used in various ways to achieve different objectives. It can be used to assess the risks of operating an existing system within a given operating environment and mode. The analysis can determine if additional safeguards or alternative operating modes could contain the risks in an existing system. Threat/risk analysis can also evaluate technology alternatives and provide recommendations for designing a new system given environmental, operational, and budgetary constraints.

Our threat/risk analysis approach is based on models that characterize and relate the threat and risk elements of the system, and a methodology that plans, organizes, and guides the analysis to produce meaningful assessments of the threats and risks associated with the system. The methodology provides guidance for creating instances of the models and for conducting the risk analysis.

The threat/risk model has been broken down into two submodels, the threat model and the risk model. These models contain common entities, namely, "threat agents", "assets", and "safeguards", but address these aspects from different points of view. The threat model deals with identifying the entities that comprise these components and determines their attributes and relationships. The risk

model identifies and relates threat events that cause security failures which in turn produce damaging outcomes. It also includes an additional component for calculating risks.

OBJECTIVES

The threat/risk models have been designed to be applicable in a government environment, to have an information technology orientation, and to be compatible with the framework defined in the Strawman Model [Katzke 85].

Government Applicability

Government organizations are mandated to conduct threat and risk assessments in relation to classified or sensitive information and other assets. This has generated considerable interest in practical models and methods that can be used to conduct assessments. Most security damage in the public sector does not have a direct monetary effect. Methodologies that attempt to put a dollar value on all types of damage are not appropriate in many government and industry environments. This issue has been addressed by allowing security damage to be defined in terms of non-monetary outcomes.

Many risk methodologies do not address information technology security standards which exist in the government. These methodologies often assume that all options are open and that no baseline requirements exist. The proposed threat/risk model addresses this problem by allowing certain security policies and standards to be included in the "safeguards" portion of the model. This ensures that minimum standards are met.

Information Technology Orientation

Many risk assessment methodologies fail to account for the hardware and software platform used by the organization. These methodologies often assume that there is no existing system to consider. This issue has been addressed by introducing "safeguards" as a component of the model to characterize the security mechanisms implemented by the existing or proposed system. This emphasis on safeguards follows the risk management framework proposed by [Katzke 85], subsequently evolved in the risk management workshops [Workshop 88] and [Workshop 89]. The model allows for distributed safeguards in order to address the distributed nature of today's information systems.

The threat/risk model addresses the organization and its system platform by limiting its scope to only those entities and events that are applicable. It addresses the organization's system platform by analyzing the security mechanisms of existing or proposed products, systems, and procedures. It also identifies and assesses the cost exposures and non-cost effects (confidentiality, integrity, and availability) of target assets within the existing system due to security failures.

Strawman Framework

The threat/risk models use terms that are familiar to system users, senior management, and other threat/risk model builders. Much of the standard terminology of the Strawman Model is utilized with some extensions. The various elements of the threat model are referred to as "entities" and each entity can have descriptive "attributes." The elements of the risk model are "events," which can also have "attributes." There are also functional relationships amongst entities and attributes.

The threat/risk model that has been proposed can be used for future applications, system environment changes, or security mechanism upgrades. To update the model, the specific tasks defined in the methodology are repeated taking into account the system changes. The iterative nature of the model ensures that threat/risk assessments can be refined or revisited throughout the life-cycle of an operational system. Furthermore, the model supports trade-off analysis among system alternatives and comparative cost-benefit analysis.

THREAT/RISK ANALYSIS METHODOLOGY

The threat/risk analysis methodology is a comprehensive and structured approach for analyzing the threats and risks of computer systems and networks. The methodology includes phases for preparation, threat assessment, risk assessment, and recommendations.

Preparation Phase

The preparation phase is required to plan the strategy for the threat/risk assessment. The choices made will reflect the size of the system, the value and sensitivity of assets, and the nature of the perceived threats. The plan should clearly identify the scope of the system and the level of detail required in the threat/risk analysis. It also must be decided if proposed safeguards will be included within the analysis.

The organizing step is required to identify all of the inputs to the threat assessment. Some inputs, such as threat descriptions, may come from external sources while other inputs may exist internally. The preparation phase identifies inputs such as the system security policy, statement of sensitivity, mode of operation definition, contingency plans, and disaster recovery plans. A checklist is used to ensure that no inputs are forgotten. When a required input cannot be found, steps must be taken to produce that input. Questionnaires and interviews can be used to collect missing information.

Threat Assessment Phase

The threat assessment phase puts information into the threat model by specifying entities, entity attributes, and entity relationships. An instance of the threat model is thereby obtained that defines the threat agents, safeguards, and assets pertaining to the information system under consideration. It should be noted that the term "threat model" is used due to its general acceptance even though it includes descriptions for assets and safeguards. The scope of the threat analysis is controlled by the scope of the information system under consideration as defined in the preparation phase.

The first part of the threat assessment is to load the asset component of the threat model. This documents all of the assets within the system boundary and defines attribute values. The second part is to load the threat component, which includes definitions of all threat agents and their attributes. The third part is to load the safeguard component, which includes a definition of all safeguards and their attributes.

The threat assessment implicitly includes a vulnerability assessment. Vulnerability is included as an attribute of assets and safeguards. Asset vulnerabilities are estimated by considering attributes of threat agents (such as intention) and attributes of the asset (such as value). Safeguard vulnerabilities are estimated by considering attributes of threat agents (such as potency) and attributes

of the safeguard (such as effectiveness). Vulnerability analysis is heuristic in nature and is often performed based upon past experience. In our interpretation, vulnerability analysis is a qualitative estimate of risk in the absence of detailed knowledge about events and event probabilities.

The fourth part of the threat assessment is to link the entities of the model via functional relationships. This shows which safeguards protect which assets, as well as which threat agents target which assets. In addition, the relationships show that some entities are composed of other entities. For example, a high-level system safeguard may be composed of subsystem safeguards.

Risk Assessment Phase

The risk assessment phase of the methodology puts information into the risk model by specifying events, event attributes and event relationships. An instance of the risk model is thereby created that defines the possible threat events, security failure events, and damaging outcome events. Probabilities are introduced as attributes of events in the risk model to deal with the likelihood of these events. Probabilities are a measure of the expectation that a particular event will occur. It is suggested that the selected time period be one year. Information from the threat model is used to help compute the probabilities of these events occurring.

While threat assessment is qualitative, risk assessment is mainly quantitative. Risk is calculated as the expectation of a given level of damage over a given period of time resulting from threat events that cause security failures producing damaging outcomes. The risk calculation, therefore, includes both the probabilities of events and the severity of the damage to assets. Since risk assessment is more rigorous than threat assessment, it may be decided to only perform risk assessment for parts of the system.

Trade-off analysis is used to study risk by varying threats, assets, and safeguards. Various alternatives are analyzed, although the scope of possible adjustments is system-specific, and trade-offs among alternative safeguard solution sets are examined. It is often difficult to reduce threats as such, although there is considerable flexibility when designing a new system. Asset exposure can often be changed by altering the scope of the computer system or network. An organization usually has control over safeguard selection, although these choices are usually subjected to a cost-benefit analysis. The analysis phase continues until the maximum risk acceptability is obtained.

The methodology permits iteration at various levels. A typical approach might be to establish a baseline from mandated requirements (safeguards) and initial identification and assessments of assets and threats. Initial risk based trade-offs and cost-benefits may be analyzed as various safeguard sets and strengths are evaluated. Later on in the project life-cycle, as information becomes more accurate and new safeguard alternatives arise, the threat and risk assessments and analysis work could be redone to produce more refined recommendations.

Recommendations Phase

The recommendations phase is intended to document the results of the threat/risk analysis and to provide an overall statement of risk. The document is often directed towards senior management and presents alternatives, options, and recommendations for action. The recommendations may suggest adding additional security mechanisms, altering the assets in the system, or reducing threats. If recommendations for change are accepted, the threat/risk methodology can be used in an iterative manner to develop a new risk assessment.

THREAT MODEL

To fully understand the threat assessment phase, it is necessary to examine the details of the threat model. The entities of the threat model have primary attributes and sometimes secondary attributes, which provide even more detailed information. The threat model consists of threat agent entities, asset entities, safeguard entities, and functional relationships.

Threat Agent Entities

A threat agent T_i is an entity (e.g. person, organization, or thing) that desires to or is able to trigger an event that can compromise the security of an asset. The attribute threat agent identifier or $T_i(\text{ident})$ is the unique information identifying each threat agent. Secondary attributes indicating threat agent type (i.e. natural or human), geographic location, and historical behaviour may also be specified.

The attribute threat agent potency or $T_i(\text{pot})$ is an aggregate expression of the potential of a threat agent and indicates what the threat agent can do. The secondary attribute threat agent capability or $T_i(\text{cap})$ indicates the individual ability of a threat agent to act, or to be effective. The secondary attribute threat agent resources or $T_i(\text{res})$ indicates the resources at the disposal of the threat agent. These attributes are relatively independent of the safeguard and asset components of the model.

The attribute threat agent intention or $T_i(\text{intent})$ indicates the threat activities that the threat agent is liable to mount and indicates what the threat agent will do. It has secondary attributes of motive and determination. The secondary attribute threat agent motive or $T_i(\text{mot})$ indicates which assets or safeguards the threat agent is likely to attack, and is a function of the threat agent's perception of asset vulnerability and value. The secondary attribute threat agent determination or $T_i(\text{det})$ indicates the degree to which the threat agent will pursue the desired asset.

Asset Entities

An asset A_i is an entity that is of value to an organization. It is assumed that the asset's value is sufficient to warrant concern for the asset's protection. The attribute asset identifier or $A_i(\text{ident})$ is the unique information identifying each asset. The attribute asset role or $A_i(\text{role})$ indicates if an asset exists for its own purpose, or whether it is a secondary asset providing support or protection functions. The attribute asset ownership or $A_i(\text{own})$ indicates the individual or organization who owns the asset.

The attribute asset type or $A_i(\text{type})$ indicates if the asset is tangible or intangible (i.e physical asset or information asset). Asset type can take on values such as classified information, sensitive information, classified asset, or sensitive asset. The attribute asset grouping or $A_i(\text{grp})$ can take on a value from the following set: physical, environmental, information, hardware, software, communications, operations, human resources, and documentation. There could be other secondary attributes specifying geographic location, system or subsystem name, and whether the asset is internal or external to the information system being analyzed.

The attribute asset value or $A_i(\text{val})$ indicates the value of the asset to the organization. Asset values are measured against the secondary attributes of sensitivity (for confidentiality), criticality (for integrity and availability), and replacement value (for monetary value). The attribute asset vulnerability or $A_i(\text{vuln})$ indicates the degree to which successful attacks might be launched against

this particular asset. This attribute is functionally related to threat agent entities, safeguard entities, and other attributes of the asset. For example, asset vulnerability may be related to accessibility, complexity, user friendliness, fragility, etc. Vulnerability is considered a static property of assets, and is supplemented by a more detailed risk analysis of outcome events in the risk model.

Safeguard Entities

A safeguard S_i is a mechanism that protects assets from threat agents by thwarting threat activities. Safeguards must be included in the threat model to account for the complexities of the information system. Since safeguards are assets, they inherit all asset attributes. In addition, each safeguard entity has attributes relating to the protection mechanism that it provides. If safeguards are interpreted as all mechanisms providing confidentiality, integrity, and availability then all functional components of an information system would be considered as safeguards.

The attribute **safeguard type** or $S_i(\text{type})$ indicates whether the safeguard is active or passive. The attribute **security function** or $S_i(\text{func})$ indicates the general purpose and function of the safeguard and can take on values from the set of deterrence, prevention, detection, containment/mitigation, and recovery. The attribute **safeguard cost** or $S_i(\text{cost})$ shows the cost of implementing the safeguard, and is required for performing a cost/benefit analysis.

The attribute **safeguard effectiveness** or $S_i(\text{eff})$ indicates how well the safeguard is able to perform a protection function. The attribute **safeguard vulnerability** or $S_i(\text{vuln})$ indicates the degree to which the functionality of the safeguard could be compromised. This functional vulnerability is the dual of safeguard effectiveness. Note that safeguards are also vulnerable as assets, which involves a broader assessment including threat agents and other assets.

The attribute **safeguard requirement** or $S_i(\text{req})$ indicates if the safeguard is required (mandated) as a baseline security measure by a government security policy or standard. Safeguard requirements are functionally related to the value of an asset. Safeguard requirements map to government standards for physical, procedural, personnel, and information technology security.

Functional Relationships

Entities are the essential elements of the threat model and correspond to real objects in the system that are of consequence to the threat analysis. The entities and relationships of a sample instance of the threat model are illustrated in Figure 1.

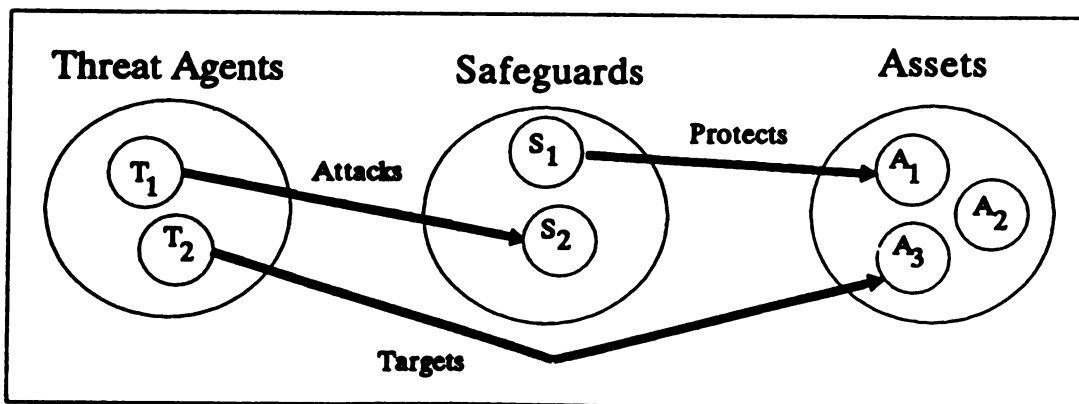


Figure 1: Sample Instance of the Threat Model

The relationships in Figure 1 show which assets are targeted by which threat agents, which assets are protected by which safeguards, and which safeguards provide protection against which threat agents. Other relationships could have been added to show a component hierarchy of safeguards and assets or to show how safeguards and assets are distributed.

RISK MODEL

To fully understand the risk assessment phase, it is necessary to examine the details of the risk model. Events are the key elements of the risk model. An attribute is a property of an event that provides additional information about the event. Note that events are related to the entities of the threat model, and to the attributes of these entities. The risk model consists of threat events, failure events, outcome events, and event relationships.

Threat Events

A threat event E_i is an attack against the system caused by a threat agent that has the potential to compromise the security (i.e. monetary or non-monetary loss) of an asset. A threat event is only an attempt to compromise security, and may be blocked by security safeguards before any damage is done.

The attribute threat event identifier or $E_i(\text{ident})$ is the unique information identifying each threat event and relates the event to a set of threat agents. There are secondary attributes relating to geographic location, system or subsystem identifiers, asset(s) attacked, etc. The secondary attribute threat event activity or $E_i(\text{act})$ indicates whether the event is related to espionage, sabotage, subversion, terrorism, criminal acts, accidents, or natural hazards.

The attribute threat event frequency or $E_i(\text{freq})$ is the expected rate of occurrence of the threat event and must be converted to an annual frequency. The frequency of occurrence of a threat event is a function of the threat agent's potency and intention. The attribute threat event undesirability or $E_i(\text{undes})$ indicates the degree to which the event is considered detrimental to the organization. It is a function of asset value and outcome severity.

Failure Events

A security failure F_i is an event that breaches the security of a safeguard. The security failure event is the one main element that has been added to the Strawman Model. It was felt that to go directly from a threat event to an outcome event does not adequately reflect the role of safeguards in the information system. Given a threat event, there can be many possible failure events, each with its own degree of severity. For example, suppose there is a threat event that attempts to read encrypted data. The possible failure events related to the encryption mechanism could be a total failure (data in clear-text), a partial failure (weak key), or no failure (fully encrypted).

The attribute failure event identifier or $F_i(\text{ident})$ is the unique information identifying each failure event and identifies the set of safeguards involved. There are secondary attributes relating to geographic location and system or subsystem identifiers. The attribute failure severity or $F_i(\text{sev})$ shows the severity of the security failure and can range from no failure to total failure. The attribute failure probability or $F_i(\text{prob})$ is the expectation that a failure will occur. This is a function of the threat event, safeguard effectiveness, and safeguard vulnerability.

Outcome Events

A damaging outcome O_i is an event resulting in an undesirable change in the state of an asset's security. Given a threat event and a corresponding security failure, there can be many possible outcome events, each with its own degree of severity. For example, suppose there is a threat event that attempts an unauthorized terminal access to a large file and that there is a total security failure. The possible damaging outcomes could be one screen of compromised information, one chapter of compromised information, or a total compromise of information if the threat agent had sufficient time to browse through the entire file.

The attribute **outcome event identifier** or $O_i(\text{ident})$ is the unique information identifying each outcome event and relates to the assets affected. There are secondary attributes relating to geographic location and system or subsystem identifiers. The secondary attribute **outcome consequence** indicates whether the damaging outcome is related to unauthorized disclosure, destruction, removal, modification, or interruption. The attribute **outcome probability** or $O_i(\text{prob})$ is the expectation that the outcome will occur. The attribute **outcome severity** or $O_i(\text{sev})$ indicates the degree of damage. Outcome severity is related to asset value and has secondary attributes of sensitivity of disclosure, effect of corruption, maximum acceptable unavailability, and replacement cost.

Event Relationships

Events can be related so as to form a scenario. The current risk model only allows a scenario to be a simple sequence comprised of a threat event, a failure event, and an outcome event. As an example, assume a threat event of a hacker trying to dial into a computer system. The possible failure events could be that the hacker gets total system access, access to one account, or no access. The damaging outcomes would then relate to the information present in the various accounts.

The risk model intuitively captures the temporal flow of how security problems might be manifested in an operational system. One or more threat agents pose an attack against the system. If the system is vulnerable, this attack may lead to a security failure which can result in a damaging outcome expressed as a monetary loss and/or an intangible loss relating to the confidentiality, integrity, and/or availability of assets. This concept is shown in Figure 2.

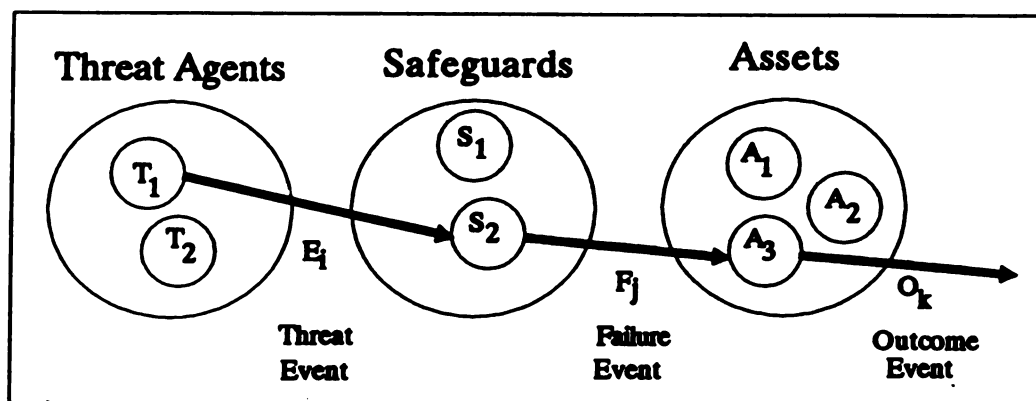


Figure 2: Sample Instance of the Risk Model

Calculating Risk

Risk is calculated based upon the probabilities of events and the severity of outcome damage. The expected loss for each outcome event is $O_i(\text{sev}) \times O_i(\text{prob})$. To see how risk is calculated, assume a scenario where threat event E_1 causes F_j which then causes outcome event O_k . The contribution to total risk by this scenario is given by the following equation:

$$X = E_1(\text{freq}) \times F_j(\text{prob}) \times O_k(\text{prob}) \times O_k(\text{sev})$$

The total system risk can then be calculated by adding the risks associated with every scenario. Note that the product of the threat event frequency and failure event probability is the expected failure event frequency. The product of the failure event frequency and outcome event probability is the expected outcome event frequency.

CONCLUSION

The Strawman Model has provided a useful framework for developing threat and risk models. This paper has shown how both the static and dynamic aspects of risk analysis can be modelled by using a static threat model and a dynamic risk model. Vulnerability is seen as an attribute of the static threat model, while risk is calculated based upon the event probabilities defined in the dynamic risk model. An information technology perspective is maintained throughout by considering safeguards as part of both the threat and risk models.

REFERENCES

- [Katzke 85] Stuart Katzke, "Federal Information System Risk Analysis Workshop,"
Montgomery, Alabama, January 1985.
- [Mayerfeld 89] Harold Tzui Mayerfeld, "A Synthesis of Working Group Reports from the
First Computer Security Risk Management Model Builders Workshop," July
1989.
- [Toth 91] Kalman C. Toth, Mark W.L. Dennison and John F. Clayton,
"A Practical Approach to Threat/Risk Analysis," 1991 Third Annual Canadian
Computer Security Symposium, Ottawa, Ontario, May 1991.
- [Workshop 88] "First Computer Security Risk Management Model Builders Workshop,"
Denver, Colorado, May 1988.
- [Workshop 89] "Report on the Second Risk Model Builders Workshop,"
Ottawa, Ontario, June 1989.

PREDICATE DIFFERENCES AND THE ANALYSIS OF DEPENDENCIES IN FORMAL SPECIFICATIONS

D. Richard Kuhn

Computer Systems Laboratory
National Institute of Standards and Technology
Gaithersburg, Md. 20899
kuhn@swe.ncsl.nist.gov

ABSTRACT

Working with formal specifications often involves an iterative cycle of development and change, either to correct errors discovered in a proof attempt, or to reflect changes in requirements. Making changes requires an understanding of the dependencies among terms in the specification. Boolean differences may be used to determine dependencies among functions in Boolean algebra. This paper introduces the notion of *predicate differences* in predicate calculus. The paper shows how predicate differences may be used to analyze the effect of changes to formal specifications; to investigate the conditions under which invalid assumptions will render a system non-secure; and in some cases may help to simplify re-verification of a modified formal specification.

1. Introduction

In working with specifications expressed in mathematical logic, one generally encounters formulas of the form $P \Rightarrow S$.¹ For example, P may be the specification of some programmed function F , and S is the security or safety property that F must ensure. Another example is the refinement of a specification R , where the refined specification R' must be shown to imply R , that is, $R' \Rightarrow R$. In developing trusted systems, it is usually necessary to show that a set of transitions P_i imply a set of security invariants S_j . The proof obligation is thus of the form $P_1 \& P_2 \& \dots \& P_n \Rightarrow S_1 \& S_2 \& \dots \& S_n$, where each P_i and S_j is an implication $A \Rightarrow B$. Usually, terms in the various P_i will be found in the S_j as well.

Like any software, formal specifications are likely to require changes, either in development or to meet changing requirements. The safety or security condition is often stable. Although the behavior of the system may change, it must still meet the security requirement, so the proof becomes $P' \Rightarrow S$ rather than $P \Rightarrow S$, where P represents the old function specification, P' the new function specification, and S the security specification. It would be helpful to have some method of analyzing the effect that the change from P to P' would have on the invariant security requirement. This paper examines a method of determining dependencies among terms of

¹ The symbols " $\&$, $|$, \neg , \Rightarrow " represent *and*, *or*, *not*, *implies*, respectively. The exclusive or operation is denoted by \oplus .

specifications written in first order logic, and ways to verify that the new specification meets the requirements of the invariant.

Formal specifications of security properties often require very large formulas in mathematical logic. Understanding the relationships between terms in the formulas can be challenging, even with the help of interactive theorem provers (which are essentially expression simplifiers with some limited inference capabilities). When a proof seems to be impossible, it is necessary to understand why. To change the specification, it is necessary to understand the effect that the change will have. Both of these situations require an understanding of the dependencies among terms. We are interested in the contribution of a particular term to the implication, and in the effect that a change to the term will have on the truth of $P \Rightarrow S$. Depending on the formulas involved, changing the value of a variable, x , may or may not affect the truth of the implication. In general, the values of other terms will determine whether a change in the value of x will change the implication $P \Rightarrow S$.

It is often necessary to change the antecedent P but still show that it meets the consequent, security condition S . The change is typically made by replacing a term from the antecedent with another term. An additional conjunct may be added as well. For example, suppose the invariant is $A \ \& \ B \ \& \ C \ \& \ D \Rightarrow S$, and it is changed to $G \ \& \ B \ \& \ C \ \& \ D \Rightarrow S$. To specify precisely the modifications to be studied here, let P represent the antecedent, and M represent the modification term, i.e. the new conjunct to be added to the antecedent. Then the modified version of P with variable x replaced is given by $P_x^?$, and the new antecedent is given by $P_x^? \ \& \ M$.² For the example, the desired new invariant $G \ \& \ B \ \& \ C \ \& \ D \Rightarrow S$ is given by $(A \ \& \ B \ \& \ C \ \& \ D) \Delta \Rightarrow S$. If the invariant is a formula in propositional logic, the effect of such a change can be determined the Boolean difference. A generalization of the Boolean difference for predicate logic will be described in Section 3.

2. Boolean Difference

The Boolean difference [Reed, 1954; Akers, 1959], can be used to calculate the dependency of a Boolean function on a literal x_i of that function. The Boolean difference of x_i with respect to F , dF/dx_i , gives the conditions under which the value of F will change if the value of x_i changes. Boolean differences have been used in digital circuit testing [Marinos, 1971], [Reed, 1973] and in computer security access control [Trueblood and Sengupta, 1986]. The Boolean difference has been generalized to multi-valued logic for VLSI circuit testing [Bell et. al, 1972], [Lu and Lee, 1984], and [Whitney and Muzio, 1988].

For a function $F = f(x_1, \dots, x_i, \dots, x_n)$, the Boolean difference of F with respect to x_i is

$$dF/dx_i = f(x_1, \dots, x_i, \dots, x_n) \oplus f(x_1, \dots, \neg x_i, \dots, x_n).$$

This is equivalent to

$$dF/dx_i = f(x_1, \dots, 0, \dots, x_n) \oplus f(x_1, \dots, 1, \dots, x_n),$$

which follows from the fact that x_i must be either 0 or 1. The difference dF/dx_i is an expression that does not contain x_i .

² The notation $P_x^?$ represents predicate P with every free occurrence of variable x replaced by term e , with suitable renaming to prevent variable capture.

A useful property of the Boolean difference is that

$$dF/dx_i = \begin{cases} 1 & \text{if } F \text{ is unconditionally dependent on } x_i \\ 0 & \text{if } F \text{ is unconditionally independent of } x_i \\ F' & \text{an expression not containing } x_i, \text{ otherwise} \end{cases}$$

To see intuitively why dF/dx_i gives the conditions under which a change in the value of x_i will change the value of F , consider that F will change if either (a) it is initially true and changing the value of x_i makes it false: $F \& \neg(F \downarrow_{x_i})$, or (b) it is initially false and changing the value of x_i makes it true: $\neg F \& (F \downarrow_{x_i})$. Note that the disjunction of (a) and (b) is, by definition, the exclusive OR.

The Boolean difference of a function $F=f(F_1, \dots, F_n)$, with respect to one of its component functions F_i is

$$dF/dF_i = f(F_1, \dots, F_i, \dots, F_n) \oplus f(F_1, \dots, \neg F_i, \dots, F_n).$$

The *partial Boolean difference* gives the effect on the truth value of a Boolean formula of a component of the formula, through a particular term. For a formula $F=f(F_1, \dots, F_n)$, the partial Boolean difference of F with respect to F_i with respect to a term x_j of F_i , is

$$dF/d(x_j | F_i) = dF/dF_i \& dF_i/dx_j$$

3. Predicate Difference

The Boolean difference can be generalized to a *predicate difference* in predicate calculus. The properties of the predicate difference are similar to those of the Boolean difference. However, the Boolean difference with respect to a term gives the conditions under which a change in the value of the term will change the value of the Boolean function. A Boolean term can change only from x to $\neg x$. The change to a predicate depends on the term substituted for x . Thus a predicate difference is with respect to a particular change x/e (the substitution of term e for free variable x), rather than simply with respect to x . Note also that the predicate difference with respect to a change x/e may still contain x .

Definition 1. Independence: P is independent of x/e when P has the same truth value as $P \downarrow_{x/e}$, i.e. $P \equiv P \downarrow_{x/e}$.

Definition 2. Dependence: If P is not independent of x/e , then P is dependent on the value of x/e .

Definition 3. Predicate difference: The predicate difference for a predicate P with respect to variable substitution x/e , denoted $dP \downarrow_{x/e}$, is $P \oplus P \downarrow_{x/e}$.

Lemma 1. $dP_i^x = 0$ iff P is independent of the value of x .

Proof.

Assume ("if" direction) $P \equiv P_i^x$ {definition of independence}

Then $(P \Leftrightarrow P_i^x)$

$\equiv (P \oplus P_i^x = 0)$ {definition of \oplus }

$\equiv (P \oplus P_i^x = 0)$

Assume $(P \oplus P_i^x = 0)$ ("only if" direction)

$\equiv (\neg(P \Leftrightarrow P_i^x) = 0)$ {definition of \oplus }

$\equiv (P \Leftrightarrow P_i^x)$

(End of proof.)

Definition 4. Unconditional Dependence: P is unconditionally dependent on x/e if P has the opposite truth value of P_i^x , i.e. $(P \Leftrightarrow \neg P_i^x) \& (P_i^x \Leftrightarrow \neg P)$

Lemma 2. $dP_i^x = 1$ iff P is unconditionally dependent on the value of x/e .

Proof.

$(P \Leftrightarrow \neg P_i^x) \& (P_i^x \Leftrightarrow \neg P)$

$\equiv \neg(P \equiv P_i^x)$

$\equiv (P \oplus P_i^x) = 1$

(End of proof.)

If dP_i^x is not 0 and not 1, then the resulting formula can be solved for 1 to determine the conditions under which P_i^x will be dependent on x . Note that if e is a Boolean and $e = \neg x$ in a propositional formula, the predicate difference is equivalent to the Boolean difference.

4. Partial Predicate Difference

The predicate difference of a predicate formula $F=f(F_1, \dots, F_n)$, consisting of component formulas connected by $\&$, $|$, or \Rightarrow , with respect to one of its component formulas F_i is

$$dF/dF_i = f(F_1, \dots, F_i, \dots, F_n) \oplus f(F_1, \dots, \neg F_i, \dots, F_n).$$

Definition 5. Partial Predicate difference: the partial predicate difference gives the effect on a formula of a component of the formula, through a change in a particular term. For a formula $F=f(F_1, \dots, F_n)$, the partial predicate difference of F with respect to F_i with respect to a change in a term x_j/e of F_i , is

$$dF/d(F_i)_{e'}^x = dF_{\neg P_i}^{F_i} \& dF_{e'}^x$$

5. Application to Dependency Analysis

Let I be an invariant $P \Rightarrow S$. To determine the effect on the invariant I of changing term x in P to e , the partial predicate difference dI/dP_i^x can be computed. This gives the conditions under which the invariant will change value, in other words, the conditions under which it becomes false, since it was true before the change.

If the invariant I has already been shown and we wish to modify P to P_i^z , we can compute the conditions under which the value of the invariant will change using the following result:

Theorem 1. Let I be an invariant $P \Rightarrow S$. Then I is dependent on the value assigned to x in P under the conditions given by $\neg P \ \& \ P_i^z \ \& \ \neg S \equiv P_i^z \ \& \ \neg S$.

Proof

$$\begin{aligned}
 dI/dP_i^z &= (P \oplus P_i^z) \ \& \ \neg S && \{\text{Definition 3 and 5}\} \\
 &\equiv (P \ \& \ \neg P_i^z \mid \neg P \ \& \ P_i^z) \ \& \ \neg S && \{\text{Definition of } \oplus\} \\
 &\equiv \neg P \ \& \ P_i^z \ \& \ \neg S && \{\text{assumed: } (P \Rightarrow S) \equiv \neg(P \ \& \ \neg S)\} \\
 &\equiv P_i^z \ \& \ \neg S && \{(P \Rightarrow S) \Rightarrow (\neg P \ \& \ \neg S \equiv \neg S)\}
 \end{aligned}$$

(End of proof.)

The form $\neg P \ \& \ P_i^z \ \& \ \neg S$ may be more useful if we expect the change x/e to maintain the invariant, because showing either $P_i^z \ \& \ \neg P = 0$, or $P_i^z \ \& \ \neg S = 0$ is sufficient to show that $P_i^z \Rightarrow S$. If $P_i^z \ \& \ \neg P$, is easier to calculate, and the result is 0, then there is no need to compute the predicate difference. Note that by Lemma 1, the invariant is independent of the change if $P_i^z \ \& \ \neg S = 0$, which is equivalent to $P_i^z \Rightarrow S$.

After analyzing the effect of the change, if a conjunct M is added to the antecedent, it is necessary to show that the new antecedent maintains the security invariant. There are then two ways to proceed with showing that the modified antecedent $(P_i^z) \ \& \ M$ maintains the invariant. The first is to show directly that $P_i^z \ \& \ M \Rightarrow S$. The second is to show that the modification guarantees that the invariant will not change value by showing that the conditions under which it becomes false do not occur, i.e., the antecedent $P_i^z \ \& \ M$ implies the negation of the partial predicate difference dI/dP_i^z i.e.: $P_i^z \ \& \ M \Rightarrow \neg[dI/dP_i^z]$. Proving this is equivalent to proving $P_i^z \ \& \ M \Rightarrow S$ directly. This result is proved formally below.

Theorem 2. $[(P \Rightarrow S) \ \& \ (M \Rightarrow \neg[dI/dP_i^z])] \Leftrightarrow [(P \Rightarrow S) \ \& \ (P_i^z \ \& \ M \Rightarrow S)]$

Proof

$$\begin{aligned}
 &(P \Rightarrow S) \ \& \ (M \Rightarrow \neg[dI/dP_i^z]) \\
 &\equiv (P \Rightarrow S) \ \& \ (M \Rightarrow \neg(P_i^z \ \& \ \neg S)) && \{\text{Theorem 1}\} \\
 &\equiv (P \Rightarrow S) \ \& \ (M \Rightarrow (\neg P_i^z \mid S)) \\
 &\equiv (P \Rightarrow S) \ \& \ (M \Rightarrow P_i^z \Rightarrow S) \\
 &\equiv (P \Rightarrow S) \ \& \ (M \ \& \ P_i^z \Rightarrow S)
 \end{aligned}$$

(End of proof.)

Thus, if the modification term M implies the negation of the predicate difference, the invariant will be preserved.

5.1. Example

Consider a system which uses a token to control access to a network. To gain access, a user must have both a valid token and the right password. The system maintains the following state invariants (among others) as security requirements.

A user is authorized only if the token is authorized:

$(u_auth \Rightarrow t_auth)$

A token is authorized only if its password is active (non-zero):

$(t_auth \Rightarrow pw \neq 0)$

We wish to ensure that the following state transition invariant holds:

A token can be activated (i.e., its password changed from zero to non-zero) only by the security officer:

$(pw' \neq 0 \ \& \ pw = 0 \Rightarrow s_auth)$

The password changing function is

```
chgpasswd(input_val)
{
  /* if security officer, then change password to input value */
  if (s_auth) pw := input_val
}
```

This *chgpasswd* function is modeled by

$(s_auth \Rightarrow pw' = \text{inval}) \ \& \ (\neg s_auth \Rightarrow pw' = pw)$

A proof is done to show that the state invariants plus the effect of the *chgpasswd* function ensure the state transition invariant (the function must also maintain the invariants, but this is not shown for conciseness).

$(u_auth \Rightarrow t_auth) \ \&$
 $(t_auth \Rightarrow pw \neq 0) \ \&$
 $(s_auth \Rightarrow pw' = \text{inval}) \ \&$
 $(\neg s_auth \Rightarrow pw' = pw)$
 $\Rightarrow (pw' \neq 0 \ \& \ pw = 0 \Rightarrow s_auth)$

Suppose that the design is to be changed to allow either the user or the security officer to change passwords, rather than requiring the security officer to do so. The *chgpasswd* function specification then becomes:

$((s_auth \mid u_auth) \Rightarrow pw' = \text{inval}) \ \& \ (\neg(s_auth \mid u_auth) \Rightarrow pw' = pw)$

After making the change to the specification, a new proof must be conducted. If the proof fails, the specification must be analyzed manually to determine why, then appropriate changes made. The conditions under which the change will affect the state transition invariant can be calculated using the predicate difference. As it turns out, the predicate difference is 0, so the change will not affect the invariant. By Theorem 1, the predicate difference is

$$\begin{aligned}
&(u_auth \Rightarrow t_auth) \ \& \\
&(t_auth \Rightarrow pw \neq 0) \ \& \\
&(u_auth \mid s_auth \Rightarrow pw' = inval) \ \& \\
&(\neg(u_auth \mid s_auth) \Rightarrow pw' = pw) \ \& \\
&\neg(pw' \neq 0 \ \& \ pw = 0 \Rightarrow s_auth) \equiv 0
\end{aligned}$$

Depending on the problem, the predicate difference may be either more or less effort to calculate than a new proof. The advantage in computing the predicate difference is in determining the conditions under which a change will render non-secure a system that was previously shown secure.

6. Analyzing the Effect of Security Flaws

One important problem in security evaluations is to determine the effect of violations of assumptions. In general, violations of assumptions will affect the security of the system under some conditions, but not make the system non-secure all the time. The predicate difference for a hypothesized violation of assumptions gives the conditions under which the security invariant does not hold.

In a state machine model a proof is given that transitions T_i imply the security invariant S , i.e., $T_1 \Rightarrow S \ \& \ T_2 \Rightarrow S \ \& \ \cdots \ \& \ T_n \Rightarrow S$. A violation of assumptions in a transition, such as the failure of a variable to maintain a specific value, can be modeled by letting a term e represent the potential new value of a variable x , then computing the predicate difference $d(T \Rightarrow S)_x^e$. The predicate difference gives the conditions under which the invariant will change truth value, that is, the conditions under which the system would not be secure.

7. A Metric for Predicate Changes

A metric for changes to a predicate can be defined by using the predicate difference to define a partial order: $x/e \leq z/f$ if $dP_x^z \Rightarrow dP_f^e$ (x may equal z and e may equal f). Also define $x/e < z/f$ if $dP_x^z \Rightarrow dP_f^e$ but not $dP_f^e \Rightarrow dP_x^z$. The partial order $x/e \leq z/f$ expresses the fact that the change x/e is "smaller" than z/f . The smallest change x/e is no change at all, where $dP_x^x = 0$, as shown in Lemma 1.

If $dP_x^z \Rightarrow dP_f^e$ then it could be said that P_x^z differs less from P than does P_f^e . To compare how two predicates Q and R differ from P , the differences $P \oplus Q$ and $P \oplus R$ can be computed. (We do not necessarily know what substitutions x/e , if any, will make P_x^z equal to Q or R .) If $P \oplus Q \Rightarrow P \oplus R$ then Q differs less from P than R , otherwise R differs less than Q (unless $Q = R$).

7.1. Example

Given a predicate $(a \mid b)$, does a/c represent a bigger or smaller change than $a/(a \mid c)$? The predicate difference $d(a \mid b)_c^e$ is $c \ \& \ \neg b \mid b \ \& \ \neg c$, and $d(a \mid b)_{a \mid c}^e$ is $\neg a \ \& \ \neg b \ \& \ c$. So $d(a \mid b)_c^e \Rightarrow d(a \mid b)_{a \mid c}^e$, i.e., a/c is a bigger change than $a/(a \mid c)$. Although the substitution $a/(a \mid c)$ is a greater *text* change than a/c , the predicate that results from $a/(a \mid c)$ simply enlarges the number of states (since $a \mid b \Rightarrow a \mid b \mid c$), but a/c changes the predicate to define a different set of states.

8. Application to Verification

Some of the previous results can be used in strategies for computer assisted theorem proving. In secure systems verification one often proves invariants of the form $P_1 \Rightarrow S_1 \ \& \ P_2 \Rightarrow S_2 \ \& \dots \ P_n \Rightarrow S_n$. Many proof tools treat the system being specified as a finite state machine. To prove consistency with an invariant S , the user shows that the new values of variables after each state transition maintain the invariant. The proof is inductive. The initial conditions are shown to satisfy the invariant, then each transition is shown to maintain it by substituting values of variables that change in a transition into the invariant. The proof is: $invar \Rightarrow invar'$, where $invar'$ is the invariant with the postcondition values of variables substituted in. A proof by contradiction is used. The system substitutes the new values of variables changed in a state transition (as given by the postcondition) into the invariant to get $invar'$, and generates the conjunction $invar \ \& \ \neg invar'$. The user must show that this conjunction results in a contradiction, that is, $invar \ \& \ \neg invar' = 0$, which is equivalent to $invar \Rightarrow invar'$.

8.1. Example

Suppose the invariant is $p \mid q \Rightarrow r$, and the effect of the state transition is $q' = p \ \& \ q \mid q$. The new value of the invariant is $p \mid (p \ \& \ q \mid q) \Rightarrow r$, so the invariant is maintained. This is shown by showing a contradiction: $[p \mid q \Rightarrow r] \ \& \ \neg[p \mid (p \ \& \ q \mid q) \Rightarrow r] = 0$.

Suppose we are proving that a system maintains the following invariant:

$$(1) \quad (w \mid t \mid u \Rightarrow p \ \& \ d \ \& \ l) \ \&$$

$$(2) \quad (w \Rightarrow t) \ \&$$

$$(3) \quad (t \Rightarrow u)$$

The new value of w , denoted $N''w$, is given by the substitution $w/(t \ \& \ d \ \& \ p) \mid w$. The system assumes the negation in preparation for proof:

$$(4) \quad (N''w \mid t \mid u \ \& \ \neg(p \ \& \ d \ \& \ l)) \mid (N''w \ \& \ \neg t)$$

After substitution, we have $(P_1)_t^s \ \& \ \neg S_1 \mid (P_2)_t^s \ \& \ \neg S_2 \mid (P_n)_t^s \ \& \ \neg S_n$, i.e.

$$(4) \quad ((t \ \& \ d \ \& \ p) \mid w \mid t \mid u \ \& \ \neg(p \ \& \ d \ \& \ l)) \mid ((t \ \& \ d \ \& \ p \mid w) \ \& \ \neg t)$$

The system assumes this new information and the user is required to show a contradiction between the assumed formula and the invariant. At this point the proof would proceed by taking the two disjuncts in turn. The first,

$$(5) \quad ((t \ \& \ d \ \& \ p) \mid w \mid t \mid u \ \& \ \neg(p \ \& \ d \ \& \ l))$$

would be proven by directing the system to simplify

$$(6) \quad (w \mid t \mid u \Rightarrow p \ \& \ d \ \& \ l)$$

in the invariant, since the second conjunct is the negation of the consequent in (6). This simplifies to $\neg w \ \& \ \neg t \ \& \ \neg u$. A contradiction can now be derived by directing the system to simplify the left conjunct of (5). The proof of the first disjunct of (4) is now complete.

The second disjunct of (4) which is

$$(7) \quad ((t \ \& \ d \ \& \ p \mid w) \ \& \ \neg t) \text{ can now be proven by first directing the system to simplify it, resulting in an assumption } w.$$

Directing the system to simplify (2) now results in a contradiction because the consequent t contradicts the second conjunct, $\neg t$ of (7). The proof is now complete.

A second strategy is suggested using some of the previous results for predicate differences. By Lemma 1 and Theorem 1, if the conjunctions of the modified antecedents $(P_n)_t^*$ with the negations of the original antecedents, $\neg P$ & $\neg S$, are all 0, then the invariant is independent of the change, that is, the invariant is maintained. Computing the result shows that the predicate difference is indeed equal to 0, so the invariant is maintained:

$$(\neg(w \mid t \mid u) \& (t \& d \& p \mid w)) = 0$$

and

$$(\neg w \& (t \& d \& p \mid w) \& \neg t) = 0.$$

Note that since

$$(\neg(w \mid t \mid u) \& (t \& d \& p \mid w)) = 0$$

it is not necessary to compute

$$\neg(w \mid t \mid u) \& (t \& d \& p \mid w) \& \neg(p \& d \& l).$$

9. Summary and Conclusions

Predicate differences can be an effective analytical tool for evaluating the effect of changes to formal specifications. They may also be useful in re-verifying specifications after modification; determining if a change will cause a previously secure system to become non-secure; and as a metric for changes to predicates.

Examples presented in this paper were based on real specifications, but additional experience is needed to explore the technique. Integrating the calculation of predicate differences into a verification tool for a formal specification language would be helpful toward this end. The formal specification language Z provides a schema calculus that seems particularly suitable, if tools for manipulating Z schemas become available.

10. Acknowledgements

Suggestions by Bill Majurski, Jim Lyle, John Cherniavsky, and the anonymous referees were helpful in clarifying the text.

11. References

- [Akers, 1959] S.B. Akers. "On a Theory of Boolean Functions," *SIAM Journal* Vol. 7, No. 4.
- [Bell et. al, 1972] N. Bell, E.W. Page, and M.G. Thomason, "Extension of the Boolean Difference Concept to Multi-valued Logic Systems," *Proceedings of the 1972 Symposium on the Theory and Applications of Multiple-Valued Logic Design*
- [Gries, 1987] D. Gries. *The Science of Programming*, Springer Verlag, New York, 1987.
- [Lu and Lee, 1984] H. Lu and S.C. Lee, "Fault Detection in M-Logic Circuits Using the M-Difference," *Proceedings of the International Symposium on Multiple Valued Logic, 1984.*"
- [Marinos, 1971] P.N. Marinos. "Derivation of minimal complete sets of test-input sequences using Boolean differences," *IEEE Transactions on Computers*, Vol. C-20, No. 1.

- [Muller, 1954] D.E. Muller. "Application of Boolean Algebra to Switching Circuit Design and Error Detection," *Transactions of the Institute of Radio Engineers*, Vol. EC-3.
- [Reed, 1954] I.S. Reed. "A Class of Multiple-error Correcting Codes and the Decoding Scheme," *Transactions of the Institute of Radio Engineers*, Vol. IT-4.
- [Reed, 1973] I.S. Reed. "Boolean Difference Calculus and Fault Finding," *SIAM Journal of Applied Mathematics*, Vol. 24, No. 1.
- [Trueblood and Sengupta, 1986] R.P. Trueblood and A. Sengupta. "Dynamic Analysis of the Effects Access Rule Modifications Have Upon Security," *IEEE Transactions on Software Engineering*, Vol. SE-12, No. 8.
- [Whitney and Muzio, 1988] M. Whitney and J. Muzio. "Decisive Differences and Partial Differences for Stuck-at Fault Detection in MVL Circuits,"

Preventing Weak Password Choices

Eugene H. Spafford
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907-1398
spaf@cs.purdue.edu

June 1991

Abstract

A common problem with systems that use passwords for authentication results when users choose weak passwords. Weak passwords are passwords that are easy to guess, or likely to be found in a dictionary attack. Thus, the choice of weak passwords may lead to a compromised system.

Methods exist to prevent users from selecting and using weak passwords. One common method is to compare user choices against a list of unacceptable words. The problem with this approach is the amount of space required to store even a modest-sized dictionary of prohibited password choices.

This paper describes a space-efficient method of storing a dictionary of words that are not allowed as password choices. Lookups in the dictionary are $O(1)$ (constant time) no matter how many words are in the dictionary. The mechanism described has other interesting features, a few of which are described here.

Keywords: passwords, dictionaries, password aging

1 Introduction

Passwords are a commonly-used method of authentication. A unique sequence of characters is presented to the system when identification is needed. This sequence is then compared with a stored sequence, perhaps after some transformation (e.g., encryption). A match provides the proof of identity.

One weakness with password systems is the choice of the password. If the choice of possible characters to use in the password is too small, or if the overall length of the password is too short, the password may be compromisable. Even a rich character set may not be sufficient to create secure passwords if the combination of characters is restricted to an arbitrary set of possibilities. Thus, good password choice should avoid common words and names (cf. [1, 6, 10, 12, 15]).

As an example, consider the UNIX¹ password system.[12] The current password mechanism is based on a cryptographic transformation of a fixed string of zero bits, using the user-supplied password as a key. The transformation is an altered version of DES encryption, performed 25 times. The transformation is sufficiently slow so that exhaustive keyspace attacks are currently not practical, although fast implementation such as *deszip*,[3] can perform many thousands or tens of thousands of comparisons per second.

In UNIX, the encrypted version of the password has traditionally been kept in a world-readable file; the safety of the passwords has been protected by the time-complexity of an exhaustive attack. Thus, one of the keys to the safety of UNIX passwords is a large potential keyspace for passwords. If the full character set is used, and seven or eight-character passwords are chosen, the number of potential passwords to be searched is far too large to be successfully searched, even at high speed.² Unfortunately, users often select passwords that do not exploit the large keyspace available. Instead, they choose common words and names, or simple transformations of those names. This greatly simplifies an attacker's task.

This tendency to select weak³ passwords has led to a number of system break-ins,

¹ UNIX is a trademark of Unix System Laboratories, Inc.

² Assuming a usable character set of 120 characters, there are 43,359,498,756,302,520 (4.34e16) possible passwords of length one through eight. At 50,000 attempts per second, an exhaustive search of this keyspace would require over 27,480 years to complete.

³ Strength being defined as the ability to resist a dictionary-based attack, and weakness as its opposite.

some quite highly publicized: cf. [14, 18, 20, 21, 23]. Current technology is such that construction of a large pre-encrypted dictionary on-line using optical disks is easily done. By creating such a dictionary, a password search and attack may be easily conducted in a matter of seconds. Without such a database, but using a tool such as *deszip* on a modern workstation, it is possible to make a full scan of 300,000 dictionary entries against several hundred passwords in a matter of a few hours or days.

Despite wide-spread publication of good password policy and the risks inherent in bad passwords, users continue to select weak passwords. This is a continuing threat to the best-managed systems. (For example: [2, 7, 8, 9, 10, 11, 15, 19, 22, 24].)

There are four basic methods for a system administrator to enforce better password security on a computer system:

1. Educate and encourage users to make better choices of passwords.
2. Generate strong passwords for users and do not allow them to choose passwords of their own creation. This is often done using some random password generator.
3. Check passwords after-the-fact and force users to change those that can be easily broken with a dictionary attack.
4. Screen users' password choices and prevent weak ones from being installed.

This first method, that of educating users to choose strong passwords, is not likely to be of use in environments where there is a significant number of novices, or where turnover is high. Users might not understand the importance of choosing strong passwords, and novice users are not the best judges of what is "obvious." For instance, novice users (mistakenly) may believe that reversing a word, or capitalizing the last letter makes a password "strong."

A further problem is if the education provided to users on how to select a password is itself dangerous. For instance, if the education provided gives users a specific way to create passwords — such as using the first letters of a favorite phrase — then many of the users may use that exact algorithm, thus making an attack easier.

The second method of strengthening passwords is to generate the passwords for the users and not allow them the opportunity to select a weak password. For this mechanism to work well the passwords need to be randomly drawn from the whole keyspace. Unfortunately, this method also has flaws. In particular, the "random" mechanism chosen might not be truly random, and could be analyzed by an attacker.

Furthermore, random passwords are often difficult to memorize (especially if they are changed (*aged*) regularly). As a result, users may write the passwords down, thus providing an opportunity to intercept them without the effort of a dictionary search.

The third method of preventing poor password choice is to scan the passwords selected, after they are chosen, to see if any are weak. This is supported by many systems, including *deszip* and COPS.[5] There are significant problems with this approach:

- The dictionary used in the search may not be comprehensive enough to catch some weak passwords. Outside attackers might think of these choices, but the password scanner would not include them in the search.
- The scanning approach takes time, even for a fast implementation. A lucky (or determined) attacker may be able to penetrate a system through a weak password before it is discovered by the scanner. This is especially a problem in an environment with a very large number of users.
- The output of a scanner may be intercepted and used against the system.

Additionally, there is not always a correlation between finding a weak password and getting it replaced with a stronger one. At many universities, for example, faculty members have repeatedly been informed of the weakness of their passwords as exposed by a scanner, but they have not chosen new passwords in years. The administration of university systems is such that it is impossible to force faculty members to choose better passwords.

The fourth method, that of disallowing the choice of poor passwords in the first place, appears to have none of the drawbacks mentioned above. However, it too has difficulties associated with it. In particular, the storage required to keep a sufficiently large dictionary may prevent this method from being used on workstations and small computer systems. For instance, the standard UNIX dictionary, */usr/dict/words*, is about 25,000 words and 200,000 bytes of space. A dictionary of 10 to 20 times that size would be necessary for reasonable protection; there are over 170,000 words in Webster's New World Dictionary, and that would occupy well over a million bytes of disk storage. That figure does not include many slang and colloquial words and phrases, nor does it include any user names, local names and phrases, likely words in foreign languages, or other strings shown to be poor password choices. A moderately comprehensive dictionary I have used in password research has over 500,000 entries, and requires almost five million bytes of storage.

Maintaining such a large dictionary is also difficult. To add new words or phrases means that the dictionary must have additional space overhead for indexing or it must be sorted after each addition — otherwise, lookups take time proportional to the length of the dictionary. In small computer environments, neither of these alternatives may be appropriate.

2 OPUS

The OPUS Project⁴ is intended to address the space problems associated with a sufficiently complex password screening dictionary. The goal is to derive a mechanism that provides protection equivalent to a comparison against a large dictionary, yet be small enough to be practical in a small computer environment.

2.1 The Dictionary Filter

The central component of this system is a Bloom filter-encoded version of the dictionary.[4] A Bloom filter is a well-studied probabilistic membership checker, often used in applications such as spelling checkers.[13, 16, 17] It works as follows: a word to be entered into the filter is passed through n independent hash functions generating integer values. Each of these values is used as an index into the filter, represented as a bitmap. The bits (one per hash function) corresponding to the input word are then set. This procedure is repeated for each word to be entered into the filter.

When a lookup is to be performed, the word to be examined is passed through the same hash functions and the corresponding bits in the filter are examined. If any of the bits is reset (i.e., not set), then the word is determined not to be present in the dictionary. If all the corresponding bits are set, the likelihood is high that the word was in the list that was used to build the dictionary. In the case of OPUS, this means the choice is rejected as a weak password choice. The probability of a false rejection can be set arbitrarily low by increasing the size of the bitmap and increasing the number of hash functions used; an obvious upper bound on the size of the hash table is the size of the plaintext dictionary.

To be more exact, assume we have a hash table of N bits, and d independent hash functions. From [4], with n words we have the proportion of bits left unset, ϕ , equal

⁴Obvious Password Utility System.

to

$$\phi = \left(1 - \frac{d}{N}\right)^n$$

A word will be falsely shown as present in the dictionary if and only if it hashes to a set of bits that are all set. The expected proportion, P , of words in the input space that will be mistakenly shown as in the dictionary is thus

$$P = (1 - \phi)^d$$

From these equations, we can derive appropriate values to choose for our filter and hash functions.

For example, suppose we pick $n = 250,000$ words for the dictionary, and we wish to have a 0.5% chance ($P = 0.005$, i.e., one out of 200) of false positives on any given text string. If we choose six uniform hash functions, we will need 2,800,000 bits of storage and achieve $\phi = 0.586$. This works out to a file of 350K bytes. Doubling the chance of false positives to 1% ($P = 0.01$) results in needing only 300K bytes of storage for the dictionary with six hash functions. Storing the full dictionary as plaintext would likely take in excess of 2 Mb of storage. Thus, we are able to achieve almost a seven-fold compression with only a small loss of accuracy.

As can be seen from the above examples, with the appropriate choice of hash functions it is possible to greatly reduce the storage necessary to keep an extensive dictionary of words to compare against password choices. By making queries on the dictionary with variations of the candidate password — upper/lower case, reversed, trailing digit, etc. — it should be possible to quickly check for the strength of the password. Each probe into the dictionary is basically a constant-time operation, so the number of words in the dictionary has no effect on the time of access. If the union of all the probes results in a positive response, the user is told to try again.

2.2 Other Features

The model of the dictionary used in OPUS provides benefits other than simple dictionary lookup. By providing a writable interface to the dictionary for the system administrator, it is a simple task to add the representation of new words to the dictionary. The administrator can therefore augment the dictionary with local user names and colloquialisms. Adding words to the dictionary requires no expensive sorting or

temporary storage. Furthermore, the system administrator never needs to be concerned if a word has already been added — adding a word more than once has no effect.

The OPUS system also supports password aging. With password aging, users are required to change their passwords periodically. However, a common fault with password aging is that users attempt to reuse old passwords, and this may present a security risk.

OPUS can be configured so that whenever a password is changed, it is added to the dictionary. Thus, if a user attempts to reuse an old password, she will find it already in the dictionary, and the choice will not be allowed. As seen from the value of ϕ , above, there is plenty of room in the dictionary for adding new words, so even prolonged operation will not result in a noticeable degradation of service. Also, simple steps need to be taken to prevent very frequent changes of passwords that might degrade the filter, such as putting a minimum time for which a new password must be kept before a change is again allowed.

One obvious problem with updating the dictionary in this manner is the possibility of an attacker using delta information to craft a set of password attempts. That is, by observing the changes made to the filter when another user changes his password, an attacker might be able to use the hash functions to derive a set of possible text strings that account for the changes, and use these in a penetration attempt.

A related problem is if an attacker finds a way to use the dictionary as a filtering mechanism to exclude patterns when doing a brute-force key-space search to break passwords. Doing a probe into the dictionary will determine if a candidate is a possible choice or not, thus saving (some) on the computation required to perform an exhaustive search.

Luckily, there is a simple way to defeat these problems. Instead of hashing plaintext words into the dictionary, OPUS first encrypts the words to be entered or examined. The encryption must be something time-consuming, similar to multiple rounds of the DES function, and computationally infeasible to reverse. The hashing algorithms are then applied to the encrypted string rather than to the plaintext. Thus, to gain any information from the dictionary, either as a pre-screen or as a source of delta information, would require much more computational effort than some other approach (e.g., exhaustive key-space search).

To further confound attackers, the key used to encrypt the input words should either be site-selectable, or generated as a function of the input word itself. For

instance, if something similar to the UNIX mechanism is used, the first and last letter of the input word, converted to uppercase, could be used as the “salt.” As there is never a reason to recover words from the dictionary, this choice of key is something that probably cannot be recovered unless the plaintext word is known.

3 Final Remarks

This paper has discussed the motivations and design behind a system for preventing users from installing weak passwords. The system should be compact and simple to customize and enhance. It can be used standalone, as a front-end to an existing password program, or coupled with some form of password generator so as to prevent the accidental generation of a word susceptible to dictionary attacks.

The choice of hashing algorithms used with the system is critical for the success of the filter. Choosing non-uniform or overlapping hash algorithms reduces the effectiveness of the Bloom filter by increasing the incidence of false positives (effectively shrinking the number of useful bits employed). When possible, the hash algorithms should be chosen to produce the same results whether used on a string or on its reverse. This will allow probes for common words and their reverses to be made simultaneously. Case-insensitivity can also be used in the hash functions, but this may result in too great a narrowing of the key space; words in monospace, or with only a leading or trailing capital letter are perhaps the only combinations that need to be examined.

A UNIX version of OPUS is being constructed. It will be preloaded with a locally-developed dictionary of almost 500,000 strings. Experiments will then be conducted to determine, for this dictionary, the optimal working size and number of hash functions. Further experiments will determine the accuracy rate for rejection of candidate passwords that are not present in the real dictionary, and the speed of operation. By performing side-by-side experiments with users selecting potential passwords and comparing a dictionary search with the results of the Bloom filter, it should be possible to determine the overall utility of this approach.

References

- [1] Ana Maria De Alvaré. How crackers crack passwords, or what passwords to avoid.

- Technical Report UCID-21515, Lawrence Livermore National Laboratory, 1988.
- [2] Ana Maria De Alvaré and Jr. E. Eugene Schultz. A framework for password selection. Technical Report UCRL-99382, Lawrence Livermore National Laboratory, 1988.
 - [3] M. Bishop. An application of a fast data encryption standard implementation. *Computing Systems*, 1(3):221–254, 1988.
 - [4] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July 1970.
 - [5] Daniel Farmer and Eugene H. Spafford. The COPS security checker system. In *Proceedings of the Summer Usenix Conference*. Usenix Association, June 1990.
 - [6] Simson Garfinkel and Eugene H. Spafford. *Practical Unix Security*. O'Reilly and Associates, Inc., May 1991.
 - [7] David L. Jobusch and Arthur E. Oldehoeft. A survey of password mechanisms: Weaknesses and potential improvements. part 2. *Computers & Security*, 8(8):675–689, 1989.
 - [8] David L. Jobusch and Arthur E. Oldehoeft. A survey of password mechanisms: Weaknesses and potential improvements. part 1. *Computers & Security*, 8(7):587–603, 1989.
 - [9] Daniel V. Klein. A survey of, and improvements to, password security. In *UNIX Security Workshop II*, pages 5–14. The Usenix Association, August 1990.
 - [10] Belden Menkus. Understanding password compromise. *Computers & Security*, 7(6):549–552, December 1988.
 - [11] Chris Mitchell and Michael Walker. The password predictor — a training aid for raising security awareness. *Computers & Security*, 7(5):475–481, October 1988.
 - [12] Robert Morris and Ken Thompson. Password security: a case history. In *Unix Programmer's Supplementary Documentation*. AT&T, November 1979.
 - [13] James K. Mullin. A second look at Bloom filters. *Communications of the ACM*, 26(8):570–571, August 1983.

- [14] Neil Munro. Simple password opens navy computer to hacker. *Government Computer News*, 7(15):61, July 1988.
- [15] National Computer Security Center. Password management guideline. Technical Report CSC-STD-002-85, US Department of Defense, 1985.
- [16] Robert Nix. Experience with a space efficient way to store a dictionary. *Communications of the ACM*, 24(5):297–298, May 1981.
- [17] M. V. Ramakrishna. Practical performance of Bloom filters and parallel free-text searching. *Communications of the ACM*, 32(10):1237–1239, October 1989.
- [18] Brian Reid. Reflections on some recent computer break-ins. *Communications of the ACM*, 30(2):103–105, February 1987.
- [19] Bruce L. Riddle, Muray S. Miron, and Judith A. Semo. Passwords in use in a university timesharing environment. *Computers & Security*, 8(7):569–578, 1989.
- [20] Donn Seeley. Password cracking: A game of wits. *Communications of the ACM*, 32(6):700–703, June 1989. 1989.
- [21] Eugene H. Spafford. The Internet Worm: Crisis and aftermath. *Communications of the ACM*, 32(6):678–687, June 1986.
- [22] Cliff Stoll. How secure are computers in the U.S.A.? an analysis of a series of attacks on MilNet computers. *Computers & Security*, 7(6):543–547, 1988.
- [23] Cliff Stoll. *The Cuckoo's Egg*. Doubleday, NY, NY, October 1989.
- [24] Patrick H. Wood and Stephen G. Kochan. *Unix System Security*. Hayden Book Company, 1987.

PUTTING POLICY COMMONALITIES TO WORK

D. ELLIOTT BELL

TRUSTED INFORMATION SYSTEMS, INC.

**3060 Washington Road
Glenwood, Maryland 21738**

Abstract

An examination of general policy support is undertaken using an abstraction of trusted systems termed the "Universal Lattice Machine." This policy supportability is applied to selected policies from the literature. It is shown that multinational sharing, Clark & Wilson, dynamic separation of duty, the Chinese Wall security policy, and originator control are supportable in this fashion. A constructive theoretical method of switching between isomorphic policy representations is presented in an annex.

OVERVIEW

Recognizing and documenting the fact that different-seeming policies governing the access by people to information can actually have strong commonalities (in fact, exhibit actual mathematical isomorphism, as shown in [BELL90]) is only a first step. Putting that result to practical use requires several further steps. One needs to resolve the question of whether the provision of policy conversion logic within a TCB will be overly complex and cumbersome. One needs to determine which policies of interest can indeed be addressed using the conceptual and actual lattice-policy-enforcing machines available, and conversely, which cannot. One needs to devise policy commonality tools to be provided with trusted systems that enable a system security administrator to reap the benefits implicit in trusted systems for the support of different-seeming policies.

This paper focuses on the issue of supporting "policies" required by organizations, groups, or, in general, by enterprises, using the technical policies provided at the system level by lattice-policy-enforcing trusted systems. The attempt is made throughout to keep clear and distinct the two notions of "policy", that of the enterprise and that of the system. (See also [STER91] and [TDI91] for discussion of the distinction and its importance.) The terms "enterprise policy" and "technical policy" will be used when the distinction between the two levels of discourse needs to be emphasized or made clear.

The paper begins with the introduction of a generalized, conceptual trusted system, termed the "universal lattice machine". Extensions to the basic functionality that are implicit in the basic properties are then introduced. The extensions are binding, exclusion, roll-back, and n-person control. Each extension is realized two ways, one using discretionary access control

mechanisms only and the other using non-discretionary access control mechanisms. Then, using the universal lattice machine construct, several identified enterprise-policies from the literature are addressed. It is demonstrated that multinational sharing, Clark and Wilson, dynamic separation of duty, the Chinese Wall policy, and ORCON can be directly treated using universal lattice machine functionality and assurances. The paper concludes with directions for further work and an annex that resolves in the negative the question of whether policy conversion logic would be overly complex for inclusion in a minimized TCB.

UNIVERSAL LATTICE MACHINE

For purposes of this discussion, we will use the notion of a universal lattice machine (ULM) that abstracts the essential features of trusted systems. A ULM has subjects and objects, as well as the ability to deal with groups of subjects (such as, but not limited to, Multics Projects or UNIX groups) and groups of objects (such as, but not limited to, UNIX filesystems). The access to objects by subjects in several access modes is restricted in two ways. The first type of restriction is based on access control lists (ACL's) and negative-access control lists (NACL's), pairing subjects and groups of subjects to objects and groups of objects. This restriction is discretionary in the sense that there is in general a capability for subjects to alter the permissions recorded in ACL's and NACL's at their own discretion.¹

The second type of restriction is based on boolean-lattice values assigned both to subjects and objects. A particular mode of access will be permitted provided that a logic equation linking the subjects and objects involved evaluates to *true*. As an example, \underline{r} access of a subject S to an object O is allowed provided $\text{lattice-value}(S) \Rightarrow \text{lattice-value}(O)$.

The general notion of a ULM as described is predicated on a central portion of the system that provides the ULM abstractions with a high degree of confidence in the immutability, correctness, and unavoidability of those abstractions (both conceptually and implementationally). That is, the presence of ULM mechanisms and limitations in the stream of access requests and mediations is assured and the metadata (which includes both data structures on the basis of which decisions are made and executables that embody the logic) cannot be altered except in known, advertised ways. In a word, the ULM presumes a reference monitor in the sense of [ANDE72] and [TCSEC85].

Part of the basic functionality that a ULM provides is the ability to alter the metadata of the ULM itself. Some categories of metadata change are altering the human-readable version of the lattice-values; altering group membership of subjects and objects; altering entries on

¹ Latitude in extending access privilege varies from instance to instance of a ULM. In some cases, an entry in an ACL (not subordinate to a NACL entry) will imply the ability to extend access permissions. In others, an explicit right to extend is required (as in UNIX ownership and in cases where an explicit control attribute exists, such as in Multics *modify* access to the parent directory). The general case here will leave the limitations on altering ACL's and NACL's unspecified.

ACL's and NACL's; establishing new user accounts or new system subjects ("creating" subjects); changing the lattice-range of a subject (that is, altering the simultaneous view-alter range of a subject to change the "trustedness" of the subject); changing the lattice-value of objects; and changing the maximum lattice-value of subjects. These basic metadata changes themselves fall into different groups with regard to their effect on previously-established confidence. Several of the functions are confidence-neutral: they do not alter the confidence in the ULM since they are part of the functionality analyzed and reviewed in the establishment of confidence. Altering human-readable versions of lattice-values and altering ACL's and NACL's fall into this category,² as do altering group membership of subjects and objects and creating subjects. Changing a subject's range alters its potential interaction with other trusted subjects and with other trusted code within the Reference Validation Mechanism (the implementation of a reference monitor). Such a change can have a substantial impact on previously-established confidence. When an untrusted subject (one whose range consists of a single lattice-value) has that level raised to a higher value (one that implies, or dominates, the original value), there is no impact on the confidence, providing that proper alterations in system state are made to retain secure state after the change. Similarly, alteration of an object's lattice-value has no impact on the confidence in the system, provided the proper bookkeeping and alterations are bound to the change.³

INTRINSIC EXTENSIONS TO ULM FUNCTIONALITY

Given the basic functionalities of altering a ULM's metadata, one can construct a set of more complex functions for the actualization of various policies. For each function, a realization using either the discretionary mechanisms (ACL's and NACL's) or the non-discretionary mechanisms (lattice-values) is possible. Four functions will be described below — binding, exclusion, roll-back, and n-person control. Each function will be described both in discretionary and non-discretionary terms and the implications of the alternate forms will be explored.

Binding. The basic concept of binding is derived from [CLWI87]. Stated narratively, what is desired is the ability to limit invocation of specific code for the processing of particular data items. The expectation is to (1) limit invocation and (2) limit manipulation of data items (designated VDI and ADI, for view-data-items and alter-data-items) to the combination of authorized invokers (AI) and identified processing code (T). Implicit is the expectation that the code, the set of authorized invokers, and the controlled data items, both for viewing and altering, can be altered or viewed only under the control of a reference validation mechanism, an assumption present in the Clark & Wilson paper. [CLAR90] Within the context of a

² Different instances of ULM's will provide different limitations on the alteration of ACL's and NACL's. The differences sometimes matter in the compound tasks that can be constructed out of the more basic functionalities under discussion here.

³ See, for example, rules change-subject-current-security-level (R10) and change-object-security-level (R11) in [BLP75, pp. 110-111] and the rule NRange in [BELL86, p. 39].

ULM, the problem will be stated as trying to limit invocation of a single transaction T to an identified set of authorized invokers $\{AI\}$ for the manipulation of the data items $\{VDI\} \cup \{ADI\}$, the sets of view-data-items and alter-data-items, respectively. Both the AI and the T will be viewed as subjects and the xDI as objects.

A discretionary solution to binding is to establish a group of subjects for the AI and give "invoke" access to T only to the subjects in the AI group of subjects by setting the ACL of T ; establish two groups of objects, the VDI and ADI groups, and limit access to those groups to T by setting the ACL of the VDI and ADI groups. This solution is subject to the depredations of safety [HRU76] to the extent that the particular instance of the ULM allows extension of access privilege based on existing access permission. If changes to the ACL's and NACL's were strictly limited to administrative action, then the effects of safety would be constrained, but a potential flow of information (*vice* the alteration of ACL's and NACL's) would still be present.

A non-discretionary solution would assign unique lattice-values to mark the various system elements. The set of subjects $\{AI\}$ would be given the mark $MARK-AI$; the transaction T , $MARK-T$; the view-data-items $\{VDI\}$, $MARK-VDI$; and the alter-data-items $\{ADI\}$, $MARK-ADI$. Invocation of T would be limited to subjects whose lattice-value implies (includes) $MARK-AI$. T would be assigned the lattice-value $MARK-VDI \wedge MARK-ADI$. Viewing $\{VDI\}$ objects would be limited to subjects whose lattice-value implies (includes) $MARK-VDI$ and altering $\{ADI\}$ objects would be limited to subjects whose lattice-value is implied by (is included by) $MARK-ADI$.⁴ The necessary relations among the lattice-values $MARK-AI$, $MARK-T$, $MARK-VDI$, and $MARK-ADI$ would depend on the actual implications of the accesses "invoke", "view", and "alter" in an instance of a ULM. For example, if invocation is a pure- e -access mode, there would be no necessary relationship between $MARK-T$ and $MARK-AI$. On the other hand, if invocation includes r -access, then one would have to have $MARK-AI$ implies (includes) $MARK-T$. Similarly, for T to view the $\{VDI\}$ and to alter the $\{ADI\}$, one needs to assure that $MARK-ADI$ implies $MARK-T$ implies $MARK-VDI$.⁵ If a ULM instance allows a pure- e invocation, then binding of a transaction T to authorized users $\{AI\}$ for the manipulation of $\{VDI\} \cup \{ADI\}$ using only confidence-neutral metadata actions can be accomplished as follows. The $\{VDI\}$ are assigned a lattice-value $a = MARK-VDI$. The transaction T and the $\{ADI\}$ are assigned the lattice-value $a \wedge t$, where $t = MARK-T$.⁶

⁴ This solution of the binding problem is derived from the solutions found in [LEE88], [KARG88], and [SHOC87].

⁵ If the altering access mode implies a viewing capability, then $MARK-T$ would have to imply $MARK-ADI$.

⁶ $MARK-ADI$ becomes $MARK-T \wedge MARK-VDI$.

Authorized invokers have the lattice-value $i = \text{MARK-AI}$ "added" to their lattice-value.⁷ Invocation of T is limited by the presence of i in the invokers' lattice-value. Viewing of $\{VDI\}$ is limited by the condition "lattice-value(subject) implies lattice-value(object)." Alteration of $\{ADI\}$ is limited by the condition "lattice-value(subject) is implied by lattice-value(object)." This transliteration of binding into a non-discretionary setting is sufficient provided that a set of related transactions is intrinsically structured so that the "sensitivity" of the successive transactions T and the various data items used in sequence sort neatly in a monotonically increasing fashion within the lattice. Where that cannot be done (as in rollback below), one must include a notion of transactions as trusted subject, as in [LEE88]. In that case, the transaction is given the ability to view objects with lattice-value MARK-VDI and to alter objects with lattice-value MARK-ADI . This version of binding includes, therefore, a non-confidence-neutral action, the inclusion of a "trusted" subject in the original sense of the term.

Exclusion. The second complex function is exclusion. This can be expressed as the requirement to have the invocation of a bound transaction exclude the invoker from the ability to invoke another transaction. Using the same notation as in the binding discussion, one can restrict an invoker of T_1 from invoking T_2 using either discretionary or non-discretionary features of a ULM. The discretionary approach would be to put the invoker of T_1 onto a NACL for T_2 as part of the execution of T_1 . That is, the action of subject S invoking T_1 would cause T_1 , running as a subject, to set the NACL of T_2 to exclude S from invoking it. This solution, of course, is subject to the safety problem, but one can argue that the safety problem is of lesser importance in the context of invoking bound transactions than in cases where the main concern is the flow of information into or out of the object (in this case, the transactions T_i). The non-discretionary solution is to introduce additional lattice-values $\text{MARK-EXCLUDE-}T_i$ that is added to the invoker's current lattice-value on invocation (that is, the invoker of T_1 has $\text{MARK-EXCLUDE-}T_2$ added to its current lattice-value) and the logic for invocation of T_2 is altered to be "lattice-value(subject) includes $\text{MARK-}T_2$ and does not include $\text{MARK-EXCLUDE-}T_2$ ".⁸

This non-discretionary solution is both aesthetically ugly and probably inappropriate to the actual needs of exclusion. It can be argued that the exclusion of actors in a sequence of transactions is really exclusion from action in a particular chain of data item manipulation rather than an exclusion from action. Thus, the notion of binding the AI only to the T is only acceptable at the lowest level of transaction-chaining. In cases where the transactions interact in chains, one needs a way to indicate the ability of an authorized invoker to supply input data items for manipulation by a bound transaction. Further, these data items for a chain of

⁷ The lattice-value i is added in the sense of having it ANDed onto whatever other lattice-value is already existing.

⁸ Note that this solution is the direct analogue of enforcing informal need-to-know through the imposition of formal categories or compartments.

transactions need assured association with each other.⁹ Using the two transactions T1 and T2 above, a subject *S* in {AI1} would be able to invoke T1 provided *S* had "supply as input" mode to {VDI1} and "invoke" mode to T1. This more natural expression of bound transactions allows a similarly natural representation of exclusion through the addition of NACL alteration to the downstream {VDIn} that are related to the chain at hand rather than a blanket prohibition on the invocation of T2 and any further bound transactions.¹⁰ It will be assumed herein that a discretionary approach to exclusion will be the norm; general permission to invoke a transaction will be limited by non-discretionary lattice-value protection, while transitory tuning of that capability for particular chains of transactions will be provided through the application of NACL's. To recapitulate, one provides exclusion among the bound transactions {T1, . . . , Tn} by adding the requirement that invocation of Tj by subject *S* requires both invocation access to Tj and "supply as input" access to {VDIj}. Successful invocation of Tj includes within its operation the NACLing of *S*, either of all the other {Ti} or of all the data items associated with the chain at hand and labeled {VDIi}. Thus subject *S* will not be able to provide the needed input data items and is excluded from performing more than one of the {Ti} in a particular chain of transactions.

Roll-back. The third complex function is roll-back. This refers to the necessity to un-do the effects of a transaction that has already been invoked. This function is nothing but a special case of bound transaction, one that un-does the actions of a paired bound transaction while recording the fact of roll-back in an unassailable audit record. The unavoidable complication here is that any restoration of input data items required of the roll-back transaction involves confidence-questioning, if only in the sense of having the transaction alter ACL's and NACL's on data-item objects that are earlier in a chain, and hence have "lower" or "sideways" lattice-values.

For a bound transaction T as above, the roll-back transaction RT will be invocable by its own authorized invokers {RAI} (this could be the same set as {AI} but need not be) and its action will be to remove NACL's from the downstream chained-data-items as well as the NACL's on the chained-data-items of type {VDI} for T (the NACL entries that were added in order to prevent a single chain from being treated more than once). The changes required are confidence-neutral, with the exception of the possibility of covert passage of information through the changing of metadata relative to objects at a lower lattice-value than the invoking subject.

⁹ If such assured association is not available, one can provide a work-around by passing along NACL's at each step in the chain rather than globally setting NACL's as described in the text.

¹⁰ It is worth noting that the exclusion concept here is directly related to the notions of "mutual exclusion" mechanisms that grew out of consideration of indivisible operations for use in isolating critical regions of crucial, shared program logic.

N-Person Control. The fourth complex function is n-person control. N-person control refers to the idea of requiring separate agents to cooperate to cause a particular action to take place. The traditional examples are the use of two keys to open a safety-deposit box and to limit the activation of a missile in a silo. In a ULM context, n-person control can be expressed as the need to have n authorized users, each with proper authorization, to jointly cause the invocation of a target action. The discretionary version of this function is provided by the use of ACL's and NACL's and no other ULM mechanism. The usual reason for n-person control makes the limitations of a discretionary solution unacceptable. The non-discretionary solution is a case of n bound transactions mutually excluded preceding a single bound transaction that embodies the protected action. This solution does not deal with the common requirement to have the n authorizations occur within a short, fixed time interval. That detail can be provided with a timed roll-back attached to each of the initiating bound transactions. The confidence implications of n-person control of a non-discretionary sort are the covert information flow concerns of altering metadata of lower lattice-valued objects.

APPLICATIONS TO SPECIFIC "POLICIES"

The application of functions that can be supported in a ULM range from the familiar and obvious to the unexpected. It is a truism, for example, that Biba integrity [BIBA77] can be supported on a ULM through the simple expedient of "turning the lattice upside down". What this means in practice is the alteration of the human-readable version of lattice-values, the most benign version of changing the metadata. The applications to be covered below include multinational sharing of data, Clark & Wilson [CLWI87] with the elaborations of [NAPO90], the Chinese wall policy of [BRNA89], and two versions of Originator Control (that is, ORCON).

Multinational Sharing. Classified information is sometimes shared between allied nations. The classified information of a particular level of sensitivity thus is divided into three subtypes: that shared by the representatives of the two countries and that held separately. As an example, suppose Eire (Ireland) and Lower Volga were to agree to share certain classified information. Then SECRET information dealt with by Lower Volgan and Irish nationals would be termed SECRET EILV Only, while the two nations would label information not to be shared as SECRET LV Nofoin and SECRET EI Nofoin, respectively. A Lower Volgan national cleared to SECRET is allowed to read information designated either SECRET EILV Only or SECRET LV Nofoin; analogously for a Irish national cleared to SECRET. A Lower Volgan is allowed to create SECRET LV Nofoin documents, but has to create SECRET EILV Only documents with adequate care that overly sensitive Lower Volgan information is not inserted into anything shared with Irish nationals. This situation gets more complicated when Yugoslavia enters into bi-lateral agreements with Lower Volga and Eire, as well as initiating a trilateral exchange of information.

This particular type of multinational sharing can be directly supported by a ULM by the expedient of assigning new human-readable forms to available lattice values. In bitmap

terms, one picks three unused categories, p , q , and r . One assigns the bitmap combinations involving p , q , and r as follows:

p	\rightarrow "EILV Only"	p, q	\rightarrow "LV Only"
q	\rightarrow "LVYU Only"	q, r	\rightarrow "YU Only"
r	\rightarrow "EIYU Only"	r, p	\rightarrow "EI Only",

and then associates the null set of categories with the string "EILVYU Only". Each Lower Volgan national is assigned the combination $\{p, q\}$, in addition to the maximum actual security clearance held; each Irish national, the combination $\{r, p\}$; and each Yugoslav national, the combination $\{q, r\}$. Normal operation of the ULM will provide the isolation of national sensitivity as desired. No change to the ULM is required beyond re-assigning the string equivalents of the categories (the lattice values) provided. In this case, just as true for Biba integrity, the solution is implicit in the ULM itself without either discretionary or non-discretionary functional extensions.

Clark & Wilson. The Clark & Wilson transactions TP can be directly implemented on a ULM as bound transactions, with separation of duty being provided by exclusion on related TP's. The dynamism of separation of duty elaborated in [NAPO90] is nothing more than the requirement for roll-back of individual steps in a single chain.

Chinese Wall. The Chinese wall policy of [BRNA89], focusing exclusively on the access of analysts to insider information on various corporate entities within conflict of interest groups, addresses two interesting complications. The first is initial free will in choosing which of a set of restricted data items (relating to a particular company within a conflict of interest group) will be accessed, causing thereafter a prohibition on access to restricted data items concerning other companies within the same group. The second is the interaction with general, open information on companies. This set of policy statements can be expressed as a set of bound transactions. All open information is assigned a lattice-value *OPEN*. Every conflict of interest group is identified as a group of objects. Every company is assigned a lattice-value.¹¹ Every restricted data item is assigned the lattice-value *OPEN* and the mark of the company Ltd, *MARK-Ltd*, to which the restricted data item refers. ACL's and NACL's are initiated with view access granted to all users and with no NACL's at all. Changes to ACL's and NACL's must be limited to the bound transaction described below, except for the usual administrative functions. Every brand-new (viewing) analyst operates at a lattice-level of *OPEN*. Retrieving information requires that the analyst's current lattice-level imply the lattice-value of the data item.¹² Any analyst may request access to any restricted data item. Requesting access to restricted data items is implemented as the invocation of a bound

¹¹ It is assumed for simplicity that every company is in exactly one conflict of interest group. The more complex case can also be treated with a few more lattice-values.

¹² One can view the retrieved information as being put into a read-only workspace.

transaction C. This transaction grants access to the requested data item provided that the subject has *MARK-Ltd* for the company Ltd or if the subject is not listed in the NACL for the data item. If there is no NACL for the subject, the subject's current lattice-value is augmented with *MARK-Ltd* and the subject is entered in the NACL for the conflict of interest group to which Ltd belongs. This solution is a discretionary form. In a manner exactly analogous to the discussion of exclusion above, a parallel non-discretionary solution is possible, using an additional set of lattice-values *NOT-COI*. In that form, access to a restricted data item is approved provided the subject's lattice-value implies *MARK-Ltd* or it does not imply *NOT-COI*, for the conflict of interest group to which Ltd belongs. In the second case, the current subject's lattice-value is augmented by $MARK-Ltd \wedge NOT-COI$.¹³

ORCON. The next application is in the realm of "Originator Control" or "ORCON" of material. For this discussion, the focus will be on groups of individuals representing organizations.¹⁴ Such organizations will be presumed to produce two types of documents, released ones and pre-release material. Released documents correspond to those marked ORCON and the limitation that pertains is not to include or cite the released document or the information without explicit approval. Pre-release material is draft material and not-as-yet released documents. The release of material involves an explicit decision and action to move the report into the category of Released.

Individuals outside of organization Q whose parent organization P has been granted access to a particular released ORCON Q report can read that report. Further, they are allowed to produce draft material based on or including the ORCON information. This preparation of a draft is necessary in order to provide Q organization the context of a request to release the ORCON Q information. In what one could term "single-level ORCON," an agreement by organization Q for organization P to include data or implications from ORCON Q material would allow organization P to release a report listing nothing more than ORCON P. Multilevel ORCON would address the process of releasing material with joint originator controls of the nominative form ORCON P & Q.

Addressing ORCON within a ULM context requires the assignment of lattice-values to subjects and to objects and the provision of actions to cover the release decision and process in a way that preserves the intent of Originator Control. The initial discussion will be limited to single-level ORCON. Identify two lattice-values *DRAFT* and *REL*. (for "released"). Each user is assigned a lattice-value associated with the user's organization (of the form *ORG-P* or *ORG-Q*), the lattice-value *DRAFT*, and other lattice-values related to organizations (of the

¹³ The exclusive focus on receiving information in [BRNA89] makes the treatment here relatively simple. The inclusion of the obviously-needed maintainers of the information being protected complicates the situation significantly.

¹⁴ The case of individuals can of course be included by viewing each individual as a group of one.

form *MARK-P* or *MARK-Q*).¹⁵ A user can run at any level below the maximum level allowable, with the condition that the organizational mark and the *DRAFT* mark must be present. Released material has as lattice-value {*REL*, *MARK-Q*} for the information produced by organization Q.¹⁶ Reading of objects is governed as follows:

- (1) if the object is labeled *REL*, then the portion of the lattice-value of the requesting subject (exclusive of the organizational designator and *DRAFT*) has every lattice-value of the object (exclusive of the *REL* itself), subject to ACL and NACL constraints; and
- (2) if the object is labeled *DRAFT*, then the subject's lattice-value (less the organizational designator) implies the lattice-value of the object, subject to ACL and NACL constraints.

The implication of these conditions is that one can read a released object if all the *REL* lattice-values on the object are part of the subject's lattice-value and one can read a draft object if all the markings on the object are part of the subject's lattice-value. Discretionary controls fine-tune the ability to include or exclude readers.

The logic formulation of the non-discretionary conditions above is as follows:

$$\text{lattice-value}(S) - \{DRAFT, ORG-PARENT(S)\} \implies \text{lattice-value}(O) - \{DRAFT, REL\}.$$

Writing of objects is more restricted:

- (3) the object is labeled *ORG-PARENT(S)* and *DRAFT*, and the object's lattice-value implies the subject's lattice-value.

These restrictions allow the reading of ORCON information released by organizations to organizations, further restricted by ACL's and NACL's on the individual data objects. That reading makes possible the manipulation of material in a "work space" in the same functional way that the preparation of original draft material for release as Originated and Controlled information. In order to meet with the intent of ORCON, however, there needs to be no way that an organization can release material without the explicit approval of the organization that

¹⁵ For this discussion, it is assumed that each organization will mark all its released material with *REL* and a single organizational designator. Finer grained access can be provided with ACL's and NACL's. The ability to use more than one organizational designator complicates the exposition but not the concept.

¹⁶ It will be presumed that *REL* material is only read. Maintenance can be viewed as being done by repeated draft-to-release actions. The addition of the ability to alter released material would require a few more limitations and restrictions.

provided released material in the first place. What is required is a bound transaction with the exclusive ability to perform releases.

A release, in this context, is the removal of the lattice-value *DRAFT* and the substitution of the value *REL*. What is required is a non-controvertible and unavoidable ability to create a copy of an existing object with a different lattice-value attached to it; further, it must be controlled by the originator. As a simple example, suppose subject *S* from the *P* organization has used material marked $\{REL, MARK-Q\}$ in the preparation of a draft report *R* now marked $\{DRAFT, ORG-P, MARK-P, MARK-Q\}$. An authorized individual from the *P* organization needs to approve the re-marking of *R* to $\{REL, MARK-P\}$. At the same time, an authorized individual from the *Q* organization needs to approve that same re-marking. This complex function is 2-person control of a copy-sideways transaction. In this case, there is no overlap between the authorizers of the two halves of the pre-copy step. Thus it suffices to use bound transactions, one for *P* organization approval, one for *Q* organization approval, one to impose 2-person control on the final bound transaction, and the final bound transaction *T1* itself, that effects the sideways copy.¹⁷

Interestingly, multilevel ORCON can be handled in exactly the same way, using a similar final bound transaction *Tm*. The only difference between *T1* and *Tm* is that *Tm* copies sideways from $\{DRAFT, ORG-P, MARK-P, MARK-Q\}$ to $\{REL, MARK-P, MARK-Q\}$. Moreover, the extension from a bilateral decision between two organizations to a multilateral decision requires only the substitution of an *n*-person control front-end in place of the 2-person control described.

Other treatments of ORCON include [GRAU89], [McMN90], and [AELO90]. Those treatments and the one here are complementary, in the following manner. [GRAU89] and [McMN90] illustrate that the usual system-level technical policy mechanisms for discretionary and non-discretionary access control do not patently match the enterprise-policies for ORCON. Both propose conceptual mechanisms (PACL's and ORAC's, respectively) that better match enterprise-policy ORCON, as well as support other needs. The independent control of PACL's and ORAC ACL's by different agents is a feature not covered in the treatment here.¹⁸ [AELO90] provides a taxonomic and analytical tool for the consideration of enterprise-policies with an eye towards the implications of implementation. This treatment addresses the conceptual match of a ULM's intrinsic technical-policy mechanisms to ORCON (viewed as an enterprise-policy). In a sense, this approach is inductive and short-term: how can current and existing concepts be used now and what documented needs are outside the scope of current conceptual technology? The other work is deductive, constructive, and mid-

¹⁷ Clearly roll-back can be used to extricate the system from an anomalous state when there is a difference of opinion about the release.

¹⁸ One should note that the combining function for ORAC ACL's is a logical AND. Thus, ORAC ACL's cannot directly support an enterprise-policy of the form "either the Comptroller's Office or the Personnel Office can authorize a person's access to that type of dossier".

to long-term: what new concepts are needed? how can more efficient and simpler solutions be brought to fruition? The complementary use of both approaches is clearly what is needed in order to address current needs as best one can while assuring that better analytical tools and enterprise-policy support will be available in the future.

FUTURE DIRECTIONS

There are several areas of investigation and work that merit further attention. One is further analysis of enterprise policies. The initial treatments here should be used as a basis for complete analysis and proof-of-concept prototyping. That exercise should help clarify which extended ULM functionality should be addressed and implemented directly so as to realize benefits of simplicity and performance. A similar analysis of other enterprise policies should also be undertaken. Taken as a whole, these analyses should help delineate the truly different enterprise policies from the only apparently different ones.

Another area that deserves attention is the match between actual trusted computer systems and the features postulated for ULM's. The facilities for groups of subjects and groups of objects, especially the setting of ACL's and NACL's and the addition and deletion of items from groups, are not fully realized in all implemented trusted systems. To the extent that those features of ULM's provide a necessary flexibility, those features, or equivalent ones, will have to be conceived and implemented.

One topic of this sort that requires further attention involves the operational embedding of trusted subjects into a system with previously established confidence. In most of the simple situations, where various bound transactions are largely independent, one can often embed the required policy without the need for trusted subjects. Even in those cases where roll-back or convoluted data references force a trusted subject, it is usually the case that the exact functionality required of these trusted subjects is to cause an exact copy of an object to be created at a lattice-value that is not at-or-above the working lattice-value of the subject. That observation raises the possibility that provision of a "trusted copy" or a "trusted append" operation to a trusted system might allow for containment of the confidence-shaking that one will experience when inserting a trusted subject into an operational system.

A particularly important example of needed ULM functionality relates to the size of lattices that need to be supported. As was noted in [LEE88] and elsewhere, the use of lattice-policy mechanisms can require the use of enormous numbers of lattice-values. The fact (cited in the Annex) that the size of the full logic-lattice on n policy alphabet letters is $2^{2^{**n}}$ is confirming in that regard. But it must be remembered that isomorphism results are implacable in the sense that implementation of a policy that is demonstrably a lattice-policy as if it were not a lattice-policy does not allow one to escape the size implications. A general-purpose implementation of such a policy will be a lattice-policy implementation no matter what. Thus the lattice explosion to very large lattices is intrinsic to the problems being addressed.

Given the fact that the lattices will be very large, it is worth recalling the latent lesson of the traditional method of implementing the traditional lattice policy as a direct product of two separate lattices. Attention should be directed towards the ability to implement lattice-policy mechanisms in a way that allows the conceptual use of many different lattices, combined as a Cartesian product, rather than forcing the use of a single lattice. One can imagine, for example, a sensitivity label as consisting of a list of lattice-values of the following form: (lattice-21, value-18), (lattice-77, value-54), (lattice-116, value-42).

SUMMARY

The use of policy commonalities in the form of policy isomorphism and policy conversion logic can be an important force, both in the analysis of proposed and mandated enterprise policies and in the selection of lattice-based features and mechanisms for initial implementation or optimization in trusted systems of the future. The broad applicability of this perspective has been demonstrated by the analysis here of a wide variety of enterprise policies in terms of the lattice-based policies enforced by Universal Lattice Machines.

References

- [AELO90] M. D. Abrams, K. W. Eggers, L. J. La Padula, and I. M. Olson, "A Generalized Framework for Access Control: An Informal Description," *Proc. 13th National Computer Security Conference*, Washington, D.C., 1-4 October 1990, 135-143.
- [ANDE72] J.P. Anderson, "Computer Security Technology Planning Study," ESD-TR-73-51, Vol. I, AD-758 206, ESD/AFSC, Hanscom AFB, MA, October 1972.
- [BELL90] D. E. Bell, "Lattices, Policies, and Implementations," *Proc. 13th National Computer Security Conference*, Washington, D.C., 1-4 October 1990, 165-171.
- [BELL86] D. E. Bell, "Secure Computer Systems: A Network Interpretation," *Proc. Second Aerospace Computer Security Conference*, McLean, VA, 2-4 December 1986, 32-39.
- [BLP75] D. E. Bell and L. J. La Padula, "Secure Computer Systems: Unified Exposition and Multics Interpretation," MTR-2997, The MITRE Corporation, Bedford, MA, July 1975. (ESD-TR-75-306)
- [BIBA77] K. Biba, "Integrity Considerations for Secure Computer Systems," The MITRE Corporation, Bedford, MA, April 1977.

- [BIRK48] G. Birkhoff, **Lattice Theory** (1st ed.) American Mathematical Society: Ann Arbor, Michigan, 1948.
- [BRNA89] D. Brewer and M. Nash, "The Chinese Wall Security Policy," *Proc. 1989 Symposium on Security and Privacy*, Oakland, CA, May 1989, 206-214.
- [CLWI87] D. D. Clark and D. R. Wilson, "A Comparison of Commercial and Military Computer Security Policies", *Proc. 1987 IEEE Symp. on Security and Privacy*, 27-29 April, 1987, Oakland, CA, 184-194.
- [CLAR90] D. D. Clark, private communication, April, 1990.
- [GRAU89] R. Graubart, "On the Need for a Third Form of Access Control," *Proc. 12th National Computer Security Conference*, Baltimore, MD, 10-13 October 1989, 296-303.
- [HRU76] M. A. Harrison, W. L. Ruzzo, J. D. Ullman, "Protection in Operating Systems," *Comm. ACM* 19, 8 (August 1976), 461-471.
- [KARG88] P. Karger, "Implementing Commercial Data Integrity with Secure Capabilities," *Proc. 1988 Symposium on Security and Privacy*, Oakland, April 1988, 130-139.
- [LEE88] T. M. P. Lee, "Using Mandatory Integrity to Enforce 'Commercial' Security", *Proc. 1988 IEEE Symp. on Security and Privacy*, 18-21 April, 1988, Oakland, CA, 140-146.
- [McMN90] C. J. McCollum, J. R. Messing, and LA. Notargiacomo, "Beyond the Pale of MAC and DAC — Defining New Forms of Access Control," *Proc. 1990 Symposium on Security and Privacy*, Oakland, May 1990, 190-200.
- [NAPO90] M. Nash and K. Poland, "Some Conundrums Concerning Separation of Duty," *Proc. 1990 Symposium on Security and Privacy*, Oakland, May 1990, 201-207.
- [SHOC87] W. R. Shockley, "Implementing the Clark/Wilson Integrity Policy Using Current Technology," *Proc. 11th National Computer Security Conference*, 17-20 October, 1987, Baltimore, MD, 29-37.
- [STER91] D. F. Sterne, "On the Buzzword 'Security Policy'," *Proc. 1991 Symposium on Security and Privacy*, Oakland, 20-22 May 1991, 219-230.
- [TCSEC85] *Department of Defense Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD, December 1985.

ANNEX — POLICY CONVERSION.

The issue of treating policy conversions at the lattice-theoretic level is parameterized by those implementations that are of interest. Three obvious candidates are (1) the traditional combination of a totally ordered set and the powerset of a given set; (2) abstract data types with some form of comparison (for example, domination, covering, meet, or join); and (3) undefined symbols combined into logical formulas using AND, OR, and NOT. The first category represents the usual implementation for classified governmental practice, clearances, classification and formal compartments. It is in a sense an "installed base", both of policy perspective and of policy implementations. The second category, abstract data types, represents security levels as opaque, uninterpreted tokens with explicit operations available for manipulation, specifically the operations of comparing for equality and for dominance. The third category is uncommon, but seems to hold great promise for being able to represent a wide variety of narrative policies directly.

In this paper, attention will be limited to the first and third categories. The problem to be solved is how to represent a partial order of the traditional trusted-systems sort in pure-logic terms, and, conversely, how to represent pure-logic in terms of a traditional partial order represented as a characteristic function on a set of elements, usually in the form of a bitmap.

The basis for conversions from pure-logic to bitmaps is contained in the following result:

(Thm 11) The meet-irreducibles of the boolean lattice on the alphabet A are

$$\bigwedge_{a \in A} s(a), \quad \text{where } s(a) \text{ is either } a \text{ or } \neg a \text{ for } a \text{ in the alphabet } A. \quad [\text{BIRK48, p. 163}]$$

Hence the concern ". . . [that] the policy conversion code (which will have to be inside a Trusted Computing Base) could become intricate and possibly of some size" raised in [BELL90, p. 168] proves to be unfounded.

Let A be a set of uninterpreted symbols and refer to A as the "policy" alphabet. By (Thm 11), the minimal boolean lattice including A has as its meet-irreducible those wff's¹⁹ that consist of the meet of exactly $|A|$ wff's, each one of which is either an element a of A or the negation ($\neg a$) of an element a of A . Inasmuch as each meet-irreducible is equivalent

¹⁹ "wff" is a "well-formed formula", a syntactically correct sequence of symbols from the alphabet A and the special set of symbols $\{ \wedge, \vee, \neg, (,) \}$.

to a characteristic function for the powerset of A , the number of meet-irreducibles is $2^{|A|}$ and the total number of elements in the lattice is $2^{2^{|A|}}$.²⁰ [BIRK48, p. 163]

The embedding of a policy alphabet A with operations (\wedge, \vee, \neg) thus proceeds by allocating $2^{|A|}$ meet-irreducibles, associating each one with one of the elements of (Thm 11), and associating with each element in the resulting lattice a reduced version of the meets of the constituent meet-irreducibles. As an example, let $A = \{a, b, c\}$. The set of meet-irreducibles of the generated lattice L is as follows:

$$\{ a \wedge b \wedge c, a \wedge b \wedge \neg c, a \wedge \neg b \wedge c, \neg a \wedge b \wedge c, \\ a \wedge \neg b \wedge \neg c, \neg a \wedge b \wedge \neg c, \neg a \wedge \neg b \wedge c, \neg a \wedge \neg b \wedge \neg c \}.$$

The 0 element of L is *false*, or $a \wedge \neg a$; the 1 element is *true*, or $a \vee b \vee c$. The elements a and $a \vee \neg b$ are the elements

$$(a \wedge b \wedge c) \vee (a \wedge b \wedge \neg c) \vee (a \wedge \neg b \wedge c) \vee (a \wedge \neg b \wedge \neg c) \\ \text{and } (a \wedge b \wedge c) \vee (a \wedge b \wedge \neg c) \vee (a \wedge \neg b \wedge c) \vee (a \wedge \neg b \wedge \neg c) \\ \vee (\neg a \wedge b \wedge c) \vee (\neg a \wedge \neg b \wedge c),$$

respectively.

The reverse embedding, putting traditional compartments into a pure-logic context, is even easier. It is in fact the special case mentioned before. Let the set of compartment names be the policy alphabet. The embedding above applies.

One should note here that the hierarchical portion of traditional sensitivity labels has not been treated explicitly. Since a totally ordered set can clearly be embedded in a boolean lattice (with height equal to the cardinality of the totally ordered set; or with a set of meet-irreducibles with one fewer elements than the cardinality of the set), this omission is only apparent. Further, the usual practice of treating the traditional lattice as the cross-product of two lattices, the totally ordered hierarchy and the non-hierarchical compartments, points out that the existence of an embedding into a single lattice does not necessarily argue for a conceptual or implementational superiority of a single-lattice perspective.

²⁰ If the policy statements do not require \vee and \neg (or \wedge and \neg), then the size of the required lattice can be reduced considerably. In fact, in those cases, one can use the lattice with $\neg A$ or A as the set of meet-irreducibles, respectively. The first case corresponds to the traditional non-hierarchical-compartments situation for classified information.

Reconciling CMW Requirements with Those of X11 Applications

Glenn Faden

*Sun Microsystems, Inc.
Mountain View, CA 94043*

© 1991 Sun Microsystems, Inc.

ABSTRACT

This paper discusses some of the issues in meeting the Compartmented Mode Workstation (CMW) requirements while still supporting commercial applications. The reader is assumed to have a general familiarity with CMW and window systems. The security policy is summarized, and followed by a discussion of how it has been interpreted in the real world of X11 programming. Applying restrictions to the X protocol prevents clients from interfering with each other, while still providing enough functionality to make these programs useful. Special considerations are given to the root window, selections, grabs, and atoms to meet the needs of existing applications.

Keywords: *Systems Application - Secure Architectures; X11 Window System™; CMW*

"The essence of security is telling lies; the art of security is ensuring that it is done by suppressing the truth rather than by inventing falsehoods."

David Rosenthal

1. Introduction

The SunOS™ CMW Window System provides the user interface for SunOS CMW. All user interaction with the system is initiated through the window system. The window system allows the user to perform multiple tasks concurrently, and to operate at multiple sensitivity levels in a single login session. The window system must be trusted to provide the necessary mandatory and discretionary access controls (MAC and DAC) described in [1], and to provide a *trusted path* by which users can be assured that they are communicating with trusted applications. The window system is based on the Sun's OpenWindows, and supports both the X Window System protocol and Sun's NeWS protocol. Only the X11 protocol [2] has been modified to meet the CMW requirements; the NeWS protocol which is based on the PostScript language, is reserved for privileged clients.

Many papers have been written about the lack of security in the X Window System [3, 4, 5]. Since X was designed with insufficient mechanisms for enforcing security, programmers have had to rely on conventions such as those described in the Inter-Client Communications Manual (ICCCM)[6] and those provided by various toolkits, to provide some order in the X environment. Many of the solutions proposed for making X more secure have the unfortunate side effect of limiting the number of applications which will run without modification. When at-

tempting to support Commercial Off the Shelf (COTS) applications, requiring even minor modifications become impractical. So we are left with the problem of trying to provide adequate security while imposing the fewest restrictions on existing protocols and conventions.

The problems needing solution are:

- to protect the data displayed by subjects and entered by users from being read or modified by subjects based on MAC and DAC policies.
- to prevent clients from interfering with the security policy which includes the normal operation of certain trusted clients like the window manager and the selection agent.

Unfortunately the X protocol and the conventions of most toolkits provide some very thorny problems in meeting these goals.

2. An X11 Overview

The *server* provides the basic windowing mechanism. It handles client connections, demultiplexes graphic requests onto the screens, and multiplexes input back to the appropriate clients. It directly controls the keyboard, monitor, and pointer. A *client* is an application program

connected to the window system server by an interprocess communication path. The program is referred to as a client of the window system server.

The X protocol deals with objects known as resources which are maintained in the address space of the window server. Some of these objects are created automatically by the server, and others are created in response to requests from clients. The standard X protocol imposes very few restrictions on access to these resources, and they can normally be modified or destroyed by any client connected to the server. Included in the list of X objects are:

Window	A <i>window</i> is an abstraction of a displayable region on the workstation screen.
Pixmap	A <i>pixmap</i> is a three-dimensional array of bits. A pixmap is normally thought of as a two-dimensional array of pixels, where each pixel stores an N-bit value, where N is the depth of the pixmap. Both windows and pixmaps are referred to as <i>drawables</i> .
Property	Windows may have associated <i>properties</i> , each consisting of a name, a type, a data format, and some data. They are intended as a general-purpose storage and intercommunication mechanism for clients.
Atom	An <i>atom</i> is a unique ID corresponding to a string name. Atoms are used to identify properties, types, and selections.

These resources are uniquely identified by numbers known as XIDs which are used by the clients and the server in protocol requests, replies, and events.

The X protocol also provides synchronization primitives for clients to take control of certain resources. These are known as *grabs*. Protocol requests exist to grab the keyboard, individual keys, the pointer, or the server. When the keyboard is grabbed no other clients can receive keyboard input. When the pointer is grabbed, no other clients can receive motion events or button press events.

3. The Security Model

The trusted version of the X11/NeWS Server is responsible for implementing most of the CMW security policy for the window system. It is analogous to the UNIX kernel in that it maintains information and performs services on behalf of many clients. The basic security model is defined in terms of subjects and objects. In the X Window System subjects are clients of the window server, and ob-

jects are the resources maintained by the server. The following security attributes are associated with clients in SunOS CMW:

Sensitivity label	A classification and compartments set that is used as the basis for mandatory access control decisions.
Information Label	A classification, compartments set, and markings set that is used to represent the actual classification and required handling of the data with which it is associated.
User ID	An integer which uniquely identifies a user. The server may share the screen with multiple users.
Privileges	A set of rights granted to a process to perform actions that would otherwise be prohibited by the security policy.

The SunOS CMW Window System maintains a sensitivity label, an information label, and a user ID, for those resources that are created on behalf of clients. Normally, when a resource is created, the client's sensitivity label and user ID are applied to the resource. Access to these resources are controlled by the server according to the following policies:

- A client cannot access any resource whose sensitivity label is not dominated by that of the client.
- A client cannot modify any resource whose sensitivity label is different from that of the client.
- A client cannot access any resource whose owner is a different user from that of the client.
- Resources that are created automatically by the server during initialization are publicly accessible to all clients, but may not be modified by them.
- Appropriately privileged clients may violate any of these policies.

In addition to sensitivity labels, the server also maintains an information label on each object that can be modified by ordinary clients. The information label is initially *system low*. When an object is modified as a result of a client request, the client's current information label is floated up with the previous label of the object, to form a new la-

bel. When an object is read by a client, the information label of the data associated with the object is passed back to the client and conjoined with the client's process information label.

The mechanism for passing security attributes is implemented on top of the UNIX socket mechanism, and is known as Trusted Sockets [7]. The decision was made to rely on Trusted Sockets, rather than extending the X protocol to pass additional state information on each request. Changing the X protocol was rejected because it would restrict interoperability to those clients that were recompiled with a non-standard X library. Furthermore, clients could bypass any such code that was placed in the X library.

For window objects there are some extra considerations for information labels. Since windows are maintained in a hierarchical tree, the top of each client window subtree has two special information labels. The first is the *display* information label which is the conjunction of all the information labels of the windows in that subtree. The other is the *input* information label which is used to label keystrokes which originate from any window in that subtree.

4. Trusted Clients

Although the server is responsible for most of the access control decisions in the window system, it does not determine the user interface or the conventions for interaction between clients. There are a small number of privileged clients that comprise the *Trusted Computing Base* (TCB) user interface. These clients are responsible for implementing specific CMW requirements, and are generally independent of each other. This modular approach allows the clients to perform their functions with a minimal set of privileges and to be customized without affecting other TCB components. Other CMW systems have implemented all of these functions into the window manager [8], or into a Security Services Client [4].

4.1 Logintool

Logintool is the first component of the SunOS CMW Window System to execute. It is started by *init*, and in turn starts the X11/NeWS server. After identification and authentication of the login user, the *session clearance* is determined from the intersection of the user's default clearance and the maximum sensitivity label of the workstation. Once the user is given the chance to further restrict the default session clearance, a new session is established, and the rest of the privileged clients are started by **logintool**. These include the window manager and the selections agent, among others. These clients are critical to proper operation of the window system. The system can be administered to cause an automatic logout if any of these privileged clients exit.

Logintool also starts a user thread of execution for unprivileged processes to be initiated with the user's environment. The processes started from this thread are started with the same sensitivity label as the user's home directory. This label is called the *session low* label.

4.2 The Window Manager

The window manager is based on *olwm*, the OPEN LOOK Window Manager in Open Windows. In addition to the functions normally performed by a window manager, *olwm* displays a CMW label (a combined information and sensitivity label) for each top-level window, which accurately reflects the data in the window (and subwindows), and displays an input information label for each top-level window. See FIGURE 1.

Along the bottom of the workspace, the window manager maintains a dedicated region which is known as the *Screen Stripe* (See FIGURE 2.). This area extends the full width of the workspace and is not movable, resizable, or obscurable. Its background color is distinguished

FIGURE 1. The Trusted Window Frame

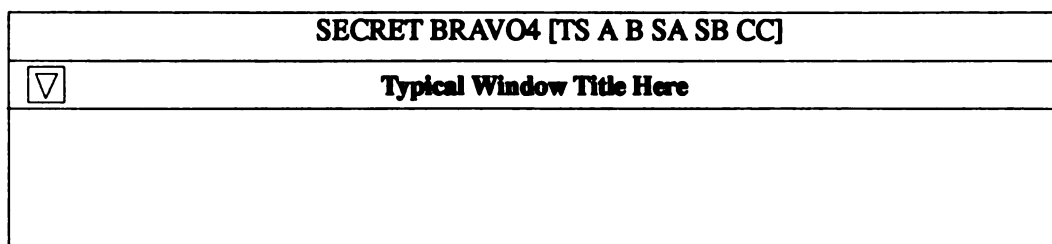








FIGURE 2. The format of the Secure Screen Stripe

		CONFIDENTIAL NOFORN
		CONFIDENTIAL PROJECT X LIMDIS D/E NOFORN [TS A B SA C]

from the background color assigned to other objects rendered by the window manager. It provides a number of trusted path functions:

- it displays the input information label of the window which has the keyboard focus.
- it displays the CMW label of the window associated with the pointer (either the window containing the pointer, or the pointer grab window).
- it alerts the user to active pointer grabs and keyboard grabs.
- it provides confirmation that the user is communicating with the trusted path by displaying a *trusted chevron* .
- it provides an area on the screen from which the trusted path menu can always be invoked.

Applications which provide components of the trusted path inherit a unique privilege from **logintool**. The window server distinguishes all of the X resources created by such clients by setting a flag in each resource XIDs. Then the window manager examines this flag to determine whether a window is associated with a trusted client.

When the pointer is in a component of the trusted path, the *trusted chevron*  appears in the lower left portion of the screen stripe. When an untrusted client has affected an active pointer grab, the *pointer grabbed icon* appears. Normally this field is blank.

When the keyboard is associated with a trusted path window, the *trusted chevron* appears in the upper left portion of the screen stripe. When an untrusted client has affected an active keyboard grab, the *keyboard grabbed icon* appears. Normally this field is blank.

There are two choices for controlling the input focus: click-to-type and focus-follows-mouse. When focus-follows-mouse is selected, the input information label is updated whenever the pointer crosses a window boundary.

In click-to-type mode, the user must explicitly set the focus using the **SELECT** pointer button before the input information label is updated.

4.3 The Selection Agent

The selection agent is another trusted path component. It is responsible for mediating inter-window data moves, such as cut and paste or drag and drop operations. The window server has been modified to redirect all selection requests to the selection agent which then performs the following trusted functions:

- it prevents unauthorized transmission between clients.
- it provides an interactive confirmer for setting the information label of pasted data.
- it identifies the selection holder and the selection requestor.
- it provides a window to view the data before accepting it.

The selection agent implementation supports arbitrary types of selections including text, graphics, and binary data. This provides greater support for COTS applications.

4.4 The Workspace Menu Manager

The workspace menu manager is a privileged program which may be started from a user shell. It provides the interface by which the user may launch user programs at sensitivity labels between the session low label and the session clearance. This program runs with the user's environment so that it may pass arbitrary parameters to any program it launches. Programs may be launched automatically from a start up file, or from a menu. Both the start up file and the menu are user configured.

5. Special Considerations

Some resources and protocol requests present special problems in terms of their effect on security and compatibility. The following discussion provides examples of how we resolved these problems. Refer to [2] for additional background information.

5.1 The Root Window

The root window doesn't actually belong to any client, although the window manager controls some of its behavior. It is a public object, which by our definition would not be writable by unprivileged clients, but would be readable by any client. What we really care about, however, is not whether anyone can write it, but whether another client can detect that it has been written. Furthermore, we have tried to avoid returning errors for operations which we can safely ignore. Since some clients will terminate on a error replies, it is our goal to return the fewest errors necessary.

We make a distinction between the window attributes, which are protected at *system low*, and the actual drawable, which is protected at the session clearance. Letting any client draw on the root window (via **PutImage** or **PolyLine**) does not violate security, as long as we prevent the client from writing into any subwindows by enforcing **ClipByChildren** mode. When a lower level client attempts to read back the root window (via **GetImage**), it is given a constant value of all zeroes.

Protecting the root window drawable at the session clearance also solves another problem. Since all windows regardless of their sensitivity label are children of the root window, the **GetImage** must be constrained from returning those images of windows which the client does not dominate. The production of such a sanitized view is a difficult task, and is rendered unnecessary by this security policy.

Creating subwindows of the root window can be likewise unrestricted, provided that other clients can not discover the existence of such windows. The **QueryTree** function has been modified to hide this information.

The server can ensure that the trusted screen stripe window cannot be obscured, but clients must receive some help here as well. The window manager can ensure that the windows it manages are placed appropriately. Override redirect windows, such as pop-up menus, should not be clipped, and therefore the screen height returned in the

display structure (via **XOpenDisplay**) and the apparent root window height returned by **GetGeometry**, is reduced by the height of the trusted screen stripe.

In order for clients to receive events associated with windows, they must express interest in specific event classes via **CreateWindow** or **ChangeWindowAttributes**. We restrict modifying the interest mask on windows that the client cannot write. Expressing interest on the root window would normally be okay, because clients cannot discover the interest masks of other clients. Unfortunately, many clients send events (via **SendEvent**) to the root window, which in turn would then be delivered to any client who had established a matching event mask on the root window. Rather than limiting the delivery of events based on MAC and DAC of the event, we have chosen to prevent clients from establishing interest masks on the root window as well. This does not affect most COTS applications.

The policy for creating properties on the root window is restricted to normal clients whose sensitivity label is equal to the session low label. Before the selection mechanism was developed, X clients implemented cut and paste operations by modifying **CUTBUFFER** properties on the root window. This mechanism is now obsolete and clients should rely on selections instead.

5.2 Grabs

Some CMW systems restrict clients from grabbing the keyboard or pointer [5], except as passive grabs. The restriction is enforced because users may be unaware that their keystrokes are being redirected. However, this policy limits the number of COTS programs that will run without modification in the CMW environment. Grabbing the keyboard is commonly done in notices, where a carriage return can be used to select the default action, and normal input must be suppressed until the user responds. This is sometimes called a modal dialog.

SunOS CMW allows keyboard grabs by implementing the following policy:

- Keystrokes are not delivered unless the sensitivity label of the grab window dominates the input information label of the focus window.
- Keyboard grab states are displayed in the trusted screen stripe. The server reports keyboard grab status changes to the window manager so that this state is continuously displayed to the user.

- A grab interrupt key is provided so that the user may terminate excessively long or errant grabs. This grab interrupt key has precedence over a normal keyboard grab.

When the server is grabbed (via **GrabServer**), the server operates on behalf of the grabbing client, deferring requests from other clients. Server grabs can be used to guarantee that the screen will not change during a sequence of requests, or that certain state information will remain consistent. There are quite a few problems that server grabs represent:

- First of all, a client can prevent the user from performing any further operations on that workstation. If the client cannot be killed by some external means such as remote login, the system must be rebooted.
- Second, the user cannot easily determine which client is doing the server grab, because the server does not report status, nor accept queries from any other client, including the window manager which maintains the viewable system status in the screen stripe.
- Third, there is no way to force a time-out on server grabs.

Therefore, we have made the server grab a privileged request. Normal clients who attempt to do a server grab will not be informed of the failure, however, because there is no error defined for this condition in the X protocol. Therefore, COTS applications will not be affected overtly. However, there may be some side effects which result from denying the **GrabServer** request. In practice, these differences have proven to be minor.

5.3 Override-redirect Windows

Menus and other pop-up windows are usually implemented as *override-redirect* windows. This means that the window is not reparented by the window manager and does not receive the standard window border that is applied to other top-level windows. Its label is displayed in the screen stripe whenever it grabs the pointer, or the user places the pointer over the window.

In SunOS CMW, an override window cannot get keyboard input except by issuing a keyboard grab. It does not have its own input information label, and cannot be used to set the input focus via **SetInputFocus**.

5.4 Transparent Windows

Normally windows are filled with a client specified background before they can be drawn upon. However, a client may specify a null background whereby the pixels that were previously rendered by another client are left unchanged. This allows the client to create visual effects like transparent shadows or windows which appear to have irregular shapes, such as the OPEN LOOK notice window. Unfortunately, these windows may contain data from clients whose label dominates the client owning the window. To prevent unauthorized data flow, the server prevents clients from reading back the pixels of transparent windows.

5.5 Selections

A selection agent is necessary to provide a way for clients to complete such operations as cut and paste. In order to preserve isolation, clients must not be able to get attributes which can be set by other clients. For example, a client may establish itself as the owner of a selection via **SetSelectionOwner**. If another client queries the owner of the selection, the server will respond with the redirected window of the Selection Agent.

When a client requests a copy of a selection (via **ConvertSelection**), the request is redirected to the selection agent, which in turn gets a copy from the real selection holder. The selection agent (with input from the user) determines whether the requested data can be copied on to the requesting client window property, and completes or denies the original request. Although the default time-out provided by most X toolkits is too short to allow enough time for the user to examine the data, apply a label, and confirm the request, this can usually be customized by the user. If the toolkit does not provide a resource to allow the user to lengthen the time-out period, the utility of this mechanism is severely limited.

Some toolkits use multiple **ConvertSelection** requests to pass attributes as well as data in cut and paste operations. The user can configure the selection agent to automatically confirm the transmission of these attributes, subject to the limits of the covert channel policy.

5.6 Atoms

During each login session, the window server keeps an internal list of all predefined atoms and their corresponding name strings. Clients can make a new entry (by using **InternAtom** request) in the server's internal list, creating a new atom and specifying the name associated with this atom.

The client-created atom will remain defined even after the client who defined it has exited. Therefore, the ownership information about which client created an atom is not valid after the client has disconnected from the server. In the SunOS CMW Window System, there is no ownership information associated with an atom. Moreover, once an atom is created, the window server does not provide any request to change the string associated with an atom identifier. Thus an atom can be considered as a read-only object after it is created.

To prevent information from being passed through atom names, some CMW systems polyinstantiate atom IDs at each sensitivity level at which they are interned. As an alternative, we can associate with each atom all the sensitivity levels at which it has been interned. Clients cannot read the string associated with the atom (via `GetAtomName`) unless their sensitivity label dominates a label at which the atom was interned. Therefore the ID for any atom string can be a constant within the login session; trusted clients, like the selection agent, don't have to do translations from one labeled name space to another.

5.7 Connection Access Control

The SunOS CMW Window System enforces two restrictions with respect to connection requests:

- The user ID of the client must be in the access list provided by the login user.
- The sensitivity label of the client must be dominated by the session clearance of the login user.

Because the system requires the use of Trusted Sockets to connect to the window server, we do not rely on any special authentication scheme, such as Kerberos or MIT-MAGIC-COOKIE-1. In the CMW environment, the network is assumed to be trusted, and the user credentials supplied by the socket services are used without authentication by the X server. Instead, we provide a trusted tool that allows the user to grant or deny access to individual users based on the username. This server access control list is maintained by using the existing host list protocol (`ChangeHosts`), except a new address family is used to specify the valid usernames. Since the security attributes are retrieved from the connection as socket attributes, the client does not need to be aware of the access control mechanism. No changes are required to the X library and the existing address families, Internet, DECnet, and Chaos are simply ignored.

In addition, privileged clients may set the label of their connection and assert privileges via the connection. The X server allows clients which have asserted appropriate privileges to bypass certain security policies.

6. Conclusion

While there are quite a few additional considerations for applying security to the X protocol, the issues discussed here are representative of the general solution. Although [1] imposes severe restrictions on the X11 environment, it is still possible to run many COTS applications. By careful analysis and flexibility, Sun has met the security goals and still provided enough compatibility with existing conventions and toolkits, so that the commercial goals have been met. The application of the security policy has been carefully applied to each protocol request and each object to prevent the policy from becoming unnecessarily restrictive.

References

- [1] Security Requirements for System High and Compartmented Mode Workstations. John P.L. Woodward, MITRE MTR 9992 Revision 1, DIA Document Number DDS-2600-5502-87, November 1987.
- [2] Xlib Programming Manual, volume 1 & 2. A. Nye. O'Reilly and Associates, Inc., 1988.
- [3] LINUX - a Less INsecure X server. David. S. H. Rosenthal, Sun Microsystems, Inc., April 1989.
- [4] Trusted X Window System, Volume 1: Design Overview. J. Picciotto, MITRE, February 1990.
- [5] An X11-based Multilevel Window System Architecture. Mark E. Carson, Janet A. Cugini, IBM
- [6] X Window System: Version 11 - Inter-Client Communications Conventions Manual. David S.H. Rosenthal, MIT X Consortium, January 1989.
- [7] Trusted Interprocess Communication. SecureWare.
- [8] CMW+ Trusted Facilities Manual. SecureWare, January 1990.
- [9] A Proposal for an X11 Protocol Extension to Implement B1/CMW Secure Workstations. E. Cande, Digital Equipment Corporation, January 1990.

- [10] X11/NeWS Design Overview. R. Schaufier, Sun Microsystems, Inc.,
- [11] Xlib - C Language X Interface, Protocol Version 11. R. W. Scheifler, R. Newman, J. Gettys, Massachusetts Institute of Technology, September 1987.
- [12] X Window System Protocol Specification, Version 11. R. W. Scheifler, Massachusetts Institute of Technology, September 1987.
- [13] The X Window System. R. W. Scheifler, J. Gettys, ACM Transactions on Graphics, Vol. 5, No. 2, April 1986, Pg. 79-109.

Trademarks

UNIX and OPEN LOOK are trademarks of AT&T. The X Window System is a trademark of MIT. PostScript is a trademark of Adobe Systems. SunOS, OpenWindows and NeWS are trademarks of Sun Microsystems.

RESTATING THE FOUNDATION OF INFORMATION SECURITY

**Donn B. Parker
SRI International
Menlo Park, California 94025**

INTRODUCTION

Information security is unlike other information technology disciplines yet its development has progressed as if it were the same, addressing purely technical issues. Other disciplines in information technology seem to have no devious potential adversary, save the usual complexity and problems in logic. In security, however, we must add the challenge of active, unpredictable human adversaries accidentally or intentionally causing failures and losses in systems. Adversaries have great freedom in attempting to achieve their often-changing goals. Technologists and systems managers who are inexperienced in loss events and untrained in security must nonetheless protect assets—including new assets—created by users and fixed in time, place, and form, often with little correct intelligence information about adversaries' plans or actions or about users' needs for protection.

Consequently, security technologists and architects tend to use their information systems, technical background, and knowledge to create static, artificial, predictable, and sometimes loose systems of controls that protect against only those adversaries they can imagine and against limited, idealistic attacks, ignoring the remaining huge array of possible adversaries and their methods of attack. For example, an espionage agent whose motivation is hatred for his target might fail in his attempt to obtain and disclose secret information in a Trusted Computer Systems Evaluation Criteria (TCSEC)¹ B2 rated system but then change his goal and proceed to do far more harm by destroying the information instead. The system design protected against the loss of secrecy but not the destruction of information, loss of availability, or damage by failure of authorized persons to act when necessary.

The TCSEC Trusted System Criteria (Orange Book), the only formalized system of security controls for confidentiality that exists, is an example of this narrow technological approach. This set of criteria is also the only known one that meets the U.S. National Security Agency (NSA) mission goal of protecting the secrecy of information in multilevel security military systems (the mission of protecting military systems from fraud and loss of integrity and availability is the responsibility of other military organizations such as intelligence, police, or audit). The Orange Book deviates from the common definition of data integrity by defining it as "The state that exists when computerized data is the same as that in the source documents and has not been exposed to accidental or malicious alteration or destruction." The U.S. Federal Information Processing Standard 73² uses a similar definition that is in disagreement with the common meaning, which is limited to wholeness and completeness, not conformance to fact or reality.

Another example is the "Draft #2 July 23, 1990, Guidelines and Recommendations on Integrity" prepared for the Third Integrity Workshop, September 26, 1990, at the National Institute of Standards and Technology (NIST)³. This is one of the best available documents on integrity, yet its contents are still flawed by the failure of attendees at the first two workshops to reach agreement on a definition of integrity. Unfortunately, the attendees decided to narrow the definition to focus on one component of integrity that all could agree upon and move ahead toward the solution in the hope that a better definition would evolve. The component of integrity they chose was "ensuring that data changes only in highly structured and controlled ways"—an approach and definition that pose serious fundamental problems.

In the first place, security cannot ensure (make certain or guarantee) anything. The limits of security are to preserve information assets from loss and to do what the owners want done if their information is attacked. Ensuring, however, is the role of owners, system designers, implementors, managers, users, and quality control engineers. Second, integrity deals with wholeness and completeness, not accuracy, correctness, and precision resulting from direct or indirect change. Information could lose its integrity completely yet not change at all if, for example, the specifications of wholeness and completeness change. There are two separate purposes to control the change of information. One is to preserve its intrinsic form, completeness, and wholeness for integrity. The other is to preserve its extrinsic state of conformity to fact and reality for authenticity. Finally, the definition of integrity is faulty because it states only how to achieve integrity but does not actually say what integrity is.

The basic problem is that integrity is only one of two important security attributes to consider. The other attribute is authenticity, meaning conformance to fact and reality, of undisputed origin, and genuine, or true. If the Third Integrity Workshop had addressed this duality, then one component of preserving both integrity and authenticity would definitely be allowing users to change information only in highly structured and controlled ways to preserve its completeness and genuineness. However, this component would still be deficient because it would not include anticipating changes to external conditions or values that cause a loss of authenticity.

The remainder of this paper points out the looseness, inconsistency, and lack of completeness that typify even major efforts (heroic as they may be) to specify major systems of security controls or to reach practical and consistent answers about the nature and scope of information security. If such limitations can be overcome in other disciplines of information technology, why not in security? The answer is that an operating system, for example, is acceptable if it mostly or partially works; partial success, however, is not enough in security where the adversaries are intelligent humans, not technical barriers. If it is to succeed, security must be sturdy and have a tight enough weave to avoid fatal flaws that skilled and determined adversaries can exploit. It is measured by its weakest point. In contrast, an operating system is measured by the sum of features that work correctly minus the sum of its misoperating features. An operating system with flaws may still be acceptable.

RESTATING THE FOUNDATION OF INFORMATION SECURITY

The generally accepted definition of information security is preservation of confidentiality, integrity, and availability (some say continuity instead of availability) of information. In addition, information losses are generally said to be from modification, destruction, disclosure, and use. The four losses loosely match with the three attributes in obvious ways (i.e. confidentiality with disclosure), and these attributes and types of loss have for many years seemed complete, comprehensive, and adequate.

These three attributes, the four types of loss, and the methods of achieving and preserving security are in fact incomplete, dangerously simplistic, and inconsistent; consequently, they are failing. Their widespread acceptance has led to an inability adequately to define the terms used, as the Introduction indicates. Their use (as well as technological limitations and special interests of sponsoring organizations) has led to suboptimization of security and makes secure system models such as the Trusted System Criteria and Clark-Wilson Integrity Model⁴ difficult to match with the attributes and especially with the types of losses. Such deficiencies make systems that use the policies in these models obvious targets for misuse outside of their intended narrow and incomplete single-attribute purposes, for example, causing loss of availability of a TCSEC-evaluated secrecy system or violating the confidentiality of information stored in a Clark-Wilson Integrity system.

Another deficiency relates to availability. Availability refers only to being present or accessible for use and does not encompass usefulness and fitness for a purpose. Therefore, another attribute is necessary to preserve usefulness of information as well as its presence.

The solution is to expand the three attributes of security by adding authenticity (genuineness, conforming to fact, or correct) and utility (fitness for a purpose) to the original three—integrity, confidentiality, and availability—and to replace totally the loss types with the inverses of the attributes, for example, loss of integrity. Computer systems must then maintain all five attributes at an acceptable, evaluated level to be secure in any combination of attributes or any single attribute. This would avoid the current attempts to extend the common meaning of integrity to include accuracy and correctness and availability to include usefulness. The remainder of this paper supports these proposed changes on the basis of actual and anticipated loss experience and of generally accepted definitions, showing the appropriateness, applicability, consistency, comprehensiveness, and benefits of the proposed new attributes of information security and loss.

The results of this effort should advance toward solutions of the definitional and scoping problems in the information security field, provide a truly comprehensive identification of all known types of information losses to be protected against, and aid in producing comprehensive, consistent, and more effective systems security policies and models. Authenticity and utility add explicit preservation objectives of conformance to fact and fitness to function, respectively, to each of the five levels of abstraction—information, applications, operating system, hardware (including communications hardware), and information workers (users). A matrix providing specific definitions, control examples, and types of loss for each combination of attribute and level appears in Table 1. In this table, each level (column) represents an object with the property of the designated attribute (row); for example, an application loses confidentiality as a result of piracy or espionage, and the specified controls protect the confidentiality of the application. Another useful matrix would treat each level except information as a subject acting to preserve the attribute of the lower levels to the left in the matrix, treated as objects; for example, an application protects the confidentiality of information by limiting distribution of data to programs on a strict need-to-use basis.

Other approaches to the foundation issue are less formal. One concludes that the whole subject is relatively unimportant and is at the level of how many angels can dance on the head of a pin. In this approach, the objectives of security are open-ended; confidentiality, integrity, and availability are acceptable descriptors if one interprets them in very general ways (e.g., integrity includes correctness and therefore the addition of authenticity is unnecessary). Another approach is conservative in concluding that security should limit itself to attributes that have yes or no achievement answers, and that one should ignore correctness and conformance to the real world requiring user or owner judgment until the more intrinsic purposes are accomplished. This conservative approach would result in rejecting the Clark-Wilson Integrity Model, leaving the model to the accounting profession to implement, and concentrating on preserving the authenticity of information but not its conformance to requirements outside of computer systems. Such an approach would cover theft of information but not fraud. This paper rejects these approaches in an attempt to achieve a comprehensive, definitive foundation or at least make progress toward that goal.

Security practitioners could apply the conclusions in this paper to help ensure that the tests they are applying to their risk assessments, security reviews, controls selection, and implementation are truly anticipating the complete, material range of threats and countermeasures. For example,

Table 1
LEVELS OF ABSTRACTION ADDRESSED BY SECURITY ATTRIBUTES

Security Attributes for Preservation	Information	Applications	Operating System	Hardware	Users
Confidentiality	Known to only a limited few. A set of rules used to determine whether subject has access to an object.	Limited access to code and use.	Limited access to code.	Limited observation of devices, specifications.	User-controlled access and use of private personal information.
Control examples	Labels, encryption, DAC, MAC, reuse prevention	Copyright, patent, encryption, test and production separation, physical-access locks, labels.	Copyright, patent, labels, physical-access locks.	Patent, trade secret, physical-access locks, serial numbers, protective packaging.	Law, preauthorization for release of information, due notice.
Loss examples	Disclosure, inference, espionage	Piracy, trade secret loss, espionage	Piracy, espionage	Espionage, theft, reverse engineering	Violation of privacy
Authenticity	Genuine, accepted by conformity to fact, valid	Genuine, unquestioned origin	Genuine, unquestioned origin	Genuine, unquestioned origin	Unquestioned identity
Control examples	Audit log, accounting controls, verification, validation, dual control	Vendor assurances, pedigree documentation, maintenance log, hash totals, serial checks, escrow	Vendor assurances, pedigree documentation, maintenance log, hash totals, serial checks, escrow	Vendor assurances; labels, serial numbers; physical protection; locks; fire, shock, water protection	Passwords, tokens, biometrics
Loss examples	Replacement, false data entry, change, failure to act, repudiation, deception, misrepresentation, fraud	Piracy, misrepresentation, replacement, fraud	Piracy, misrepresentation, replacement, fraud	Replacement, change, sabotage	Misrepresentation, impersonation
Integrity	Unimpaired, complete, whole	Unimpaired, all present, ordered	Unimpaired, all present, ordered	Unimpaired, all present	Honest, forthright, loyal, ethical
Control examples	Hash totals, check bits, sequence numbers, missing-data checks	Hash totals, pedigree checks, escrow, vendor assurances, sequencing	Hash totals, pedigree checks, escrow, vendor assurances, sequencing	Redundancy, vendor assurance, testing, inventory	Separation of duties, background investigation
Loss examples	Larceny, taking, changing, fraud, concatenation	Theft, fraud, change, concatenation	Theft, fraud, change, concatenation	Sabotage, parts theft	Negligence, injury, ethical violations
Utility	Fit, created for use	Fit, executable	Fit, executable	Workable, assured of action	In good mental and physical health, motivated, assured of action
Control examples	Testing, backup, recovery plans	Testing, maintenance	Testing, maintenance	Testing maintenance	Medical exams, environment protection, safety
Loss examples	Change, sabotage, converted	Sabotage	Sabotage	Change, failure, sabotage	Accidents, injury, injunction
Availability	Present, accessible, usable when and where needed	Usable when and where needed, accessible	Usable when and where needed, accessible	Usable when needed, accessible	Present when and where needed
Control examples	Redundancy, backup, recovery plan	Escrow, redundancy, backup, recovery plan	Escrow, redundancy, backup, recovery plan	Physical access assured, resources (power) protected	Physical protection, communication, transport available
Loss examples	Failure to provide or act, sabotage, larceny	Larceny, failure to act, interference	Larceny, failure to act, interference	Sabotage, larceny, access destroyed	Disappearance, kidnapping, hostage

adding utility extends their charter to help preserve the usefulness of computers as well as their availability. Freedom from the constraints of the four old types of loss from modification, destruction, disclosure, and use encourages practitioners to consider real additional threats such as sabotage by workers who fail to act when required or maliciously conform to the rules, fraud by misrepresentation or repudiation, and denial of computer services by slowing responsiveness or prolonging the response. Forcing integrated attributes into systems of controls policies will accelerate our advancement beyond the narrowly defined TCSEC and Clark-Wilson criteria to computer products that are secure from the full range of threats, not just from subsets that adversaries can easily subvert by changing goals to attack where controls are lacking.

This paper presents each of the attributes separately in more detail, with discussions on how to integrate them, put them in a more appropriate priority order, evaluate them in security reviews, and use them better to categorize types of losses. However, security policies, criteria, system models, and architecture require modification to account for the extended attributes and address them in integrated fashion to achieve practical and effective information security.

Confidentiality

Considerable knowledge about confidentiality and how to protect it is available, at least within simple computer systems if not yet within networks of computers. Achieving confidentiality requires numerous controls that have other purposes as well, such as authenticity. Starting with the various dictionary definitions (on the assumption that security definitions should at least be consistent with traditional definitions), we have no trouble with confidentiality: the state of being private or secret, known to only a limited few. This definition is totally consistent with the best computer systems criteria, TCSEC and the Bell La Padula model⁵ that implements the U.S. Department of Defense secrecy policy.

Disclosure is well accepted as the primary violation of confidentiality. However, a person may think certain information is confidential, but if someone has changed the information it may have lost confidentiality because the change signaled an application program to drop protection. Modification of confidential information, or lack of modification, may violate confidentiality without actually disclosing the information. Thus confidential information that someone has secretly changed may no longer be confidential because it is different. In addition, some would say that deriving information about confidential information or its use, or making inferences about that information without disclosure—such as from traffic analysis—is loss of confidentiality, although the loss may not be obvious. Therefore, we cannot limit confidentiality to the threat of direct disclosure.

Application of the need-to-know principle is the accepted determination of confidentiality. Users and system processes should receive only the information necessary to do their jobs, and in the default or starting state they possess no information. This is appropriate in a military type of system, in which military rules of order prevail and which assumes continual threat from an adversary with unlimited resources. However, in business environments, values of trust, ethics, and friendliness prevail to increase profits, productivity, and growth and are usually more important than confidentiality. The need-to-know principle here may be stifling and inappropriate. In a business environment, the inverse principle—need-to-withhold—is often more appropriate, with a default or beginning state in which all information is freely available. Only the small amount of information that is truly confidential is withheld and available only to a few.

The conclusion is that confidentiality is well understood and that the dictionary definition is adequate and consistent with security purposes. Both principles of need-to-know and its inverse,

need-to-withhold, meet confidentiality requirements. However, the threats to confidentiality go beyond the most obvious, disclosure type of loss to encompass an open-ended set.

Authenticity

Authenticity is, by common definition, the state of being true, real, or genuine; worthy of acceptance by reason of conformity to fact and reality; of unquestioned origin; not copied, original; properly qualified; possessing authority not open to challenge. Authenticity of information refers to its extrinsic correct or valid representation of that which it means to represent. Timeliness of information is an authenticity issue since conformity to fact and reality would preclude obsolete information that no longer represents present reality. A representation of money will show a sufficiently accurate and precise amount (number of digits) to be accepted as correct in whatever national monetary units. A program is authentic if one can trace its provenance back to include the original copy and can show authorization for all changes. Hardware is authentic if it comes from the vendors specified. People are authentic if they can prove that they are who they claim to be. None of the definitions use the synonyms correct or accurate. However, inclusion of the idea of conformity to fact and reality seems sufficient reason to adopt the words correct and accurate as synonyms of authentic. Thus authenticity refers to extrinsic states whereas integrity refers to intrinsic states of information, applications, operating systems, hardware, and users.

Integrity

Integrity is the most difficult concept to consider because it is already in common (but incorrect) use in information security as encompassing both intrinsic and extrinsic states. However, by severely limiting the definition of integrity to "being in an unimpaired condition, sound, adhering to a code of values, a state of completeness and wholeness," we can assign the word its proper place among security attributes. Applied to information, it means that all of the information is present and accounted for (not necessarily accurate or correct). Integrity thus plays a more limited role and need not be the subject of a separate system policy as confidentiality is. This confirms the idea that the Clark-Wilson Model is primarily a policy preserving authenticity and integrity of information rather than integrity alone. Integrity has a greater role to play at system-policy and user-policy levels of abstraction than at the information level.

Information integrity means that when a user moves or communicates information, that information starts and ends in the same state of wholeness, unimpaired condition, and completeness. No parts of the information are missing or concatenated, encrypted, or converted in any unanticipated ways. These considerations are independent of whether the information is correct (authentic) at any given time. The information could lose authenticity if a user failed to make corrections to it between the beginning and ending states, yet it would conserve its integrity by remaining whole, unimpaired, and complete.

The International Standards Organization (ISO) recognizes authenticity and integrity as separate attributes in the Open System Interconnection Basic Reference Model⁶ by defining authenticity as assurance that the data source is the one claimed (correct). Integrity is defined as assurance that the data sent and received are the same (nothing says or implies that the data are necessarily correct) without insertions, duplications, modifications, or resending.

Information integrity controls consist primarily of checks for intrinsic problems of missing data in fields, records, and files; checks for missing fields, records, and files of variable length and number; and check bits and bytes, hash totals, and message and transaction sequence checks. At higher levels, integrity concerns completeness, compatibility, consistency of performance, history, failure reports, and communication between levels. At the user level, honesty,

forthrightness, loyalty, and ethics are key integrity factors. A background investigation of a person's past helps determine his or her integrity.

Utility

Utility means the state of being useful or fit for some purpose, designed for use or performing a service. The preservation of utility for security purposes becomes quite clear with this definition. Information utility means that information must be useful, for example, in such a way that it may be tested to determine if it is authentic or has integrity. Information has utility if it is in a directly useful form (e.g., expressed in the expected units of dollars, not yen). One way to sabotage a system of accounts while preserving the other attributes of integrity, authenticity, availability, and confidentiality would be secretly to convert all U.S. monetary values to the correct values in yen. Utility loss can occur—and availability remain untouched—when information is encrypted, a source program is without a compiler, hardware has no power, or a computer user is too tired to work.

Availability

Availability is the least understood and most ignored purpose of security, with the exceptions of recovery planning and backup. Formal security policies usually omit it. And in the form of recovery planning as a means of restoring information services it often appears as separate from security. In a broader context, this is the arena of business resumption planning. Availability is defined as the state of being present, accessible, or obtainable; capable of use for a purpose, immediately utilizable. The difference from utility (usefulness rather than usability) is significant; availability means that something is usable for a specific purpose at a specific place and time and is ready for immediate use, independent of whether it is useful.

The primary controls that help preserve availability are redundancy, data backup, and protection from physical harm. The application of these controls is usually external to systems and therefore is usually missing from the security aspects of systems architecture or design. Exceptions are built-in hardware redundancy, applications checkpoint restart capability, and electronic vaulting of data. The design of operating systems also facilitates the backup, physical removal, and restoration of data. Availability may become a more inclusive purpose in systems security policy as systems assume more control over their physical environments and redundancy and backup can become more automatic, as with electronic vaulting.

INTEGRATING THE ATTRIBUTES OF INFORMATION SECURITY

Security consists of preserving the attributes of confidentiality, integrity, authenticity, utility, and availability and the unique, nonoverlapping values each represents. Security technologists attempt to categorize controls for a specific purpose, such as by the use of passwords for authentication, cryptography for confidentiality, and systems of controls such as the TCSEC criteria for confidentiality and the Clark-Wilson Model for integrity and authenticity. Such categorization, however, may suboptimize the security of a system for only one or two attributes, whereas adversaries' strategies in attacking systems are not so constrained and leave the systems even more vulnerable to attacks on the missing attributes.

Each control has several purposes, direct or indirect, for protecting information, systems, or people (e.g., TCSEC includes some integrity value, helps assure availability, etc.). This also means that the Information Technology Security Evaluation Criteria⁷ (ITSEC, the new draft criteria in Europe) is faulty in dividing its functionality by attribute. For example, Tandem has been forced to seek both the German F2 Confidentiality and F7 Availability categories of evaluation for their Nonstop System product. But the product possesses other security purposes

of integrity and authenticity that the evaluation does not recognize. One result is that these evaluations aid attackers by guiding them to attack evaluated systems for purposes and in ways in which the systems have not been evaluated. Systems must be evaluated and rated secure to varying degrees in all five respects to demonstrate their complete range of security or lack thereof.

Authenticity of a computer program or system means that all its desired features (vendor, color, size, contents, parameters, code, constants) are acceptably correct. Utility means that when in use it works in some useful way, although not necessarily correctly. Integrity means it is all there, with parts, documentation, code, etc. in the right order or position. Availability means that it is in the right location or that someone in a desired location can use it. Confidentiality means that unauthorized persons don't know about its existence, about certain of its qualities, or about certain contents (when it may be okay to know about its existence). Each security attribute applies unique values that the others do not provide. If any attribute is missing or lost in the face of any material threats, security will be deficient. The addition of any other specific attribute, such as reliability or auditability, will alter security, will not focus on protection from loss, or will encroach on other subjects.

Although defining security policy for one attribute to be preserved may be useful for theoretical purposes, in practice system security must be seamless in all aspects because adversaries are likely to switch from one goal to another. The overall purpose is to protect from all kinds of loss, and means of protection from loss are so diverse that controls overlap in purpose.

The TCSEC focuses almost entirely on confidentiality because that is the mission of its creators. Only a small serendipitous integrity and authenticity value is apparent. The Clark-Wilson Integrity Model calls for a TCSEC Trusted Computer Base almost as an afterthought and doesn't specify how to integrate it, although others have discussed this issue^{8, 9, 10}. However, the definition of security must incorporate all attributes simultaneously to guard effectively against any adversary who can (and likely will) switch from one goal to another in attacking a system. The information security community must reach agreement on generally accepted models and develop prototypes for integrated attribute system policies before evaluation criteria at the level of TCSEC and ITSEC can be successful.

Should integrity include authenticity, as current practice assumes? The primary reason for not doing so is that events could occur in which integrity could be preserved and authenticity lost (or the converse) in a way that has security implications. If the two exist in combination, the loss of one aspect of integrity or of authenticity would cause the entire attribute to be lost. Therefore, the two attributes must be separate. Similarly, should availability include utility? The answer is the same; usefulness and usability are different, and one can exist without the other.

Priority Treatment of Attributes

A listing of the five attributes in order from most to least attention received would show confidentiality first, and then availability, authenticity, integrity, and utility. Confidentiality is first because of the massive efforts by the U.S. NSA to preserve military and diplomatic secrecy, which resulted in the Orange Book and TCSEC evaluation program. In addition, civil rights advocates have worked intensively to preserve personal privacy, thus increasing the need for confidentiality. Availability is next because of the great efforts of users and the service industry to provide recovery of information and information services. Although these availability efforts may be of poor quality in many cases, they are quite widespread. Authenticity, integrity, and utility are mostly achieved in application controls that derive from traditional accounting practices, and many security practitioners consider them routine and mundane. This is unfortunate and short sighted.

Security attributes should appear in the order of general importance to encourage more prudent attention. This order would be availability, authenticity, integrity, utility, and confidentiality, even for military and diplomatic purposes. If information and services are unavailable, all else is immaterial, and no other security problem exists. Therefore, the first order of business of any information security effort is at least to provide adequate backup copies of information, alternate services, and contingency plans for unavailability of services. Next, information and services must be genuine, then complete, and then useful in that order. If information and services are not available and genuine, then whether they are complete is immaterial. The user can often make information complete and restore it to usefulness by increasing the effort or using additional means; the loss of these attributes is therefore often retrievable.

After all these attributes are achieved and preserved, the relatively small amount of information that is secret is worthy of and could benefit from confidentiality controls. This order of importance is as applicable to personal information and military and diplomatic information as to other information. The value of the small amount of secret information is often not totally lost if confidentiality is lost, although in some cases the consequences of lost confidentiality could be devastating.

Many examples of information and services require a different order of priority of security attributes than the canonical one proposed above. However, for general applicability and in the absence of special conditions, the proposed order makes sense at least from the intrinsic security perspective as an end in itself.

The Security of Security

No information or mechanism in a system is more sensitive than the security controls. Therefore, in addition to the usual performance goals such as applicability, throughput, cost effectiveness, reliability, and speed of calculation, security controls must meet the security attribute requirements as well. For example, the reuse of residual information control that invokes confidentiality and has an impact on availability and utility must itself possess the security attributes in its design, development, implementation, use, and maintenance. It must be available at the proper time and place, have utility for all its purposes and for no unauthorized purposes, be complete in meeting all its specifications correctly (have integrity and authenticity), and satisfy any secrecy of mechanism and applicability requirements. These attributes also apply to the entire information security activity in an organization, including its loss experience information files, security standards, and security meetings. Security controls do not represent a level of abstraction in terms of information, applications, operating system, hardware, and users since controls exist at all of these levels. Therefore, their inclusion as another column in Table 1 is not appropriate, which is why the need for the security of security controls appears here as a separate topic.

TYPES OF LOSS

Loss is as complex as all human thought and action that might occur accidentally or intentionally to cause the loss; it is the stuff of human history as well as of detective fiction. Means of causing loss can be categorized only incompletely in ways that always leave the next possible uncategorized means of causing the next loss a mystery¹¹. While technologists were trying to solve the Trojan horse attack problem, the virus variation became the next problem, followed by the worm. What new intensely publicized, intellectually stimulating crimoid will appear next, the weasel? There will always be a new problem to face¹².

Security is most often defined as protection from specific threats of modification, destruction, disclosure, and use (or denial of use) that cause losses. However, security must also address many threats not included in or not obviously derived from these four acts. Repudiation, replacement, misrepresentation, taking, failure to act, malicious conformance, intimidation, renaming, delay or prolongation of use, and inference are also threats. Dangerous suboptimization occurs when security is restricted to defense against only four of the possible threats. Security practitioners attempt to relate each of these four threats to each of the three to five attributes preserved by security. Unfortunately, this also fails (e.g., modification can destroy confidentiality and availability as well as integrity and authenticity). The four threats have internal redundancy as well (e.g., modification could be destruction and vice versa).

The only comprehensive and unambiguous way to state all types of losses is in terms of the loss of availability, authenticity, integrity, utility, and confidentiality of information. We must abandon the current limited description of losses in terms of modification, destruction, disclosure, and use.

The Coverage of Threats Problem

The most difficult and critical problem in protection of information systems is how to specify prudent controls that reduce all material threats to acceptably low levels of likelihood of occurrence—the coverage problem. Controls must cover all material threats, because adversaries will tend to cause losses where coverage is the weakest or does not exist, since their objective is to make desired gain (cause loss) in the most effective and safe ways they can with minimal or most attractive type of effort. Therefore, coverage will be measured by its single greatest deficiency rather than the sum of its weaknesses, because the purpose of security is to prevent any intolerable single loss. (Many tolerable losses could occur where protection is not worth the effort, constraints, or cost.)

The best way of reducing the coverage problem is to know as much as possible about threats and about controls, in order to match them up to achieve best coverage. The objective is to discover coverage deficiencies (vulnerabilities) before any adversaries take advantage of them. Some deficiencies can also be discovered and corrected before some adversaries with limited capabilities act, thus reducing the number of effective adversaries. Identifying a consistent and complete set of threats and a complete and well-ordered set of controls is the most critical aspect of achieving coverage. This can best be done by having a complete and consistent set of security attributes and by labeling the threats with the same attribute names as the categories of controls (e.g., authenticity loss and authenticity controls).

INDEPENDENCE AND COMPLETENESS OF ATTRIBUTES

This expansion of the information security foundation advances independence of meaning among all five attributes and absence of any overlap or obvious incompleteness (leaving vulnerabilities unaddressed at any level of abstraction). Each attribute can be applied independently, but all should be present where general risks of loss exist (e.g., authenticity can be lost whether integrity is preserved or not). However, preserving one attribute may preclude, at least partially, the preserving of another. For example, the requirements for data base authenticity may be in conflict with the requirements for confidentiality. Confidentiality requires that users be denied read access to data for which they have no authorization. However, authenticity constraints can be defined over data in different security access classes in such a way that information from one class may leak over into another. For example, if an authenticity constraint requires that changes to data at one class reflect indirectly in the value of data at another class, then the authenticity constraint mechanism could be used as a covert channel (because varying data in one class could

make detectable changes to data in another class). The solution requires a compromise in the security policy or a redefinition of the security requirements^{13,14}.

Synonyms for the attributes help to demonstrate their relationships:

Availability	presence
Authenticity	genuineness
Integrity	completeness
Utility	usefulness
Confidentiality	secrecy

Examples of how the loss of a single attribute could occur without the loss of the others provide further insights. A loss of each attribute of my birth date (10-9-29) could be as follows:

<u>Attribute</u>	<u>Loss</u>	<u>Comments</u>
Availability	<u>10 9 29</u>	Misplaced
Authenticity	<u>11 8 30</u>	Wrong
Integrity	<u>10 29</u>	Incomplete
Utility	<u>10 9 4627</u>	Partly useless
Confidentiality	<u>10 9 29</u>	Visible

The enclosures indicate that secrecy controls apply. The usual U.S. rules apply to the format and coding. The year 4627 is valid in the Chinese lunar calendar.

Do other attributes require preservation for security? Accuracy and precision come under authenticity. Auditability might be a candidate; however, auditing as in monitoring, analyzing performance, or testing seems to be a second-order function or control acting to preserve the five attributes rather than being another one. Also, all of the attributes seem necessary in achieving auditability. Continuity seems crucial in achieving and preserving each attribute; timeliness and veracity are aspects of authenticity. I initially chose functionality, the state of being usable, as an attribute, but finally it seems to be a part of availability, whereas utility, the state of being useful, seems to be distinct from availability.

A difficulty in adding authenticity to the list of attributes is that this term already has a traditional use. Authenticity has referred exclusively to authentication of users. However, nothing precludes its application to new uses in security. Authenticity still applies to identity of users, but it now will also apply to applications, operating systems, controls, and information. All levels must be authentic as well as possess intrinsic values of integrity and utility, and security must preserve all five attributes to achieve protection from accidental or intentional loss.

MEASURING INFORMATION SECURITY

Security is said to be for reducing loss and the risks of loss. Security actions can also be functionalized to avoid, deter, prevent, mitigate, detect, recover from, apply sanctions against, transfer (insure against), and correct loss and the risk of loss. In addition, security has the equally important objective of meeting a standard of due care, although this may be argued to be the reduction of the risk of negligence. Meeting a standard of due care is at least as important as reduction of risk, when risk of loss exists in both cases. Due care (or diligence) means that one has employed controls deemed prudent by a sufficient number of others in similar circumstances or employed controls that are sufficiently available, low in cost, and meet intended objectives. In addition the standard of due care includes the option of bypassing the use of accepted controls by establishing prudent reasons before the fact.

At least four different standards of due care exist for legal, professional, insurance, and functional purposes. The functional purpose is a catchall that could include avoidance of embarrassment, avoidance of social or business dysfunction, or avoidance of economic or human loss. Meeting the legal standard of due care means winning or avoiding negligence or liability litigation and avoiding government penalties. The professional standard of due care exceeds legal requirements and requires meeting a code of ethical conduct as stated by a professional society or association or employer. The insurance standard of due care means ability to transfer loss to an insurance company.

Measuring security by risk of loss is less important than meeting a standard of due care (avoiding negligence) in achieving security and justifying adoption of controls and systems of controls. The quantification of risk for security purposes is generally not possible anyway, since sufficient loss experience is lacking for valid statistically based prediction and its application to a specific instance. "We don't know what we don't know."

CONCLUSIONS

At the earliest opportunity, all organizational policies, standards, guidelines, reports, and training materials should change to reflect the new foundation. The new five attributes are easier to remember when expressed as three, with only one small compromise of priority: 1) availability and utility, 2) integrity and authenticity, and 3) confidentiality corresponding roughly to control of existence, change, and access. Information security reviews or audits should test for each vulnerability coming from a lost attribute individually in this order (e.g., lack of controls to adequately preserve availability, especially while the remainder of the review or audit effort is going on, then authenticity, and so on). This process will encounter absence or presence of many of the same controls repeatedly, but that is expected since a vulnerability or a control will have direct or indirect impacts on the preservation of all five attributes. In fact the same vulnerability or control can have a negative impact on preserving one attribute and a positive impact on preserving another (e.g., encryption can preserve confidentiality but preclude utility or obscure the determination of whether integrity or authenticity has been preserved).

Determination of the full value of a control feature, control product, or system of controls is not possible until its contribution to or detraction from preservation of all five attributes has been considered. Otherwise, the values of controls are not fully realized and an injustice to both supplier and user would result. Therefore, the TCSEC and ITSEC single functionality (attribute preservation) approaches to evaluation are flawed unless only one functionality is desired. However, preserving only one functionality such as confidentiality makes no sense in view of adversaries' strategies.

Evaluation procedures for each control feature or product and system of controls should be on the basis of the full range of functional values (attributes.) A set of statements of functional values is necessary for all five attributes for each object of evaluation. However, this still leaves unanswered the question of how to achieve overall protection from all material threats. Not all threats and their impacts leading to a loss of one or more attributes are known. For example, failure to act as required when processing information is a common form of sabotage, and all threats in terms of failures and results of failures will never be known. Secondly, no one-to-one relationship exists among threats, information assets, and safeguards or controls. Therefore, we do not know how to cover or counter all material, likely threats with controls. And finally, we do not know the extent of constraints and cost of controls the stakeholders will be willing to tolerate.

Finally, losses are so diverse and resistant to complete identification that their definitions are better left as simply the failure to preserve each of the five attributes (i.e. loss of confidentiality rather than disclosure, since other forms of loss of confidentiality, such as inference, exist).

Categorizing loss in sufficiently complete fashion has never been achieved on the basis of functional acts that cause loss, except at levels of great generality [e.g., reading, writing, appending at the lowest level to fraud, larceny, conspiracy, espionage at the highest (criminal) level]¹¹. In summary, security practitioners must address the achievement of a standard of due care and reduction of loss and the risk of loss of availability and utility, authenticity and integrity, and confidentiality by avoiding, deterring, preventing, detecting, mitigating, recovering from, sanctioning against, transferring, and correcting information losses.

ACKNOWLEDGEMENTS

A growing number of people have been contributing ideas for this paper either through wholehearted support for its objectives or out of frustration from or disagreement with my approach to the problems and solutions. A few I can recall include William Murray from Deloitte and Touche, Robert Courtney, Dr. Willis Ware from the RAND Corporation, Dr. Peter Neumann at SRI International, Dr. Paul Karger from the Open Software Foundation, Dr. Fred Cohen, Bruce Shiotsu at Lockheed, and Will Ozier from Ozier Perry and Associates. All have been helpful.

REFERENCES

1. "Trusted Computer Systems Evaluation Criteria," DOD 5200.28-STD, U.S. Department of Defense, December 1985.
2. "Federal Information Processing Standard Publication 73, Guidelines for Security of Computer Applications," U.S. Department of Commerce, National Institute of Standards and Technology, June 1980.
3. "Draft #2 July 23, 1990, Guidelines and Recommendations on Integrity" prepared for the Third Integrity Workshop, NIST, September 26, 1990.
4. David Clark and David Wilson, "A Comparison of Commercial and Military Security Policies," *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, April 1987.
5. D. E. Bell and L. J. La Padula, "Secure Computer Systems: Unified Exposition and Multics Interpretation." EDS-TR-75-306, The MITRE Corporation, Bedford, MA, March 1976.
6. ISO International Standards Organization, "Security Architecture," Part 2 of 4, *Information Processing Systems Open System Interconnection Basic Reference Model, ISO-7498-2*, available from the American National Standards Institute, New York, 1989.
7. "Draft Information Technology Security Evaluation Criteria," Herausgeber: Der Bundesminister des Innern, Bonn, May 1990.
8. P. A. Karger, "Implementing Commercial Data Integrity with Secure Capabilities," *Proceedings of the 1988 IEEE Symposium on Security and Privacy*.
9. T. M. P. Lee, "Using Mandatory Integrity to Enforce 'Commercial Security'," *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, Oakland, California, April 1988.
10. W. R. Shockly, "Implementing the Clark/Wilson Integrity Policy Using Current Technology," *Proceedings of the NIST/NCSC 11th National Computer Security Conference*, Washington, D.C., October 1988.

11. Peter G. Neumann, "Rainbows and Arrows: How Security Criteria Address Computer Misuse," *Proceedings of the NIST/NCSC 13TH National Computer Security Conference*, Washington, D.C., October 1990.
12. Donn B. Parker, "Trojan Horses, Viruses, and Other Crimoids," Third Annual Computer Virus Clinic, Data Processing Management Association, New York, March 1990.
13. Private note from Bruce Shiotsu to Donn B. Parker, December 1990.
14. Sushil Jajodia, "Tough Issues: Integrity and Auditing in Multilevel Secure Databases" (Executive Summary), *Proceedings of the NIST/NCSC 13th National Computer Security Conference*, Washington, D. C., October 1990.

THE ROLE OF NETWORK SECURITY IN A METHODOLOGY FOR INFORMATION SECURITY DESIGN AND IMPLEMENTATION.

**Prof. J.H.P. Eloff. Department of Computer Science. Rand Afrikaans University
P.O. Box 524. Johannesburg South Africa.**

FAX : 27-11-489-2138. TEL : 27-11-489-2842.

**Mr. A.J. Nel. Department of Computer Science. Rand Afrikaans University
P.O. Box 524. Johannesburg. South Africa.**

FAX : 27-11-489-2138. TEL : 27-11-482-2190. E-MAIL : Awie.Nel. f1.n7101.z5.fldonet.org.

ABSTRACT

This paper aims to address the complicated issue of network security. Network security is approached in this paper from a functional and more specifically from a methodical point of view. A network can be seen as a collection of nodes that are connected by communication media in such a way that information can be transferred between nodes. The connection must be such, that resources like databases and printers may be shared between the nodes. A node can be any computer hardware component and may even be another network like a local area network or a wide area network. Network security needs to be addressed from a technical as well as from an application systems perspective. The technological issues on network security addresses the broader concepts such as the provision of a variety of network security services and mechanisms for example authentication and traffic analyses. Applications network security assures that a network oriented application system be designed to adhere to the computer security policy of the organization in general.

Keywords : computer security, network security, methodology, computer security management.

0. INTRODUCTION.

Network security should be included in the scope of the information security policy of an organization. Interconnectivity is one of the most widely used computer buzzwords today, however very few organizations attempt to take a reliable and security consistent approach in implementing interconnectivity. Practical experience of the authors manifested the following problems :

- In cases where network security has been addressed it is the authors' experience that it happened at a too low or heavy technical oriented level.
- A number of organizations are not sure what is really meant by the term "Network", furthermore they had problems in addressing the complete scope of network security [6].
- Senior managements' comprehension of network security at the majority of organizations lacks an understanding of the technical details thereof. They also have difficulty in understanding the concept of addressing network security specifically in the information security policy for the organization.

- Organizations experience difficulty in addressing the implementation of measures for both network security and information security. This approach is costing organizations large sums of money due to the fact that synchronized and combined countermeasures have a much more powerful effect on the displacement of risks as opposed to the implementation of individual countermeasures.
- Very few organizations take a methodical approach towards the implementation of information security (including network security) counter measures.

The authors very strongly believe that a methodical approach towards the implementation of information and network security will address the above-mentioned problems. However, following a literature study undertaken by the project team it became evident that very little attention has been paid to this subject. Current literature regarding such an approach could be briefly summarized as follows [2],[3],[4],[6]:

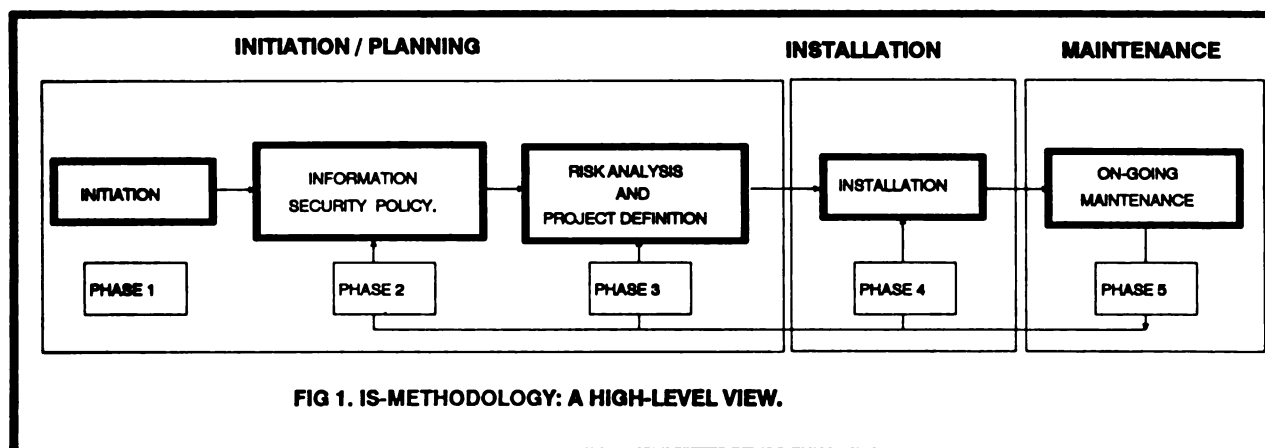
- methodology for the design for a secure network oriented application system
- methodology for the implementation of information security.

The objectives of this paper are to address both the abovementioned issues within the framework of a methodology, the so-called Information Security(IS)-Methodology.

1. INFORMATION SECURITY METHODOLOGY : IS-METHODOLOGY.

The basis of the IS-Methodology was developed at the Rand Afrikaans University by Prof. J.H.P. Eloff and K.P. Badenhorst. The reason for the development is to provide a structured approach for the specification and implementation of information security in an organization.

Fig 1. shows a high-level view of the IS-Methodology. Fig 2. gives a synopsis of the 5 main phases of the methodology. The IS-Methodology is addressing all aspects of information security and forms the framework for the remainder of this paper. Due to length and scope limitations of this paper we will only discuss the concept of technological network security as described in the Installation phase i.e. phase 4 of the IS-Methodology. Further reading matter regarding the other phases of the IS-Methodology can be obtained from [3][4][5].



(1) PHASE 1 - INITIATION : Senior management needs to be made aware of the risks involved when too little or no information security exists in the company. A specially selected steering committee needs to be established to guide the information security plan through its phases. The manager responsible for data communications and networks needs to be represented on this committee.

(2) PHASE 2 - INFORMATION SECURITY POLICY: The establishment of a formal information security policy, which is in line with organizational strategies and company mission, forms an essential basis from which to launch a risk analysis study. This policy contains the definition, framework of terminology, with special reference to network terminology, as well as a matrix depicting responsibility and accountability functions within such a framework. These will also include responsibilities for network security. The information security policy should address the very important issue of network security perimeters, which will be discussed in paragraph 4.

(3) PHASE 3 - RISK ANALYSIS AND PROJECT DEFINITION: Information security risks and associated potential losses need to be quantified and weighed against factors such as productivity, user satisfaction, network response times, cost of controls, and the like. This action will result in cost effective countermeasures and the compilation of a well-defined project plan for the installation of these measures. The compilation of a project plan at this stage of the process will force the integration of countermeasures planned for network security and other projects related to information security such as procedures for access control on mainframes.

(4) PHASE 4- INSTALLATION: The timely installation of information security countermeasures as depicted in the project plan. It is during this phase that an organization has to implement security services and mechanisms. This is especially true in the network environment where implementation of technologies such as gateways, inter-network protocols and communication links will take place.

(5) PHASE 5 - MAINTENANCE: The on-going maintenance includes a regular review of the information security status and requirements as well as the on-going implementation of controls within the development of application systems. This phase also provides for design procedures to implement application systems running in a network environment. These concepts will not be discussed in the context of this paper, the reader is referred to [2] for further reading matter.

FIG 2. OVERVIEW OF THE IS-METHODOLOGY.

Because we continually refer to Fig 3 and Fig 4 throughout this article, it is important that the reader be informed of the basic structure of the diagrams as depicted in the IS-Methodology. The arrangement of tasks and steps, depicted as boxes in the diagrams, form the basic components of the methodology. The connecting lines between them constitute a precedence structure, i.e. certain tasks and/or steps can occur in parallel with others, whereas other tasks/steps require inputs from preceding tasks. A reference structure is used where, for example,

- P4.T4.S1

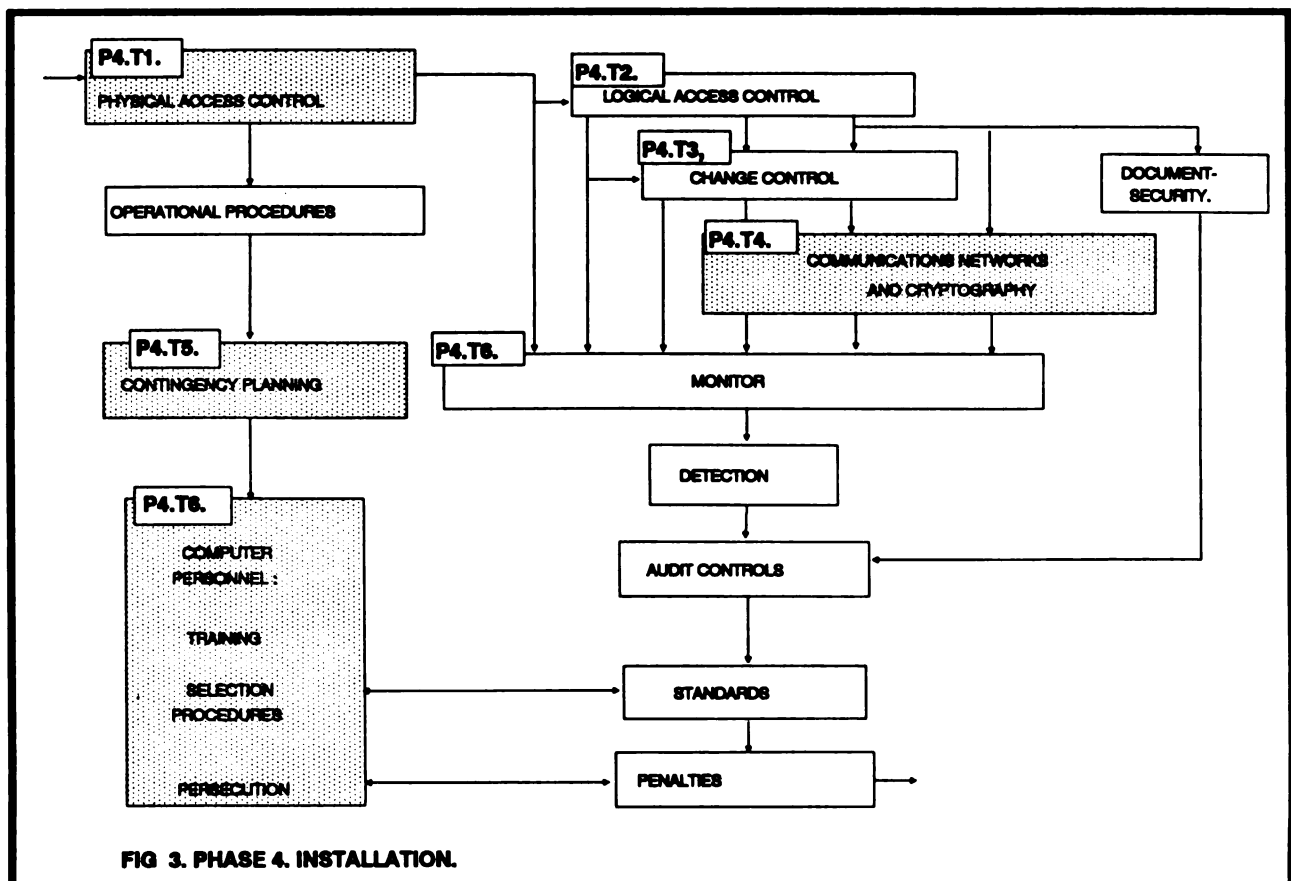
refers to phase 4, task 4, step 1.

2. IS-METHODOLOGY PHASE 4: INSTALLATION.

Fig 3. shows the main tasks in phase 4 of the IS-Methodology. Phase four of the IS-Methodology addresses all the aspects of technological information security such as physical and logical access, cryptography, disaster recovery, planning, and the like.

The installation phase of the IS-Methodology turned out to be one of the most critical because of the variety of specialized skills and line personnel involved. Some of the more important tasks in phase 3 are the following:

- **PHYSICAL ACCESS (P4.T1):** Physical Access to computer terminals and communications equipment is a very important aspect of network security. These terminals and equipment are mostly situated outside the organization itself where strict physical security is difficult. An example is the use of Automatic Teller Machines (ATMs) of banking organizations which are installed in public areas where the equipment cannot be protected by measures like security guards and strongrooms.
- **COMMUNICATION NETWORKS AND CRYPTOGRAPHY. (P4.T4):** This is the part of the IS-Methodology that includes technological network security and that will receive more attention in the rest of this paper.
- **CONTINGENCY PLANNING (P4.T5):** Because of the high level of resource sharing, that is one of the main features in the use of computer and communications networks, the possibility of destruction or corruption of these resources is very high. This leads to the need and increased importance of well-developed contingency planning especially for interconnected networks.
- **COMPUTER PERSONNEL (P4.T6):** An important aspect of network security often overlooked, is the proper training of personnel in the organization. When security includes network security, it will be important that the personnel are also trained in aspects of network communications, as well as the special aspects of network security. Examples here are the use of modems and the need for safe password and key management procedures.



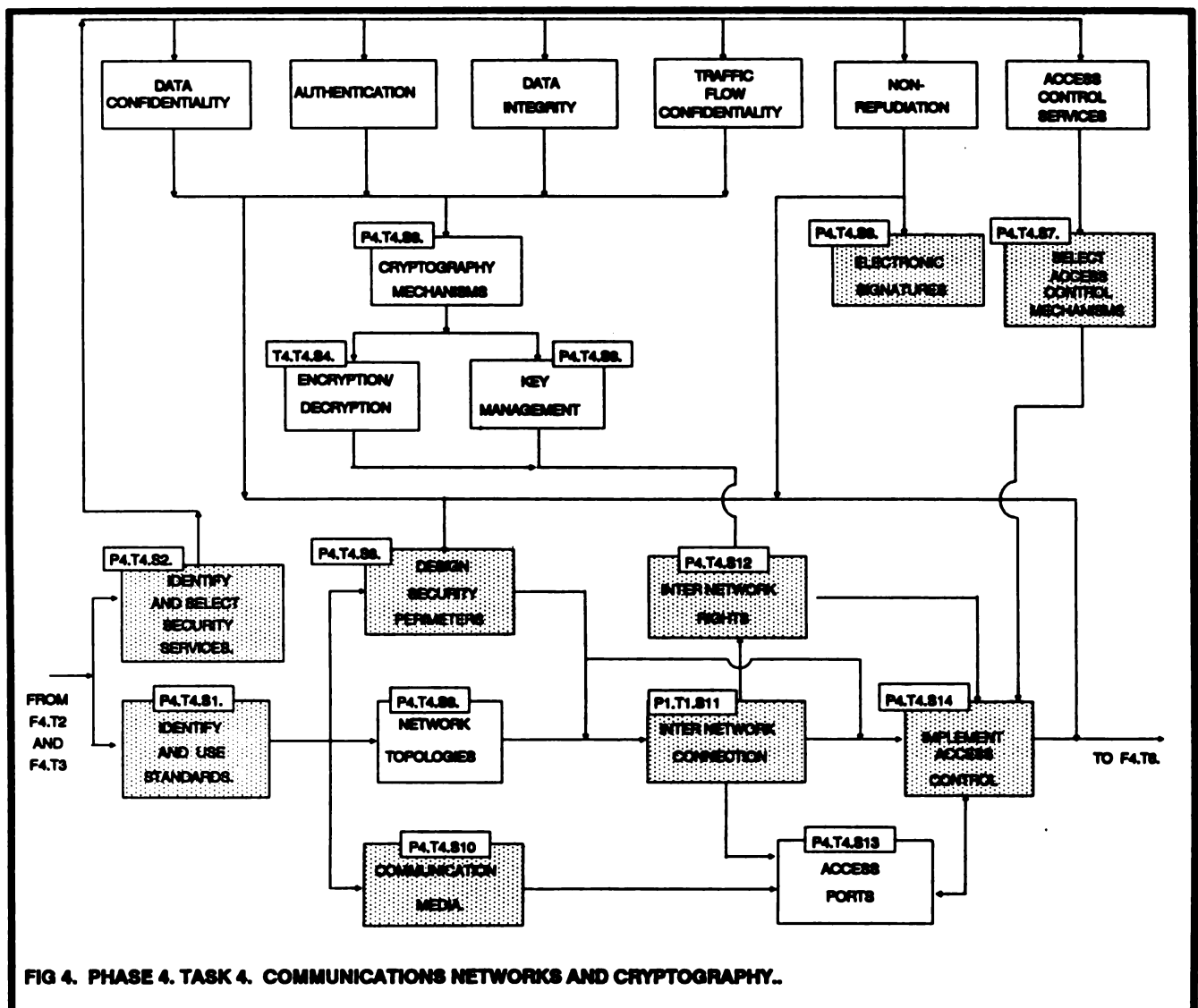
3. ACHIEVING NETWORK SECURITY - IS-METHODOLOGY : PHASE 4, TASK 4.

Fig 4. shows the main steps in Phase 4, Task 4, of the IS-Methodology regarding technological network security. We will not discuss all of the steps mentioned, but only the most important. There are a variety of security measures that are only applicable to computer and communications networks. These measures need special attention when they are planned and implemented.

A few steps (those shaded in the above diagram) from P4.T4. will now be discussed.

P4.T4.S1. Identify and select standards.

The first step will be to identify applicable standards for the implementation of network security in the organization, and to select the standards that will be used. There are quite a few international organizations that develop standards for communications networks. The problem with the majority of standards is that they are developed with interconnection in mind and not with security.



This makes it very important that the Implementors evaluate the standards with security as their main criterion.

To name only a few of the standards :

- **INTERNATIONAL STANDARDS ORGANIZATION(ISO).** The ISO is a voluntary standards body consisting of national standardization bodies in each member country. ANSI is the main US representative in the ISO. The ISO developed the Open Systems Interconnecting model (OSI- model) IS-7498 as well as a security architecture IS 7498-2 . [1][9][10]. Most large organizations in South Africa are moving towards ISO standards.
- **INTERNATIONAL CONSULTATIVE COMMITTEE FOR TELEPHONE AND TELEGRAPH (CCITT).** The CCITT is a member of the International Telecommunications Union, and makes recommendations on telecommunications and data networks.[9] Countries are represented at a national level at CCITT and the US State Department is the voting member for the US. With large private operating companies like regional BELL companies represented at lower levels of membership. In countries like South Africa where the government is in total control of telecommunications matters, it is even more important to adhere to CCITT standards.
- **AMERICAN NATIONAL STANDARDS INSTITUTE(ANSI).** ANSI is a coordinating agency for standards implemented in the U.S.A. on a voluntary basis.[9] One of the most important security related standards that ANSI approved, was the Data Encryption Standard or DES. ANSI is also a member of ISO and tries to adhere to ISO standards as far as possible but may sometimes have to adapt the standards to suit the unique aspects of North American systems.

It may at times be necessary to customize standards for a specific application. The most important factor to consider in the selection process for standards, is the attention given to security in the standard. If the ISO standards are selected, security is already included in the OSI models' security architecture [10]. For the rest of this paper it is assumed that the OSI-model and security architecture were selected were standards.

P4.T4.S2. Identify and select security services.

The next step will be to decide what security services will be needed. The OSI security addendum[10] recommends a number of services but not all of them may be necessary. For the rest of the discussion only the following services are selected :

- **DATA CONFIDENTIALITY,** to protect against unauthorized disclosure of data. The data can be any information or messages that are transmitted over the network.
- **AUTHENTICATION,** used for authenticating the identity of a communicating peer entity and the source of received information. This service will be used to make sure that the two entities that are communicating with each other, are who they claim to be.
- **DATA INTEGRITY.** These services counter the threat of accidental or intentional corruption of data. Accidental corruption may be the loss of a message or part thereof owing to a break in the transmission media or faulty equipment. Intentional corruption may be the intentional duplication, modification or deletion of entire messages or parts of messages by unauthorized intruders.

- **NON-REPUDIATION.** Uses to protect against denial of receipt, or origin of data and messages. This service ensures that a sender of a message cannot deny sending the message or deny the contents of the message. It also ensures that the receiver cannot deny receiving the message or the contents of the message.
- **ACCESS CONTROL.** Used for the protection against unauthorized use of resources accessible through a network.

In this step a decision must also be made on what security mechanisms will be used to implement the selected security services. Once again, a range of mechanisms are available. Not all of these mechanisms need to be selected.[10] Only three are needed to implement the majority of the services, namely cryptography, electronic signatures and access control mechanisms. It is also suggested that physical security forms a component of this group of mechanisms.

Only Electronic signatures will be discussed.

P4.T4.S6. Electronic signatures.

An electronic signature, also known as a digital signature, is a protocol that in effect, has the same use as an ordinary manual signature. This is a mark or sign that only the sender can make, but that the receiver and other users can easily recognize as the sender's electronic signature. Just like a ordinary signature, the electronic signature is used to confirm a message or agree with a document.

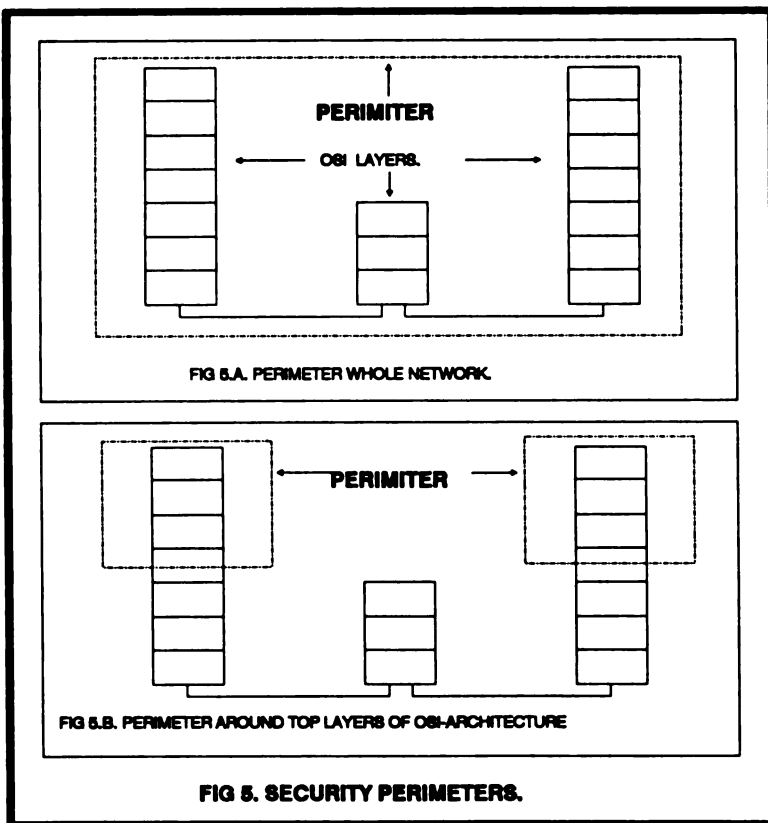
The most important conditions that an electronic signature must meet are the following:

- It must be totally unforgeable.
- It must be authenticatable. This means that it must be possible to prove that the sender made the signature and that he is the owner of the signature.

Electronic signatures are a very interesting field but rather more study is needed before electronic signatures will be accepted as fall-safe.

P4.T4.S8. Design of security perimeters.

A security perimeter is a logical boundary around an area in a network that can be trusted. It is not necessary to implement security services in these areas since security is assured by trusted personnel and equipment.[2] Those parts of the network outside the perimeters must be protected by security services as discussed under P4.T4.S2.



Security perimeters can be used in three ways:

- Perimeter around the whole network as seen in fig 5.1. Here the whole network is trusted and no special services need to be implemented.
- Perimeter around each application. In this case the whole network can not be trusted. Each application must therefore implement its own security services.
- Perimeter around top layers of architecture as seen in fig 5.B. In this case only parts of the network are trusted. The perimeters are drawn around the trusted layers of the network architecture. An example is a network that uses a router to transfer data between two sub-networks. If the routers cannot be trusted the perimeter can only be drawn around the top five layers, that includes the application, presentation, session and transport layers. This means that security mechanisms like encryption must be implemented at least in the transport layer to protect any data moving outside the perimeter.

P4.T4.S10. Communications media.

In this step a decision must be made about the type of transmission media that will be used, or are used already in the network. The most commonly used criterion for selecting transmission media has always been the cost of the media. This approach can lead to a number of problems when one tries to implement security measures. Tables 1 and 2 give an overview of the most important characteristics of some of the most used transmission media. Both these tables attempt to show the importance of security related characteristics.

TABLE 1. RADIATED MEDIA.

	RADIO	MICROWAVE	SATELLITE	C E L L U L A R TELEPHONE
COST	LOW	HIGH	VERY HIGH	HIGH
BANDWIDTH	LOW	AVERAGE	HIGH	AVERAGE
RADIATION	VERY HIGH	LOW	HIGH	LOW
INFLUENCE ON SURROUNDINGS	VERY HIGH	LOW	LOW	HIGH
INTEGRITY	LOW	AVERAGE	GOOD	AVERAGE
THREAT OF TAPPING	VERY HIGH	HIGH	HIGH	VERY HIGH
EASE OF TAPPING	VERY EASY	DIFFICULT	DIFFICULT	VERY EASY
EXTENDABILITY	HIGH	HIGH	H I G H (EXPENSIVE)	LOW (FEW FREQUENCIES)

TABLE 2. CONDUCTED MEDIA.

	TWISTED PAIR (UNPROTECTED)	TWISTED PAIR (PROTECTED)	COAXIAL CABLE	OPTICAL CABLE
COST	VERY LOW	LOW	EXPENSIVE	VERY EXPENSIVE
BANDWIDTH	LOW	LOW	HIGH	VERY HIGH
INSULATION	NONE	NONE - GOOD	GOOD	GOOD
RADIATION	VERY HIGH	AVERAGE	LOW	NONE
INFLUENCE ON SURROUNDINGS	VERY HIGH	HIGH (VARYING DEGREES)	LOW	NONE
INTEGRITY	BAD	BAD - AVERAGE	GOOD	VERY GOOD
CABLE WEIGHT	VERY HIGH	HIGH	HIGH	LOW
DANGER OF TAPPING	VERY HIGH	VERY HIGH	HIGH	VERY LOW
EASE OF TAPPING	VERY EASY	EASY	DIFFICULT	VERY DIFFICULT
MAINTENANCE (FREQUENCY)	FREQUENTLY	FREQUENTLY	L E S S FREQUENTLY	LOW

P4.T4.S11. Inter-network connection and P4.T4.S12. Inter-network rights.

Because of the increase in network usage, it has become necessary to share resources on different networks. This leads to the need for connecting different networks to each other.

This interconnection between planned and existing networks can lead to unexpected effects on network security. Special attention should be given in the planning and design of the connection point between the networks, particularly regarding the use of inter-network access rights, also discussed in the IS-Methodology under F4.T4.S11. Very important work in this area was done on "baggage collection". This system is based on the "Path Context Model"(PCM) developed by Von Solms and Boshoff at the Rand Afrikaans University in South Africa [11]. It should be noted that this is new research and the original intention was not to set or implement any standards regarding network security.

The idea of "baggage collection" works as follows: any subject that wants to get access to an object needs certain software and hardware components to satisfy the request. These components may for example be operating system software, networking software, cryptographic software, storage media or communications links. This means that there are certain allowable "access paths" between the subject and the object.

In terms of the discussion of PCM, a subject is the user who wants to send a message to a receiver, the object, by using the network, and not an ISO-type of object. The route that the message must follow through the network is the access path. There may be a number of different access paths the subject can use. The object is the receiver or any network component that must be manipulated to gain access to the final object.

If the subject's access to any of the objects in the access path can be controlled, he can be forced to use a specific path or even be denied access to the object. By associating with each object one or more selected paths, access to the object can be controlled by allowing only access to the object if the right access path is used.

The access paths associated with each object can be called the object's "Security profile". The software and hardware components making up the access path is called the "Baggage". If a subject tries to access an object, baggage is collected all along the access path. If the request reaches the object, the baggage is validated by the object's security profile, and access can be granted or refused.

The system to date is only implemented on a microcomputer environment under the MS-DOS operating system. The next phase of development will be to utilize PCM in networks and in an ISO-network environment. The authors are of the opinion that the PCM system can be valuable in the field of inter-network rights but it will be necessary to conform to some standard.

P4.T4.S14. Implement access control mechanisms and P4.T4.S7. Select access control mechanisms.

Computer and communications networks are complex systems that consist of a variety of different components like shared directories, programs, data and the like. These components need to be protected and it is essential that access to these components is strictly controlled. Any access control measure depends heavily on the positive identification and verification of the identity of the user trying to access the network. There are various identification methods that range from traditional passwords to more modern and sophisticated methods like retina-pattern recognition and voice recognition.

It can be very difficult to decide which of the different identification methods is the best and the most useful. It is important that the user or organizations who want to make use of these methods, are fully informed about the characteristics, advantages and disadvantages of these methods.

A few of the most important factors to take into consideration when evaluating identification methods, are the following :[8]

- How effective are the methods?
- How acceptable are the methods to the public?
- Is it possible to successfully implement the method?
- Cost of the method.
- Duration of the identification process.

5. CONCLUSION

The development and implementation of security requirements in organizations around the world have become major issues. One of the main problems is that technology as well as the abilities of intruders are developing at an alarming rate. This means that security measures will not give protection for an indefinite time.

This paper only covered the communication and network security specific parts of a methodology for information security design and implementation. The fact that the other aspects of the methodology were not covered here, does not mean that they are in any way unnecessary or less important. The aim of this paper was to show that communications and network security need additional attention if security measures are planned and implemented. For this purpose special attention was paid to the place of network security in the context of a well structured methodology.

BIBLIOGRAPHY.

- [1] Barnstad D.K. Considerations for security in the OSI architecture. IEEE Network Magazine, April 1987. Vol. 2. No 1.
- [2] Graft D, Pabral M. Methodology for Network Security Design. IEEE Communications Magazine. NOV. 1990. Vol. 28. No 11.
- [3] Badenhorst K.P, Eloff J.H.P. Framework of a Methodology for the Life Cycle of Computer Security in an Organisation. Computers & Security. 8(1989). Elsevier Publishers LTD , England.
- [4] Badenhorst K.P, Eloff J.H.P. Managing Computer Security: Methodology and Policy. Information Age. Vol. 12 No 4. OCT 1990. Butterworth-Heinemann Ltd.
- [5] Badenhorst K.P, Eloff J.H.P. Computer Security Methodology.: Risk Analysis and Project Definition. Computers & Security. 9(1990).
- [6] Lobel J. Proactive Network Risk Management. IFIP/SEC'90 Proceedings. Elsevier Publishers LTD , (North Holland) 1990
- [7] Pfleeger C.P. Security in Computing. Prentice-Hall Int Editions 1989.
- [8] Davies D.W. Price W.L. Security for Computer Networks: An Introduction to Data Security in Teleprocessing and EFT. John Wiley & Sons. 1987.
- [9] Black U. Data Networks : Concepts, Theory and Practice. Prentice-Hall. 1989.
- [10] International Standard ISO 7498-2. First Edition 1989-02-15. Information Processing Systems- Open Systems Interconnection-Basic Reference Model. Part 2. Security Architecture.
- [11] Boshoff W.H, Von Solms S.H. A Path Context Model for Addressing Security in Potentially Non-secure Environments. Computers & Security. 8(1989). Elsevier Publishers LTD , England.
- [12] Fitzgerald K. "Quest for intruder-proof computer systems". IEEE Spectrum August 1989.

A Secure European System for Applications in a Multi-vendor Environment (The SESAME Project)

T.A.Parker

ICL Secure Systems
Eskdale Road, Winnersh, Wokingham
Berkshire, England

Reporting on a Joint Bull, ICL and SNI* Project

Abstract

The work of the European SESAME Project is described in outline. SESAME provides distributed access control involving single log-on using a mixture of symmetric and asymmetric cryptographic techniques. Differences from Kerberos and SPX are identified.

1. Background

The large distributed systems of the present day have users who access many different applications residing in different end systems supplied by different vendors. The identity and access rights of these users need to be able to be established, communicated and managed in a way which enables these disparate components to interwork in a secure and standard manner.

For the last three years, work has been underway in Europe under the aegis of the European Computer Manufacturers Association (ECMA) to develop a standard security framework within which these objectives can be achieved, paralleling the more product oriented work being done in the USA on Kerberos [1], [2] and SPX [3]. The ECMA work is documented in [4] and [5], and has been described in various public fora in [6], [7] and [8].

This work soon reached sufficient maturity to make it important that it be validated in real distributed computer systems. To this end, project SESAME was created.

2. The Project

Project SESAME is a development and demonstrator project jointly being undertaken by Bull, ICL and SNI/Siemens, three European computer manufacturers who all consider it vital that strong and workable security across Europe-wide multi-vendor distributed systems can be provided in a standard manner. The

project has to date been partly funded by the European Commission (CEC) under the Research on Advanced Communications in Europe (RACE) programme. It is being conducted in two major stages:

Stage 1: The production of a simple demonstrator which shows the feasibility of the ECMA security framework across the systems of the three companies.

Stage 2: Further development of the security features of the demonstrator, moving towards true self-contained security, and implementing a real-life security policy. The working system of Stage 2 will contain proprietary prototype product components, interworking according to standards being laid down within ECMA and ISO. It will also be able to interwork with OSF DCE Kerberos systems. All of these features will be demonstrated.

Parallel activities are also being undertaken. These take the form of studies, and contributions to further the work of International Standards bodies.

Stage 1 is complete, and a successful demonstration was mounted for the European Commission (CEC) in March 1991. The full security architecture for Stage 2 is complete, and the three companies are at the time of writing working towards a Stage 2 implementation.

* Siemens Nixdorf Informationssysteme AG

3. Scope and Objectives

The technical scope of the SESAME Project covers a wide area including:

- management and distribution of cryptographic keys,
- distributed authentication and access control,
- provision of cryptographic services for non-repudiation,
- the integration of Referenced Data Transfer [9],
- aspects of security management, recovery and audit.

It is only the first two of these topics that are described in this paper.

The SESAME architecture aims at providing single log-on over different operating systems and platforms on both large and small networks. Implementations based on the architecture should be tailorable to a variety of customer policy requirements and investment levels. The security features should be provided in a way which can be transparent to applications which have been developed in ignorance of SESAME. The architecture should be capable of being implemented at high levels of assurance; the levels achievable under established international evaluation criteria should not be limited by the architecture.

There are parallels with Kerberos, but also important differences; however the SESAME development recognises and caters for the need to interwork with OSF DCE Kerberos. The main technical differences are:

- SESAME provides greatly improved user authentication with proper administrative control over re-tries, time-outs and simultaneous log-ons by the same user;
- it supports off-the-shelf standard applications that are unaware of the underlying security regime, as well as applications which wish to use SESAME security data in the exercise of their own access controls;
- it handles access control attributes of all types, including capabilities, group memberships, organisational roles and security labels, in a unified way;
- a user can make late but securely enforceable decisions on how his or her access rights are to be used and refined, without further recourse to a security server. This brings both performance and security benefits;
- it is possible to include the security properties of software components, and their location, in access control decisions. In particular the security properties

of the user's point of access to the distributed system can be included, and a software entity such as an application can be given access rights of its own;

- it does not require encryption of any data except cryptographic keys and similarly sized control values. This removes some constraints on its applicability that might be imposed by some Governments;
- it makes optimum use of both asymmetric and symmetric cryptographic techniques.

4. Technical Description

Sections 4.1 and 4.2 give overviews of the approach taken in the SESAME Stage 2 architecture for key distribution and for authentication and access control. These are followed by a description in 4.3 of the contents of the SESAME security certificates (the Authentication Certificate and the Privilege Attribute Certificate) which are central to the whole approach. Section 4.4 describes the different ways in which the cryptographic capabilities described in 4.1, and other techniques, are used to control the use of these certificates. Section 4.5 describes how the architecture of the target application machine is constructed in a way which helps security evaluation. Section 4.6 describes how the PAC can be qualified in terms of the access rights it carries, and its validity time.

4.1 Key Distribution

The SESAME architecture treats cryptographic key distribution separately from authentication and access control. There are two stages:

1. Two communicating entities obtain a symmetric "Basic Key" with which they will be able to communicate. This can be obtained either by using a Key Distribution Service (KDS), employing conventional symmetric cryptographic techniques, or by using public key technology and Directory Certificates. Apart from the enhancement described for security certificate protection in Section 4.4.3 below, the techniques are well known. It is a positive feature of the scheme that standard key distribution methods can be used.

In the KDS case, if one of the entities is a workstation, it may not be directly known to a KDS. However since any human user who is using the workstation will be known (if not, the user will not be using the system at all), the Authentication Service at which the user is authenticated can be used to obtain keys for the workstation. It may simply provide the workstation with a key for the KDS, leaving it to obtain further keys from the KDS,

or it may provide some keys directly. The protocol for the first of these options is used in 4.4.3.

Similarly, if public key technology is in use, the user's Authentication Service can be used to return the public key of a Certification Authority which can then be used to verify user Certificates fetched from a Directory.

2. The Basic Key may or may not be the actual cryptographic key used to protect all conversations, and there may be a stage in which one or more "Dialogue" keys are derived from the Basic Key. It will be seen in Section 4.5 that there are advantages to be had in doing this. The method of derivation is defined: specified one-way functions, seeded by a time-based unique number¹.

4.2 Authentication and Access Control

In any distributed system, if the point at which a security subject² authenticates itself to the system is not co-located with the point(s) at which it will subsequently be accessing the system's resources, the fact of authentication must be communicated to the accessed targets in a manner that is credible to them. The aim of both Kerberos and SESAME is to provide that credibility through the appropriate use of security certificates and associated cryptographic techniques.

The SESAME architecture uses two types of security certificate as the vehicles of communication: the Authentication Certificate (AUC) and the Privilege Attribute Certificate (PAC)³. Their means of protection is the same (see Sections 4.4.1 to 4.4.6), and the certificates themselves are syntactically very similar. The main difference is that the PAC contains "Privilege Attributes" describing the PAC subject's access privileges, and the AUC merely certifies the fact that its subject has been authenticated.

The main functional components and access steps for a subject to authenticate and obtain a PAC to access a target are illustrated in Figure 1. The preparatory key distribution functions are omitted. The sequence below

may have been preceded by a similar sequence by which the Subject Sponsor itself was authenticated.

The software acting as a human user's or hosted software entity's agent in making the requests for security certificates is known as a Subject Sponsor. A human user's Subject Sponsor will typically reside in the user's workstation. The point of authentication is a server of the Authentication Service (A-Service) from which an AUC for the authenticated subject can be obtained. The point(s) at which a subject's access privileges are

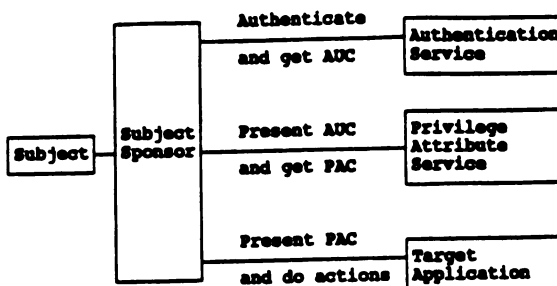


Figure 1. Functional Components

managed and authorised are servers of the Privilege Attribute Service (PA-Service), which supplies PACs on the presentation of a suitable AUC. The process is similar to the MIT Kerberos process of obtaining a TGT followed by a Service Ticket⁴. In the SESAME architecture however a single PAC can be used for multiple targets (if policy permits) and can contain a variety of access control attributes; also the two SESAME security services can be implemented together as a Combined Authentication and Privilege Attribute Service (CAPA-Service) and no AUC is then needed.

Any of the SESAME security services can be implemented distributed over a number of physical servers. The SESAME Stage 1 demonstration had a single CAPA-Service distributed over three servers.

A Subject Sponsor can itself be authenticated and be associated with access control privileges which, if security policy dictates, can be used to temper the contents of the security certificates of subjects sponsored by it.

An important feature of the SESAME implementation is that the Privilege Attributes in the PAC are globally scoped. A PAC would not contain for example a UNIX user-id or group-id for a particular target end-system, instead the values in it would be such things as a global

1 One Dialogue key would be used for integrity purposes, the other for confidentiality if required. To satisfy Government requirements, the confidentiality key could be arranged to be weaker than the integrity key.

2 This general term, abbreviated to 'subject' from here on, is used to signify either a human user or a software system component in an active role.

3 Not to be confused with the OSF DCE PAC

4 Indeed, SESAME permits the Authentication Service to supply a Kerberos TGT which can then be used by the Subject Sponsor to access servers of Kerberos ticket granting services.

access identity, an organisational role (understood from an enterprise view of the system) or a government clearance. It is the responsibility of the target end-system to provide the mapping between these incoming global values and the local access control environment. In this way the management responsibility for access control can be devolved to the parts of the distributed system that are most natural from the enterprise viewpoint: global attributes of subjects being managed in the A- and PA-Services, their impact being managed in the end-systems they are accessing.

4.3 AUC and PAC Contents

There are a number of fields common to both AUCs and PACs. They serve to control and monitor the ways in which they are used. These are outlined below, followed by descriptions of fields unique to the AUC and PAC respectively. For a full ASN.1 specification see [10]. Notice SESAME's use of different types of identity:

Certificate Identifier

for audit and revocation purposes.

Creation/Validity Times

for expiry control (see 4.4.2).

Initiator Qualification

for controlling who may use the security certificate (see 4.4.3 to 4.4.5).

Target Qualification

for controlling the targets for which the security certificate is valid (see 4.4.6).

Check Value

the integrity seal or signature, including information about the authority that signed or sealed the security certificate, and the method used.

Audit Identity

an identity of the subject suitable for audit purposes.

In addition, an AUC may contain:

Authenticated Identity

the authenticated identity of the subject of the AUC.

SS Information

if the AUC is not for a Subject Sponsor, it may contain information about the Subject Sponsor via which the subject of the AUC was

authenticated. This enables the PA-Service to set correctly the privileges in subsequent PACs for this subject.

Authentication Level

an indication of the quality of authentication performed in order to get this AUC.

and a PAC may contain:

Other PACs/ref PACs

for future extension of the proxy concept, and as a means of linking PACs together (the use of these is for further study. They are merely a gleam in SESAME's eye).

Charging Identity

for billing purposes.

PAC Type

indicates whether the PAC is for a subject, a Subject Sponsor or for a subject but having been tempered by the Subject Sponsor via which the subject is accessing the system.

Privilege Attributes

the access privileges that the PAC represents. These may include various identities, clearances, group memberships and so on, as defined in [5].

4.4 Protection of Certificates in Use

All of the techniques described below apply equally well to AUCs and PACs unless explicitly specified, and the general term "security certificate" is used to denote either of them. SESAME supports the following protection features, any of which can be used either individually or in combination:

- stopping undetected tampering
- constraining when and how many times the certificate may be used
- confining the use of a security certificate to be from the point to which it was issued,
- confining the use of proxiable security certificates to identified groups of targets (e.g. only the servers of a particular distributed service),
- linking specific actions with a security certificate
- confining the use of proxiable or non-proxiable security certificates to identified specific targets.

Each of these is now described in turn.

4.4.1 Tamperproofing

An AUC may be sealed by a symmetric key (for performance reasons) or signed. Sealing is appropriate when an A-Service's server shares secret keys with a single, or only a few servers from PA-Services and the specific single server of the PA-Service with which an AUC is to be usable is known.

A PAC is signed by the PA-Service server that issues it using its private key⁵. Use of asymmetric cryptographic technology permits a security certificate to be sent to multiple targets for validation without those targets being able to tamper with it⁶.

4.4.2 Constraining When and How Many Times the Certificate May be Used

Each security certificate contains time expiry information. It can also optionally contain a count field nominating the number of times (e.g. once) that the security certificate may be offered for use. This provides a degree of extra protection for example for long-lived PACs for use in overnight remote job entry situations, where the time of use may not be accurately known, but once it is used it is to be no longer valid. It also enables security certificates to be used for limited access purposes, akin to an admission ticket being collected at the door. Depending on the scope of use of the security certificate, policing the count field may require that it be presented by the target to a validation authority common to all of the targets for which the certificate is valid (see 4.5).

4.4.3 Non-proxiable Security Certificates

To make a security certificate usable only from the point to which it was originally issued (we shall call this the Issue Point) the certificate is linked to the cryptographic keys used for communicating from the Issue Point. This is done by associating an "identity" of the Issue Point with both the certificate and with the keys⁷. Protocols have been defined to support this functionality, either when secret keys or private/public keys are used. A simplified version of the secret key protocol is presented below. It is based on the use of a conventional Key Distribution Service and incorporates an extended

5 The terms 'private' and 'secret' are used here as defined in [11] to differentiate between asymmetric and symmetric technologies.

6 SESAME may later be extended to permit the use of symmetric PAC seals in limited circumstances, for example if a PAC is targeted at one specific target. The PAC would then have similarities with a Kerberos Service Ticket.

7 In ECMA the "identity" idea is generalised to permit the use of other attributes, see Appendix A.1 of [10]

Needham Schroeder protocol [12]. In the example below we assume that the Subject Sponsor is unknown to the system and does not contain any long term secret keys. Replay protection fields and informative, but cryptographically non-relevant fields have been omitted for clarity.

In Figure 2 below, the following notation is used:

"APA-Server" is used in this example to denote a server of a SESAME security service which supplies Security Certificates. It could be an Authentication Service or a Combined Service. In the first case, the certificate would be an AUC and the target a Privilege Attribute Service. In the second case, the certificate would be a PAC and the target an application.

(xxx)K means encrypted under key K

[xxx]K means sealed under key K

Long Term Keys:

KAK is a key shared between the APA-Server and the KDS

GTK is a key shared between the target and the KDS

Keys Established During the Protocol:

KAI is a key shared between the APA-Server and the Issue Point

KIK is a key shared between the Issue Point and the KDS

KIT is a key shared between the Issue Point and the target

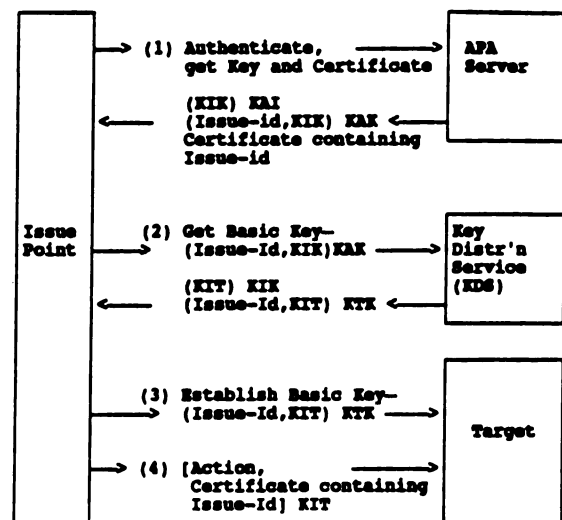


Figure 2. Symmetric Key Protocol for Non-proxiable Certificates

(1) The Subject Sponsor at the Issue Point sponsors the authentication of its subject to the APA-Server and asks for a key with which to communicate with the KDS. It also asks for a security certificate that is not to be proxiable. The authentication method used establishes a temporary secret key KAI between the APA-Server and the Issue Point; in SESAME, if the subject is authenticating using a password, KAI is a one way function of the password seeded by a time-based unique number. Three things are returned:

- a Key KIK encrypted under KAI to be deciphered at the Issue Point and which will be used there to talk to the KDS,
- a Key Package for the KDS encrypted under a master key KAK shared between the APA-Server and the KDS. This package serves to link Issue-Id with KIK. Whenever KIK is used the KDS knows it is to be associated with Issue-Id. Only a trusted and authorised server such as this A-Server can establish a KDS key linked to an identity in this manner. The KDS recognises KAK as a key of such a server,
- a security certificate for the subject, containing (among other things) Issue-Id as a control field. In this case since the Subject Sponsor and therefore the Issue Point is unknown, Issue-Id is an arbitrary value unique to the APA-Server.

(2) The subject now chooses a target to which it will be presenting the certificate. The Subject Sponsor asks the KDS for a Basic Key with which it can communicate with the target, passing the Key Package to the KDS. The KDS replies with:

- the requested Basic Key KIT, encrypted under KIK for use from the Issue Point,
- a Key Package for the target, encrypted under a key KTK shared between the KDS and the target. Because KIK was used in the request, this package links Issue-Id with KIT. Whenever KIT is used the target will know it is to be associated with Issue-Id.

(3) The subject establishes the Basic Key with the target by sending it the Key Package just received. The target now associates KIT with Issue-Id.

(4) Actions by the subject from the Issue Point, and the security certificate to authorise them are sent sealed under the protection of KIT. The target sees that the certificate contains Issue-Id as a control field and checks that it is the same as the Issue-Id associated

with KIT. If so, the certificate was offered from the valid Issue Point.

Note that the target is unable to use the security certificate itself with other targets as it cannot obtain from the KDS a Basic Key linked to Issue-Id for communicating with the other target; it is not a trusted server in this respect. When a server (or workstation) which shares a long term key with a KDS requests a Basic Key, the key will be supplied associated with a known and managed identity value which has been associated with that long term key.

If public key technology is in use for key distribution, the APA_Server can be used to generate a short term private key and create the corresponding public key certificate linked to Issue-Id, for the workstation to use to set up Basic Keys. Space does not permit a full description of this protocol⁸.

4.4.4 Control of Proxy by Initiator/Target Grouping

Real systems increasingly contain distributed application services, each of which is distributed over a number of physical servers. A subject using such a service may not know precisely which server of the service can support its requests. The subject will in such cases simply make a request on a convenient server, and expect his security certificate to be passed on as necessary. The certificate may therefore be required to be used by proxy by one server of the service with another, but nowhere else. Other target groupings can also be envisaged. The necessary controls are provided as follows:

- targets can be grouped into trust groups. Each target knows which trust group(s) it belongs to,
- a security certificate can be created so as to be usable only within one or more nominated trust groups. For each trust group, for this method of protection, such a certificate contains a pair of fields: a Protection Value and the trust group Identifier,
- the Protection Value is a one way function of a secret value (the Control Value for this trust group) initially only known to the Issue Point which requested the certificate,
- whenever the certificate is offered to a target, it is accompanied by the Control Value for the target's trust group encrypted under the Basic Key used to communicate with the target.

⁸ It would be interesting to investigate possible extensions to SPX to do this.

- on receipt of such a certificate, after decrypting the Control Value, a target makes the following checks:
 - . am I in a trust group named in this certificate?
 - . if so, does the one way function applied to the Control Value match the Protection Value for this trust group in the certificate?
 - . if so, the certificate is valid for me for use with actions sealed under that Basic Key.

The target is now in the same position as the original Issue Point with respect to the trust group used: it knows the Control Value, and can use it with the security certificate to perform actions on another target in the same trust group. Targets outside the trust group reject the certificate (motivated by self protection). The proxy scheme is secure provided that the members of the trust group do not reveal the Control Value to maliciously inclined targets outside the group.

Note that although Kerberos and DEC's SPX provide for proxy, they provide no way for the original initiator to control the subsequent propagation of proxy rights.

4.4.5 Linking Specific Actions with a Security Certificate

SESAME also permits the use of a different kind of Protection Value, a public key corresponding to a Control Value which is the private key. In this protection method, the Control Value is not sent to the target; instead the security certificate is valid within the trust group only for actions signed by the Control Value, i.e. issued from the original Issue Point. A similarity with the SPX approach is noticeable, though in SESAME this method, which is computationally significantly more expensive than the method described in 4.4.4, is used only when it is important to prevent the modification of particular actions. Otherwise, the 4.4.4 method is to be preferred⁹. Note that both methods can be used in combination if required.

4.4.6 Confining the use of Security Certificates to Specified Targets

A SESAME security certificate can be arranged to be usable only at targets which possess particular attributes. These attributes are entered in control fields in the certificate. When a certificate containing these controls is offered to a target, the target compares the attributes found, with its own attributes. If there is no match, the target rejects the PAC. Normally it is expected that this

form of control will be used with attributes which are identities, for PACs targeted at individually named target application servers; however generalisation to target groups and types is possible (e.g. this PAC is valid for all ICL electronic mail servers).

4.5 Target Machine Structure

The description so far has treated the PAC handling logic at the target as a single unit, but in reality in SESAME it is structured as shown in figure 3. below¹⁰.

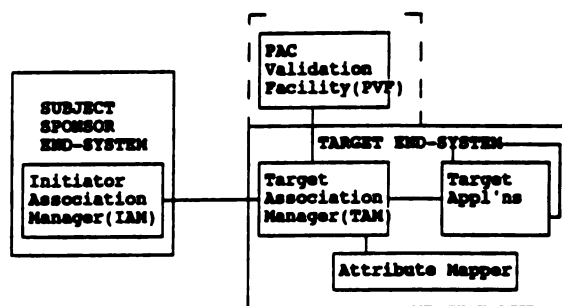


Figure 3. Target Machine Structure

Secure associations between end-systems are handled by an Association Management (AM) subsystem. There are two components of AM involved in an association: Initiator Association management (IAM) and Target association management (TAM). A PAC is transmitted to a target end-system using AM. When it arrives there, the TAM passes the PAC to a separate component, the PAC Validation Facility (PVF), which validates the PAC and establishes the cryptographic context within which TAM and IAM will converse. It also indicates to TAM the Privilege Attributes that apply in this use of the PAC (see 4.6 below). TAM is responsible for organising the necessary mapping between the incoming attribute values and their local equivalents; this may involve establishing an appropriate execution environment for the application.

The PVF may or may not be co-located with the TAM. More than one TAM, and more than one target application may share the use of one PVF, though each application may use only one PVF. Conversely there may be as many PVFs as TAMs or applications. These are configuration options.

The Basic Key described in Section 4.1 is shared not with the TAM or Target Application but with their PVF.

⁹ Though it should be noted that the use of signed operations requires no encryption for confidentiality in workstations, a feature which may please some national authorities.

¹⁰ In fact this structure is also present in Privilege Attribute Services, to handle the receipt of AUCs, but the structural benefits are not so significant since the PA-Service is a trusted security service.

It is with PVFs that Key Distribution is performed. The "Dialogue" key used between IAM and TAM is derived from the Basic Key in a manner which does not reveal the Basic Key to TAM.

Naturally the link between the TAM and its PVF needs to be protected. Commonly this will not be a problem: the PVF will be co-located with TAM. Otherwise cryptographic techniques may be needed depending on the physical security of the connection¹¹, and the same key distribution method as is used for normal Basic Key establishment is used.

Authentication of individual target applications is done by the PVF, either by means of the protocol across the link between the TAM and the PVF, or directly if the PVF is in the same end-system as the TAM, the PVF having itself been authenticated by means of its master key shared with the KDS¹².

In this way control over communication is vested in the PVFs. A TAM only obtains the correct communication key(s) if the PAC is deemed valid by the PVF. By using keys derived from Basic Keys for communication, it is possible to provide a key for encryption for confidentiality purposes which is separate from that used for integrity. The key for integrity can be as strong as necessary to provide the required assurance for access control purposes; the key for confidentiality can be either as weak as prevailing government legislation requires, or as strong as an actual government customer might want!

PVFs have relatively simple functionality and require minimal management. By logically separating the PVF from TAM and its applications, the major functions of PAC validation and control are confined to the PVFs, with consequent evaluation benefits. The ability to share PVFs eases manageability of the distributed system. A KDS needs only to share keys with PVFs, Security Servers and authenticatable Subject Sponsors. Indeed one of the roles of a PVF can be seen to be that of a kind of local KDS for the applications it supports.

4.6 PAC Qualification

Although an individual PAC can be used with more than one target, it may be required that a subject operates with different privileges with different targets. For

11 One can imagine many commercial configurations however in which this link would have a sufficiently low risk of wire taps to not require encryption. This is in contrast to links to users' workstations, whose physical security would be much harder to guarantee.

12 Or via its private key and a Directory Certificate if this technology is being used for key distribution

example a user may be cleared to SECRET and obtain a PAC which contains that clearance, but may wish to use the PAC at a particular target using a lower operating clearance of CONFIDENTIAL. In order to avoid the subject having to obtain different PACs for these uses, the SESAME architecture permits the subject to "qualify" his PAC, when it is offered to the target. The qualifier may subset the privileges in the PAC for this use, and/or may reduce the time period over which it is to be considered valid for this use.

Naturally if security policy dictated, separate PACs could have been obtained.

5. Summary and Conclusion

This paper has given an overview of the work of the SESAME project. It has identified a number of points of difference with other schemes, and the consequent benefits obtained. SESAME is not complete; Stage 2, which is just beginning, will provide the first properly secure implementation using components of product quality, but the work done in Stage 1 has already demonstrated the feasibility of the basic principles underlying the architecture.

Acknowledgements:

The development of this architecture was a joint effort by members of the SESAME architecture team from Bull, ICL and SNI/Siemens, building on work done in ECMA TC32/TG9.

The individual SESAME contributors were: Philippe Caille, Belinda Fairthorne, Per Kaijser, Tom Parker, Denis Pinkas and Michael Steinacker. Particular thanks are due to Per Kaijser and Denis Pinkas for their extensive comments and suggestions on drafts of this paper.

References:

1. J.G.Steiner, C.Neumann & J.I.Schiller, "Kerberos: an Authentication Service for Open Network Systems", USENIX Winter Conference 1988, pp.191-201.
2. J.Kohl, C.Neumann & J.G.Steiner, "Kerberos V5 Protocol Specification", MIT Request for Comments (RFC) 2nd Draft, Nov 1989.
3. "SPX: Global Authentication Using Public Key Certificates", Joseph J. Tardo and Kannan Alagappan, Proc. 1991 IEEE Symposium on Security and Privacy

4. ECMA TR/46 "Security in Open Systems - A Security Framework", July 1988.
5. ECMA-138 "Security in Open Systems - Data Elements and Service Definitions", December 1989.
6. "Security in Open Systems - A Report on the Standards Work of ECMA's TC32/TG9", T.A.Parker, 10th U.S. National Security Conference, 21-24th December 1987.
7. "ECMA and Security in Open Systems, the Second Step", J.Kruys, I4 Workshop, Copenhagen, September 1989.
8. "A Model for Security In Distributed Systems", R.Cole, Computers and Security, 1990 No. 9, Pages 319-330.
9. ISO/IEC CD 10740, Parts 1 and 2 "Referenced Data Transfer" 2nd Jan 1991
10. Authentication and Privilege Attribute Security Application", draft issue 7.1., an ECMA TC32/TG9 working document.
11. ISO/IEC CD 10181-2.2 "Information Technology - Security Framework for Open Systems - Part 2: Authentication Framework."
12. "Using Encryption for Authentication in Large Networks of Computers", R.M. Needham and M.D. Schroeder, CACM Vol.21, No. 12, December 1978.

A SECURE QUORUM PROTOCOL

Masaaki Mizuno* Mitchell L. Neilsen

Department of Computing and Information Sciences
Kansas State University
Manhattan, Kansas 66506

Abstract

In a distributed database system, several replicas (copies) of a data object may be maintained at different sites to improve reliability. However, maintaining replicas may also affect the security of the system in terms of secrecy and integrity. Thus, it is natural to integrate reliability and security issues within a replica control protocol.

In this paper, we present a secure quorum protocol which integrates a quorum protocol to attain consistency of replicated data and a cryptographic technique to attain security of data. We present two efficient methods for generating quorums which are best suited for the secure quorum protocol. Then, we present an algorithm, called the join algorithm, which is very useful for constructing a large set of quorums and show that the join algorithm may be used to improve the overall security of the secure quorum protocol.

1 Introduction

In a distributed database system, several copies (replicas) of a data object may be maintained at different sites to improve fault tolerance (reliability). Maintaining several replicas allows the system to gracefully tolerate node and communication line failures. A replica control protocol is used to ensure that different copies of a data object appear to the user as a single nonreplicated object, i.e., objects are *one-copy equivalent* [1, 3]. One well known protocol is based on weighted voting [6]. Agrawal and El Abbadi generalized weighted voting in terms of read and write quorums [1]. Associated with each data object, (several) read and write quorums are formed, each of which is a subset of copies of the data object. A read operation accesses all of the copies in a read quorum, and a copy with the largest version number is returned. A write operation writes to all of the copies in a write quorum and assigns each copy the version number that is one more than the maximum version number encountered in the write quorum. Let R and W be sets of read and write quorums, respectively. In order to ensure one-copy equivalence, the read and write quorums must satisfy the following two intersection properties:

1. **Write-write** : $G, H \in W \Rightarrow G \cap H \neq \emptyset$.
2. **Read-write** : $G \in R, H \in W \Rightarrow G \cap H \neq \emptyset$.

Maintaining replicas may affect not only the reliability, but also the security of the system. Security is concerned with the following two principal issues [4]:

- secrecy (privacy) - to prevent unauthorized disclosure of data, and
- integrity (authenticity) - to prevent unauthorized modification of data.

*This work was supported in part by the National Science Foundation under Grant CCR-8822378.

Maintaining replicas may improve the integrity of the data object. As long as an intruder has not modified all of the copies and an authorized user can detect which copies have been modified by the intruder, the user may still access a correct copy of the data object. However, maintaining replicas may decrease the secrecy of the data. In order to obtain confidential data, an intruder may access any copy of the data object.

Since reliability and security are closely related in a replicated database system, it is natural to integrate one-copy equivalence and security issues in a replica control protocol. However, relatively few such attempts have been made. Two such protocols have been proposed by (1) Herlihy and Tygar [7] and (2) Agrawal and El Abbadi [2].

This paper presents a secure quorum protocol (SQP) which integrates a quorum protocol to attain one-copy equivalence and a cryptographic technique to attain data security. By appropriately choosing certain parameters, SQP does not increase the number of accesses required to perform a read or write operation. The secure quorum protocol is best suited for a set of quorums which are all the same size, called **symmetric quorums**. We present two methods for generating symmetric quorums.

We have proposed an algorithm, called the **join algorithm**, which takes sets of quorums as input and returns a new set of quorums [8]. The join algorithm is very useful for constructing large sets of quorums. In this paper, we extend the join algorithm to generate quorums which may be used in SQP. Such quorums may be used to improve the overall security.

The organization of the paper is as follows: Section 2 briefly reviews Herlihy and Tygar's protocol and Agrawal and El Abbadi's protocol. We also present an overview of SQP. Cryptographic systems and Shamir's secret sharing algorithm on which SQP is based are reviewed in Section 3. Section 4 presents the secure quorum protocol (SQP). Section 5 describes two methods for generating symmetric quorums. Section 6 presents the join algorithm applied to SQP and a simple security analysis.

2 Review and Overview

In this section, we review Herlihy and Tygar's protocol and Agrawal and El Abbadi's protocol. Then, we present an overview of the secure quorum protocol (SQP). By reviewing these protocols, we informally introduce some important terminology.

2.1 Herlihy and Tygar's protocol

Herlihy and Tygar's protocol uses a quorum protocol to achieve one-copy equivalence and a cryptographic technique to attain security. Each replica is encoded by using a secret key. Shamir's secret sharing algorithm may be used to break the key into n pieces (called shadows), and each shadow is distributed to a different site. In Shamir's algorithm, at least t out of n shadows ($t \leq n$) are needed to recover the key, where t is called the threshold [10]. To read a data object, any t shadows are retrieved to determine the key, and then a read quorum of copies are read and decrypted using the key. The value of a copy with the largest sequence number is the current value of the object. To write a data object, the new value and the new sequence number are encrypted using the key, and then distributed to a write quorum of copies.

Herlihy and Tygar also proposed a protocol which uses two keys: one for encoding the data and another one for decoding the data. In this method, n shadows are created and distributed to n sites for each key. The thresholds, called the *encryption threshold* (t_E) and the *decryption threshold* (t_D), may be defined separately. However, compromising a key may be done by obtaining any combination of a threshold number of shadows. Thus, if $t_E \geq t_D$, compromising the encryption key also discloses the decryption key.

Note that the integrity achieved by the secrecy of the encryption key (or just the secret key in case of a single key system) is only to prevent an intruder from creating false data in the valid data domain. Herlihy and Tygar discuss another type of integrity: preventing an intruder from destroying valid data

by overwriting the data by garbage or an old copy of the data. The system can only guarantee the preservation of this type of integrity against an intruder who can modify less than t_I replicas, where t_I is called the *integrity threshold*. If each quorum, after the attack, contains at least one uncompromised replica with the current value of the data, authorized users can still obtain the correct data. This is achieved by requiring that quorum intersections have cardinality at least t_I .

2.2 Agrawal and El Abbadi's protocol

Agrawal and El Abbadi's protocol integrates weighted voting to attain one-copy equivalence and a secret sharing algorithm to attain security. A secret sharing algorithm, called Rabin's splitting algorithm [9], is used to divide a data object into n pieces and distribute the pieces to n different sites. Like Shamir's algorithm, Rabin's splitting algorithm requires at least t out of n pieces to reconstruct the original data. However, unlike Shamir's algorithm, Rabin's algorithm requires a total of only $(n/t) * |x|$ space to store data object x , where $|x|$ denotes the size of data object x . The secrecy of the data is attained by requiring an intruder to obtain any t copies of the split data. In order to attain one-copy equivalence, overlap between two quorums must contain at least t replicas. Thus, a larger number of copies must be accessed, when compared with regular quorum protocols. For example, if the size of read quorums is t , the size of the write quorum must be n .

Agrawal and El Abbadi proposed a method to reduce the overlap between quorums from t to 1. In this method, at certain points in time, complete information about a data object is held in a log at a site. This may be a security problem.

2.3 A Secure Quorum Protocol (SQP)

Our secure quorum protocol (SQP) integrates a quorum protocol to attain one-copy equivalence and a cryptographic technique to attain data security. Like Herlihy and Tygar's protocol, each replica is encoded by using a secret key, and Shamir's secret sharing algorithm is used to divide the key(s) into shadows. Unlike Herlihy and Tygar's protocol, distribution of the shadows is integrated with the quorum protocol.

The secure quorum protocol may be used with different encryption, decryption, and integrity thresholds. By appropriately choosing the size of quorums and thresholds, SQP does not increase the number of accesses required to perform a read or write operation. This guarantees that the following improved protocol may be implemented without increasing the number of accesses:

1. For better security, the secret key may be erased after each read or write operation has completed; therefore, the key is reconstructed for each operation.
2. Each data object may be encrypted and decrypted using different keys to further improve security.

The real strength of SQP comes from the join algorithm, which is very useful for constructing a large set of quorums which have the required thresholds. Furthermore, the join algorithm improves the overall security of the key.

3 Security

In this section, we briefly review cryptographic systems and Shamir's secret sharing algorithm on which SQP is based.

3.1 Cryptographic system

An encryption transformation E_K is defined by an encryption algorithm, E , and an encryption key, K . Similarly, a decryption transformation $D_{K'}$ is defined by a decryption algorithm, D , and a decryption key, K' . Transformation $D_{K'}$ is an inverse of E_K ; that is, $D_{K'}(E_K(M)) = M$, for any data object M . There are two types of cryptosystems: symmetric (also called “single-key” or “conventional”) and asymmetric (or “two-key”). In symmetric cryptosystems, $K = K'$, and in asymmetric cryptosystems, $K \neq K'$.

3.2 Shamir's secret sharing algorithm

In this section, We review Shamir's algorithm and define some terminology which is used for formally describing SQP.

In SQP, each secret key K is broken into n pieces (shadows), K_1, K_2, \dots, K_n such that:

1. with knowledge of any t shadows, computing K is easy, and
2. with knowledge of fewer than t shadows, computing K is impossible.

One such scheme was proposed by Shamir [10]. The scheme is based on Lagrange interpolating polynomials. The shadows are derived from a random polynomial h (with integer coefficients) of degree $t - 1$, where $h(0) = K$. The shadows are generated by evaluating $h(x)$ at n distinct non-zero integer values x_1, \dots, x_n . Thus, $K_i = h(x_i)$ for $1 \leq i \leq n$. We assume that each shadow K_i is stored as a pair, $K_i = (x_i, h(x_i))$.

We define an **encryption shadow assignment** to be a function $s_E : U \rightarrow \mathbf{N}$, where U is a set of replicas and \mathbf{N} is the set of all non-zero integers. For instance, $K_i = (s_E(i), h(s_E(i)))$ is the encryption shadow assigned to replica i . Similarly, a **decryption shadow assignment** is a function $s_D : U \rightarrow \mathbf{N}$. In a single-key system, $s_E = s_D$.

The **encryption threshold** t_E is the number of different shadows needed to reconstruct K_E . Similarly, the **decryption threshold** t_D is the number of different shadows needed to reconstruct K_D .

4 Secure Quorum Protocol

In this section, we present a secure quorum protocol (SQP). For simplicity, we assume that a single replica is stored at each site. Several variations of SQP may be possible based on

1. whether a secret key is associated with the whole database, a certain set of data objects, or each data object; and
2. whether each secret key is reconstructed for each operation, or is kept in volatile storage for a certain length of time.

Here, we present the most secure, but least efficient protocol, i.e., a separate key is associated with each data object, and a secret key is reconstructed for each operation.

Three keys are associated with each data object: a pair of asymmetric keys, called an **encryption key** K_E and a **decryption key** K_D , and a conventional key, called a **writer key** K_{WW} . The data object D is encrypted using K_E (the encrypted data is denoted by $E_{K_E}(D)$), i.e., $E_{K_E}(D)$ can only be decrypted by using K_D . Two encrypted copies of the version number V are associated with each data object. One copy is encrypted using K_E (denoted by $E_{K_E}(V)$) and the other copy is encrypted using K_{WW} (denoted by $E_{K_{WW}}(V)$), i.e., $E_{K_E}(V)$ can only be decrypted by using K_D , and $E_{K_{WW}}(V)$ can only be decrypted by using K_{WW} . Copy $E_{K_E}(V)$ is used for passing the version number from a writer to a reader. Copy $E_{K_{WW}}(V)$ is used for passing the version number from a writer to another writer. Thus, we assume that associated with each data object, the system maintains areas to store the two encrypted version numbers and the three shadows of the keys.

The shadows of the keys are distributed among the replicas so that

1. if a site can read shadows from the replicas in a read quorum, it can reconstruct K_D , and
2. if a site can read shadows from the replicas in a write quorum, it can reconstruct K_E and K_{WW} .

Construction of such quorums is described in Section 5.

The secure quorum protocol, which is executed at each site, is described as follows:

1. Data object initialization:

The site creating a data object randomly chooses three keys K_E , K_D , and K_{WW} . The site uses Shamir's secret sharing algorithm to divide K_E and K_{WW} into shadows such that any t_E shadows may be used to reconstruct K_E and K_{WW} . The shadow assignment for K_{WW} is the same as the shadow assignment for K_E . Similarly, K_D is divided into shadows such that any t_D shadows may be used to reconstruct K_D . The data object is encrypted using K_E . The version number is encrypted using both K_E and K_{WW} . The encrypted data and version numbers are distributed to each site, along with the shadows assigned to the site.

2. Operation execution:

- **Read operation:** In the first step, the site reads the encrypted replica, the encrypted version number (for readers), and the shadow of K_D from each of the sites in a read quorum. Then, the site reconstructs K_D from the shadows. The site decrypts all of the version numbers using K_D and determines which replica has the largest version number. Then, the site decrypts this replica using K_D and returns it. Finally, the site discards K_D .
- **Write operation:** In the first step, the site reads both of the encrypted version numbers and the shadows of K_E and K_{WW} from each of the sites in a write quorum. Then, the site reconstructs K_E and K_{WW} from the shadows, and determines the maximum version number by using K_{WW} . The copy to be written is assigned a new version number that is one more than the maximum version number. The site encrypts the new version number using both K_E and K_{WW} and the data using K_E . Then, the site writes the encrypted data and both of the encrypted version numbers to all of the sites in a write quorum. Finally, the site discards K_E and K_{WW} .

5 Secure Quorum Generation

First, we formally define the integrity threshold t_I as follows: Let W_1 and W_2 be write quorums and R_1 be a read quorum. If $|W_1 \cap W_2| \geq t_I$ and $|W_1 \cap R_1| \geq t_I$, then the quorums have integrity threshold t_I . If an intruder destroys fewer than t_I copies, then each quorum will still contain at least one uncompromised copy.

Let t_E and t_D denote the encryption and decryption thresholds, respectively. Assuming an integrity threshold of t_I , in order to obtain at least t_D and t_E different shadows, read and write quorums must contain at least $t_E + t_I - 1$ and $t_D + t_I - 1$ different shadows, respectively. Such sets of read and write quorums are said to have decryption threshold t_D and encryption threshold t_E , respectively. Read and write quorums which have a predefined integrity threshold t_I , encryption threshold t_E , and decryption threshold t_D are called **secure quorums**.

The highest level of security is obtained if the sizes of all secure write and read quorums are equal to $t_E + t_I - 1$ and $t_D + t_I - 1$, respectively, and each replica contains a different shadow. This is why symmetric sets of quorums are well suited for SQP.

Note that if $t_D \leq t_E$, shadow assignments may be defined such that any write quorum will contain at least t_D different read shadows. Then, a separate writer key K_{WW} is not necessary because a writer may obtain K_D from any write quorum and decrypt the version number using K_D .

In this section, we present two methods for constructing symmetric secure quorums. These methods may be easily modified to be used with Herlihy and Tygar's protocol and Agrawal and El Abbadi's protocol.

5.1 Weighted voting

One well-known method for generating read and write quorums is to use weighted voting [1, 5, 6]. In this section, we show how weighted voting may be modified to generate sets of read and write quorums with given thresholds. Suppose that each replica is assigned a single vote. Let $U = \{0, 1, 2, \dots, N-1\}$ be a set of N replicas. Each replica is assigned a different shadow. For example, we could let $s_E(i) = s_D(i) = i + 1$.

Given a write threshold $q_W \geq \max(\lceil (N+t_I)/2 \rceil, t_E+t_I-1)$, the corresponding set of write quorums is given by $W = \{G \mid G \subseteq U, |G| = q_W\}$. Given a read threshold $q_R \geq \max((N+t_I) - q_W, t_D+t_I-1)$, the corresponding set of read quorums is given by $R = \{G \mid G \subseteq U, |G| = q_R\}$. For example, let $N = 13$, $t_D = t_E = 4$, and $U = \{0, 1, \dots, 12\}$. Possible read and write thresholds, for different values of t_I , are given in Table 1.

Table 1. Read and Write Thresholds								
t_I	q_W	q_R	t_I	q_W	q_R	t_I	q_W	q_R
1	7	7	2	8	7	3	8	8
	8	6		9	6		9	7
	9	5		10	5		10	6
	10	4		11	5		11	6
	11	4		12	5		12	6

5.2 Cyclic read and write quorums

We have developed a new method for generating symmetric sets of read and write quorums using modular arithmetic.

Let $U = \{0, 1, \dots, N-1\}$ denote a set of N replicas. Each replica is assigned a different shadow. Suppose the read quorums are to have size k , where $\max(t_D + t_I - 1, t_I) \leq k \leq N$. Let $G_R = \{a_1, a_2, \dots, a_k\}$, where $a_i = i - 1$. The set G_R is called a **read generator**. The corresponding set of read quorums is given by

$$R = \{\{x_1^j, x_2^j, \dots, x_k^j\} \mid x_i^j = (a_i + j) \bmod N, 1 \leq i \leq k, 0 \leq j < N\}$$

Let $s = k - t_I$ and let $G_W = G_R \cup (\cup_{i=1}^m (\cup_{j=0}^{t_I-1} \{(ik + s + j) \bmod N\}))$, where $m = \lceil (N + t_I - 2k) / k \rceil$. Suppose $G_W = \{a_1, a_2, \dots, a_M\}$. The set G_W is called a **write generator**. If $t_E + t_I - 1 > M$, then we can add arbitrary elements to G_W so that $M = t_E + t_I - 1$. The corresponding set of write quorums is given by

$$W = \{\{x_1^j, x_2^j, \dots, x_M^j\} \mid x_i^j = (a_i + j) \bmod N, 1 \leq i \leq M, 0 \leq j < N\}$$

Since $|G_R| = k$, we obtain $|G_W| = \max(k + mt_I - [(m+1)k - N]^+, t_E + t_I - 1)$, where $x^+ = x$ if $x > 0$ and 0 otherwise.

For example, let $N = 13$, $t_D = t_E = 4$, and $U = \{0, 1, \dots, 12\}$. Generators for different values of k and t_I are given in Table 2.

Table 2. Generators			
k	t_I	G_R	G_W
4	1	$\{0, 1, 2, 3\}$	$\{0, 1, 2, 3, 7, 11\}$
5	1	$\{0, 1, 2, 3, 4\}$	$\{0, 1, 2, 3, 4, 9\}$
5	2	$\{0, 1, 2, 3, 4\}$	$\{0, 1, 2, 3, 4, 8, 9\}$

For example, let $k = 5$ and $t_I = 2$. Then, $G_R = \{0, 1, 2, 3, 4\}$ and $G_W = \{0, 1, 2, 3, 4, 8, 9\}$. The corresponding set of read quorums is given by

$$R = \{ \{0,1,2,3,4\}, \{1,2,3,4,5\}, \{2,3,4,5,6\}, \{3,4,5,6,7\}, \{4,5,6,7,8\}, \{5,6,7,8,9\}, \{6,7,8,9,10\}, \\ \{7,8,9,10,11\}, \{8,9,10,11,12\}, \{9,10,11,12,0\}, \{10,11,12,0,1\}, \{11,12,0,1,2\}, \{12,0,1,2,3\} \}$$

The corresponding set of write quorums is given by

$$W = \{ \{0,1,2,3,4,8,9\}, \{1,2,3,4,5,9,10\}, \{2,3,4,5,6,10,11\}, \{3,4,5,6,7,11,12\}, \{4,5,6,7,8,12,0\}, \\ \{5,6,7,8,9,0,1\}, \{6,7,8,9,10,1,2\}, \{7,8,9,10,11,2,3\}, \{8,9,10,11,12,3,4\}, \{9,10,11,12,0,4,5\}, \\ \{10,11,12,0,1,5,6\}, \{11,12,0,1,2,6,7\}, \{12,0,1,2,3,7,8\} \}$$

6 Join Algorithm

The join algorithm provides a simple and inexpensive way of combining nonempty sets of read and write quorums to form new, larger sets of read and write quorums [8]. In this section, we first review the join algorithm. Then, we extend the join algorithm to generate secure quorums. The extended join algorithm preserves all three thresholds: t_E , t_D , and t_I . Finally, we show that application of the join algorithm to SQP may improve the overall security of the keys.

6.1 Algorithm

Let U be a nonempty set of replicas and let $x \in U$. Let V be a nonempty set of replicas such that $U \cap V = \emptyset$. Let C_U denote the collection of all nonempty sets of read or write quorums under U . Define a function, $T_x : C_U \times C_V \rightarrow C_{(U - \{x\}) \cup V}$, by

$$T_x(C_1, C_2) = \{G_3 \mid G_1 \in C_1, G_2 \in C_2, G_3 = \begin{cases} (G_1 - \{x\}) \cup G_2 & \text{if } x \in G_1 \\ G_1 & \text{otherwise} \end{cases} \}$$

The join algorithm is to apply the above functions to generate sets of read and write quorums. By using the join algorithm, a set of write quorums and the corresponding set of read quorums may be obtained efficiently, even for large N .

We extend the join algorithm to generate secure quorums. Let $C_3 = T_x(C_1, C_2)$. The shadow assignments of C_3 are defined in the following manner: Let s_1 denote a decryption or encryption shadow assignment for C_1 . Then, define a function, $s_3 : (U - \{x\}) \cup V \rightarrow \mathbb{N}$, by

$$s_3(y) = \begin{cases} s_1(y) & \text{if } y \in U - \{x\} \\ s_1(x) & \text{if } y \in V \end{cases}$$

Then, s_3 denotes a decryption or encryption shadow assignment for C_3 . The following theorem proves that the join algorithm, along with the above shadow assignments, generates secure quorums that preserve the thresholds.

Theorem 1: Let U be a nonempty set of replicas and let $x \in U$. Let V be a nonempty set of replicas such that $U \cap V = \emptyset$. Let W_1 be a nonempty set of secure write quorums under U and let W_2 be a nonempty set of secure write quorums under V . Let R_1 and R_2 denote the corresponding sets of secure read quorums. Then, $W_3 = T_x(W_1, W_2)$ is a set of write quorums under $(U - \{x\}) \cup V$ and $R_3 = T_x(R_1, R_2)$ is a set of read quorums under $(U - \{x\}) \cup V$. If W_1 and R_1 have integrity threshold t_I , then W_3 and R_3 also have integrity threshold t_I . Let t_E be the encryption threshold of W_1 and t_D be the decryption threshold of R_1 . Let s_3 be defined as above. Then, the encryption threshold of W_3 and the decryption threshold of R_3 are t_E and t_D , respectively.

Proof: First, we will show that W_3 and R_3 have integrity threshold t_I . Since W_1 and R_1 have integrity threshold t_I , $|G_1 \cap H_1| \geq t_I$ for all $G_1 \in R_1 \cup W_1$ and all $H_1 \in W_1$. Let $G_3 \in R_3 \cup W_3$ and $H_3 \in W_3$. There are four cases to consider:

1. Suppose $G_3 = G_1$ for some $G_1 \in R_1 \cup W_1$ and $H_3 = H_1$ for some $H_1 \in W_1$. Then, $|G_3 \cap H_3| \geq t_I$ because W_1 and R_1 have integrity threshold t_I .
2. Suppose $G_3 = G_1$ for some $G_1 \in R_1 \cup W_1$ and $H_3 = (H_1 - \{x\}) \cup H_2$ for some $H_1 \in W_1$ and some $H_2 \in W_2$. Then, $|G_1 \cap (H_1 - \{x\})| \geq t_I$ because $x \notin G_1$. Thus, $|G_3 \cap H_3| \geq t_I$.
3. Suppose $G_3 = (G_1 - \{x\}) \cup G_2$ for some $G_1 \in R_1$ and some $G_2 \in R_2$ or for some $G_1 \in W_1$ and some $G_2 \in W_2$ and $H_3 = H_1$ for some $H_1 \in W_1$. This case is essentially the same as the above case.
4. Suppose $G_3 = (G_1 - \{x\}) \cup G_2$ for some $G_1 \in R_1$ and some $G_2 \in R_2$ or for some $G_1 \in W_1$ and $G_2 \in W_2$, and $H_3 = (H_1 - \{x\}) \cup H_2$ for some $H_1 \in W_1$ and some $H_2 \in W_2$. Then, $|(G_1 - \{x\}) \cap (H_1 - \{x\})| \geq (t_I - 1)$. Also, $|G_2 \cap H_2| \geq 1$. Therefore, $|G_3 \cap H_3| \geq t_I$.

Therefore, W_3 and R_3 have integrity threshold t_I .

Next, we will show that W_3 has encryption threshold t_E . Let $G_3 \in W_3$. There are two cases to consider:

1. Suppose that $G_3 = G_1$ for some $G_1 \in W_1$. Then, $|s_E(G_3)| \geq t_E + t_I - 1$ because W_1 has encryption threshold t_E .
2. Suppose that $G_3 = (G_1 - \{x\}) \cup G_2$ for some $G_1 \in W_1$ and some $G_2 \in W_2$. Then, $s_E(y) = s_E(x)$ for all $y \in G_2$ and $G_2 \neq \emptyset$. Thus, $s_E(G_3) = s_E(G_1 - \{x\}) \cup s_E(G_2) = s_E(G_1)$. Therefore, $|s_E(G_3)| = |s_E(G_1)| \geq t_E + t_I - 1$.

A similar argument shows that R_3 has decryption threshold t_D . \square

6.2 Example

Consider the following example, where A , B , C , and D are sets of write, as well as read, quorums.

$$\begin{aligned} A &= \{ \{1,2\}, \{2,3\}, \{3,1\} \} & B &= \{ \{4,5\}, \{5,6\}, \{6,4\} \} \\ C &= \{ \{7,8\}, \{8,9\}, \{9,7\} \} & D &= \{ \{a,b\}, \{b,c\}, \{c,a\} \} \end{aligned}$$

Suppose that the initial sets of both write and read quorums are D and that $t_D = t_E = 2$ and $t_I = 1$. Since three different sites appear in D , $n = 3$. Assume that the encryption shadow assignment for D is defined by $s_E(a) = 1$, $s_E(b) = 2$, and $s_E(c) = 3$. Further assume that the decryption shadow assignment for D is the same; that is, $s_D = s_E$. Note that any quorum in D will contain exactly two different shadows.

We may construct a new set of quorums by combining two of the above sets of quorums as follows:

- Let $E = T_a(D, A)$. Then E is given by:

$$E = \{ \{1,2,b\}, \{2,3,b\}, \{3,1,b\}, \{b,c\}, \{c,1,2\}, \{c,2,3\}, \{c,3,1\} \}$$

In this case, since node a is assigned shadow $(1, h(1))$, all nodes appearing in set A are also assigned shadow $(1, h(1))$.

- Let $F = T_b(E, B)$. Then F is given by:

$$F = \{ \{1,2,4,5\}, \{1,2,5,6\}, \{1,2,6,4\}, \{2,3,4,5\}, \{2,3,5,6\}, \{2,3,6,4\}, \{3,1,4,5\}, \\ \{3,1,5,6\}, \{3,1,6,4\}, \{4,5,c\}, \{5,6,c\}, \{6,4,c\}, \{c,1,2\}, \{c,2,3\}, \{c,3,1\} \}$$

In this case, since node b is assigned shadow $(2, h(2))$, all nodes appearing in set B are also assigned shadow $(2, h(2))$.

- Let $G = T_c(F, C)$. Then G is given by:

$$G = \{ \{1,2,4,5\}, \{1,2,5,6\}, \{1,2,6,4\}, \{2,3,4,5\}, \{2,3,5,6\}, \{2,3,6,4\}, \{3,1,4,5\}, \\ \{3,1,5,6\}, \{3,1,6,4\}, \{4,5,7,8\}, \{4,5,8,9\}, \{4,5,9,7\}, \{5,6,7,8\}, \{5,6,8,9\}, \\ \{5,6,9,7\}, \{6,4,7,8\}, \{6,4,8,9\}, \{6,4,9,7\}, \{7,8,1,2\}, \{8,9,1,2\}, \{9,7,1,2\}, \\ \{7,8,2,3\}, \{8,9,2,3\}, \{9,7,2,3\}, \{7,8,3,1\}, \{8,9,3,1\}, \{9,7,3,1\} \}$$

In this case, since node c is assigned shadow $(3, h(3))$, all nodes appearing in set C are also assigned shadow $(3, h(3))$.

By Theorem 1, the resulting quorums in G all contain at least two different shadows, and the integrity threshold $t_I = 1$ is maintained.

6.3 Analysis

In this section, we will give a brief analysis to prove that SQP applied with the join algorithm (called SQPJ) yields a higher level of security than other protocols in which each replica is assigned a different shadow, such as SQP or Herlihy and Tygar's protocol.

For example, suppose $t_D = t_E = 2$ and the total number of replicas $N = 9$. In the other protocols, there are 9 distinct shadows, each of which is assigned to a different replica. If any two replicas are compromised, the key is compromised. However, in SQPJ using the example in Section 6.2, even if two replicas are compromised, the key may not be compromised. Thus, SQPJ is more secure than the other protocols.

Table 4 compares the number of ways in which the key may be compromised if m replicas are compromised.

Table 4. Example		
m	Other ($C_1(m)$)	SQPJ ($C_2(m)$)
1	0	0
2	36	27
3	84	81
4	126	126
5	126	126
6	84	84
7	36	36
8	9	9
9	1	1

Let c denote the probability that a single replica is compromised. Then, the probability that the key is compromised by the other protocols is given by:

$$P_1(c) = \sum_{m=1}^9 (C_1(m)(c)^m(1-c)^{9-m})$$

Similarly, the probability that the key is compromised by SQPJ is given by:

$$P_2(c) = \sum_{m=1}^9 (C_2(m)(c)^m(1-c)^{9-m})$$

Some values for $P_1(c)$ and $P_2(c)$ are shown below in Table 5.

Table 5. Example		
c	Other ($P_1(c)$)	SQPJ ($P_2(c)$)
0.02	0.0131149	0.0099684
0.04	0.0477658	0.0367946
0.06	0.0978380	0.0763803
0.08	0.1583211	0.1252577
0.10	0.2251590	0.1805180

In all cases, SQPJ provides a higher level of security.

7 Conclusion

In this paper, we presented a secure quorum protocol (SQP) and two methods for generating symmetric quorums which may be used by SQP. The first method uses weighted voting and the second method uses modular arithmetic. Then, we presented an extension of the join algorithm for combining existing quorums and shadows. Application of the join algorithm to SQP may improve the overall security.

References

- [1] D. Agrawal and A. El Abbadi. Exploiting logical structures in replicated databases. *Information Processing Letters*, 33:255–260, 1990.
- [2] D. Agrawal and A. El Abbadi. Integrating security with fault-tolerant distributed databases. *The Computer Journal*, 33(1):71–78, 1990.
- [3] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley Publishing Co., 1987.
- [4] D. E. Denning. *Cryptography and Data Security*. Addison-Wesley Publishing Co., 1982.
- [5] H. Garcia-Molina and D. Barbara. How to assign votes in a distributed system. *Journal of the ACM*, 32(4):841–860, 1985.
- [6] D. K. Gifford. Weighted voting for replicated data. In *Proc. 7th ACM Symposium on Operating Systems Principles*, pages 150–162, 1979.
- [7] M. Herlihy and J. D. Tygar. How to make replicated data secure. *Lecture Notes in Computer Science*, 293:379–391, 1987.
- [8] M.L. Neilsen and M. Mizuno. Coterie join algorithm. *IEEE Transactions on Parallel and Distributed Systems*, to appear.
- [9] M. O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36(2):335–348, 1989.
- [10] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–614, 1979.

SECURITY GUIDANCE FOR VAX/VMS SYSTEMS

Debra L. Banning
Sparta, Inc.
3440 Carson Street
Suite 300
Torrance, CA 90503

The VAX/VMS environment provides unique built-in security control features for implementation by a system administrator. However, if the necessary controls are not in place, any or all the VAX/VMS security mechanisms can be easily bypassed. The DEC "Guide to VAX/VMS System Security" manual provides security-related information to increase security on VMS systems, but this manual is over 250 pages long and is difficult for the novice system administrator to follow. Therefore, to assist both novices and experienced system administrators in providing at least the minimal security for their VMS systems, SPARTA developed the "VMS System Security Guideline." This paper summarizes the contents of the guideline that is currently being used throughout the Department of Energy (DOE).

INTRODUCTION

This paper describes the "VMS System Security Guideline" prepared under contract to Lawrence Livermore National Laboratory and sponsored by the DOE Office of Safeguards and Security (OSS) classified computer security program. The purpose of this guideline is to provide guidance to VMS system administrators in establishing and maintaining a secure environment on VMS-based systems. To keep the guideline from duplicating the DEC manuals, the reader is assumed to understand the basic VMS concepts.

The purpose of this paper is to describe the important aspects of the guideline, so that the reader can determine the utility of the guideline in his/her environment. The guideline is divided into three main sections:

1. System Administrator Checklist
2. Primary Security Preparation
3. Additional Security Considerations

The System Administrator Checklist is to be used by experienced system administrators as a method for checking the security implemented on a system. Sections 2 and 3 together cover most of the security topics applicable to most environments. Initially intended for use within DOE, the guideline includes information on meeting the minimum security requirements defined in DOE's computer security regulation [7]. *Command sequences to implement security features described within this paper are provided in the guideline such that the system administrator does not need to consult additional documentation to provide basic security.* In addition, since the body of the guideline addresses security mechanisms implemented in VMS version 4.7 and before, additional appendices describe modifications to security mechanisms for versions 5.0 and 5.2.

SYSTEM ADMINISTRATOR CHECKLIST

The System Administrator Checklist should be used to periodically verify that the necessary security features have been implemented. The checklist is 7 pages long and summarizes the topics covered in the guideline. In most cases a YES/NO format is used with section references into the body of the guideline for locating additional

information. For example, the following questions, extracted from the guideline's checklist, summarize the advice given for setting up VMS Accounts and providing security for users.

Questions asked concerning setting up VMS Accounts are:

1. Have passwords delivered with the standard VMS accounts SYSTEM, FIELD, SYSTEST, SYSTEST_CLIG and DECNET been changed? (Sec. 4.1) YES___ NO___
2. Have the FIELD, SYSTEST, and SYSTEST_CLIG accounts been disabled? (Sec. 4.1) YES___ NO___
3. If your system is a MicroVax, have the accounts USER and USERP been checked for the use of simple passwords? (Sec. 4.1) YES___ NO___
4. Have the accounts ALLIN1, MRGATE, and MRMANAGER been checked for the use of simple passwords? (Sec. 4.1) YES___ NO___
5. Is the value of MAXSYSGRP less than or equal to 10 (octal)? (Sec. 4.2.1) YES___ NO___

Questions asked concerning providing adequate user security are:

6. Are the following restrictions used in the DEFAULT UAF: (Sec. 4.2.2)
 - a. Is PWDMINIMUM greater than or equal to 8? YES___ NO___
 - b. Is PWDLIFETIME less than or equal to 180 days? YES___ NO___
 - c. Are default and authorized privileges only TMPMBX and NETMBX? YES___ NO___
7. If a CAPTIVE account is used, does it have the following restrictions: (Sec. 4.2.3)
 - a. Have the flags CAPTIVE and DISCTRLY been set in the captive account? YES___ NO___
 - b. Has the login command file, LGICMD, been defined in the captive account? YES___ NO___
 - c. Does process limit equal zero (i.e., PRCLM = 0)? YES___ NO___
 - d. Is the group UIC for the captive account unique? YES___ NO___
 - e. Have the LOCKPWD, DEFCLI, DISWELCOME, DISMAIL AND DISNEWMAIL been set? YES___ NO___
8. If a captive account is used, does its login command procedure have the following restrictions: (Sec. 4.2.3)
 - a. Is the READ/PROMPT command used instead of the INQUIRE command? YES___ NO___
 - b. Is the captive command procedure restricted from using the TECO editor? YES___ NO___
 - c. Is F\$LOCATE used to search for input symbols? YES___ NO___
 - d. Has the use of LOGOUT been verified? YES___ NO___
 - e. Does the command procedure handle all error conditions? YES___ NO___
 - f. Does the command procedure file and its directory have only execute access? YES___ NO___
 - g. Is the command "STOP PROC/ID=0" used upon exiting the captive account? YES___ NO___

A NO answer to a question in the checklist, does not necessarily mean that the security implementation is insufficient.. However, it should prompt the system administrator to verify that the security provided is appropriate for his/her particular environment.

PRIMARY SECURITY PREPARATION

This section describes security features that are considered to be important considerations for most VMS systems.

VMS Accounts

The standard software distribution kit comes with 5 default accounts with commonly known passwords. These accounts are: SYSTEM, FIELD, SYSTEST, SYSTEST_CLIG, and DECNET. Several instances of unauthorized access to a VMS system have occurred because the passwords on these accounts were not changed after delivery. Of particular concern is allowing access to the SYSTEM account. Access to the SYSTEM account grants a user SYSTEM privileges that allow him/her to make any modifications to the system that he/she desires. In addition to changing passwords, the system administrator should disable those accounts not frequently used (i.e., FIELD, SYSTEST and SYSTEST_CLIG).

Security for Users

There are three important considerations for user security covered in the guideline: (1) assigning User Identification Codes (UIC), (2) using a default User Authorization File (UAF), and (3) using captive accounts.

User Identification Codes (UIC)

The UICs on a system should be controlled to assure that a unique UIC is assigned to each user. The UIC consists of a group number and a member number in the format [group,member]. The SYSGEN parameter MAXSYSGRP is used to define the set of UIC group numbers which would be used to grant the user system privileges. *Any UIC group number less than or equal to MAXSYSGRP has system privileges.* The value of MAXSYSGRP should range from 1 - 10 (octal) for most systems.

User Authorization File (UAF)

The UAF contains a record for each user account. The default UAF is used as a template for defining all user accounts. When the ADD command is used to create a new account, the default UAF is automatically used. For this reason the careful definition of the default UAF is *very important* to assure that users are not granted unnecessary privileges. The default UAF record should be reviewed to ensure that qualifiers that could pose a security problem do not exist (e.g., /PRIVILEGES=SYSPRV). Those parameters of interest to security and suggested values are:

LOGIN FLAGS: GENPWD NODISREPORT PWD_EXPIRED
PWDMINIMUM: 8
PWDLIFETIME: 180 (days)
PWDCHANGE: preexpired
AUTHORIZED PRIVILEGES: TMPMBX NETMBX
DEFAULT PRIVILEGES: TMPMBX NETMBX

The login flags used above indicate that the user's password will be machine generated, that a description of the user's last access to the account will be displayed upon login, and that the user's original password set by the system administrator is pre-expired and must be changed upon the first login. The privileges granted (i.e., TMPMBX, NETMBX) allow the user to perform basic VMS functions (e.g., create files, check on process status), perform functions related to a DECnet computer environment, and create a temporary mailbox to facilitate interprocess communication. These privileges are sufficient for the majority of VMS users.

Users who no longer require access to the system should be removed via the removal of the appropriate UAF entry. If possible, system administrators should not reuse the UICs of removed users. If a UIC is reused, the new user could inherit some or all of the access rights of the old user through existing Access Control Lists (ACL) entries.

Using Captive Accounts

Captive accounts can be used to limit a user's abilities and control access to the underlying VMS operating system by restricting them to a particular command procedure upon login. To make an account captive:

1. The flags CAPTIVE and DISCTRL must be set in order to disable the CTRL-Y interrupt.
2. In addition to the above flags, the flags LOCKPWD (only system administrator can change password), DEFCLI (user must use default command interpreter), DISWELCOME (disables welcome message), DISMAIL (disables mail delivery), and DISNEWMAIL (disables notification of new mail) should be set.
3. The login command file, LGICMD, that will be used must be specified (e.g., LGICMD=OPER.COM).
4. The number of subprocesses that can be spawned should be limited by setting the parameter PRCLM to 0 (i.e., /PRCLM=0).
5. The group UIC for the account should be unique.

An integral part of the captive account is its login command procedure. The login command procedure defines the functions that the user will be allowed to perform. Suggestions for preparing a captive command procedure are included in the guideline.

Dangerous Privileges

In most environments normal users should only have TMPMBX and NETMBX for privileges. A system administrator should be extremely cautious in granting additional privileges to a user. Below is a list of privileges considered outside the purview of normal users:

BYPASS - Allows a user to read, write, execute or delete any file on the system.

CMKRNL - Allows a user's process to change its access mode to kernel, execute a specified routine, and then return to the access mode that was originally in effect.

GRPPRV - Allows a user's process access to all files whose group number matches the group number of the process. With this privilege a user can indirectly acquire privileges granted to other group members.

LOG_IO and PHY_IO - These privileges allow a user to read and write directly to devices. Users with these privileges could destroy information on the system device, destroy user data, intercept user passwords, and expose information to unauthorized persons.

PFNMAP - Allows a user's process to map to specific physical pages of memory no matter who is using those pages.

READALL - Permits a user to bypass existing restrictions placed on files allowing the file to be READ and the protections on the file changed. Allowing the modification of file protections could lead to deletion or modification of the file.

SETPRV - Allows the user to grant himself/herself any privilege using the SET PROCESS/PRIVILEGES command.

SYSNAM - Allows a user to insert names into and delete names from the system logical name table. With this privilege the user could redefine critical system logical names, such as SYSS\$SYSTEM and SYSUAF, thus gaining control of the system.

SYSPRV - Gives a user the privileges of a system UIC when accessing files.

Protection of System Files

DEC-supplied system files are provided with default protection. The protections for these files should be reviewed periodically to ensure that no tampering has occurred. Since the list of files is lengthy, it should be printed out and compared to the list in Appendix C of the "DEC Guide to VAX/VMS System Security" [1].

Several of the system files (e.g., NETUAF.DAT, SYSUAF.DAT, AUTHORIZE.EXE) should be accessible only by system-level users and therefore, SYSTEM and OWNER should have read, write, execute and delete access and GROUP and WORLD should have no access (i.e., S:RWED,O:RWED,G,W) [2]. In addition, DEC has identified several system files (e.g., SYSHUTDOWN.COM, SYSTARTUP.COM, STARTNET.COM) that should not be granted WORLD WRITE access since it would allow an intruder to modify the file to perform unauthorized activity when run by the user. Complete lists of these files are provided within the guideline.

Break-in Avoidance/Detection

The VMS system provides several SYSGEN parameters that can be used to enable the detection and subsequent action of a possible break-in attempt. The VMS system is delivered with the default values set. These parameters should be changed as soon as possible. In particular, the default values of LGI_BRK_LIM and LGI_BRK_TMO should be changed periodically. If these values are learned, an outsider can modify his break-in technique to adapt to the set conditions. The LGI parameters, their default values and suggested changes to the values are shown in Table 1.

TABLE 1. LGI BREAK-IN PARAMETERS

BREAK-IN PARAMETER	PURPOSE	DEFAULT VALUE	SUGGESTED VALUE
LGI_BRK_LIM	NUMBER OF LOGIN FAILURES ALLOWED BEFORE BREAK-IN ATTEMPT ASSUMED	5	≤ 3
LGI_BRK_TMO	HOW LONG VMS WILL REMEMBER A LOGIN FAILURE	300 SECONDS	≤ 300 SECONDS
LGI_BRK_DISUSER	LOCKS OUT ACCOUNT WHEN LOGIN ATTEMPT LIMIT IS EXCEEDED	0	1
LGI_BRK_TERM	ASSOCIATES TERMINAL NAME WITH A USER NAME TO COUNT LOGIN FAILURES	1	1
LGI_HID_TIM	HOW LONG EVASIVE ACTION WILL BE USED	300 SECONDS	≤ 300 SECONDS
LGI_RETRY_LIM	NUMBER OF RETRY ATTEMPTS FOR LOGIN OVER DIALUP LINES	3	2
LGI_RETRY_TMO	HOW LONG BETWEEN LOGIN RETRY ATTEMPTS OVER DIALUP LINES	20 SECONDS	< 20 SECONDS

Auditing

It is usually not feasible to audit all events that occur on a system due to the additional resources required. However, a limited amount of auditing should be enabled that will aid the system administrator in tracking system events from a security perspective. Some suggestions for useful areas to audit are:

1. successful logins
2. unsuccessful login attempts
3. use of sensitive system utilities (e.g., AUTHORIZATION utility)
4. locking of user account (e.g., setting DISUSER flag in user's account)
5. break-in attempts, as defined by LGI parameters
6. changes to audit events (e.g., using SET AUDIT)
7. attempted access to audit records (e.g., accessing OPERATOR.LOG)

Audit information is printed to the operator's log file. The SECAUDIT command with five optional positional parameters can be used to selectively extract information from the operator's log file.

In addition to having the audit information written to an audit log, an alarm can be set such that an alarm message is written to a security operator's terminal. If the /ALARM flag is used, it is necessary to designate the terminal to which the alarm messages will be printed. The commands needed to audit and/or set alarms for the events mentioned above are given in the guideline.

DECnet Network Security

This section of the guideline addresses additional concerns for a system administrator implementing VAX/VMS systems in a network environment.

DECnet Account

It is important to control access to the DECnet account to minimize the possibility of remote users gaining unauthorized access to local system resources. The DECnet-VAX account currently does not have a requirement for a *privileged* default account. Therefore, only a non-privileged DECnet account should be created that has NETMBX and TMPMBX privileges. In most cases, the DECnet account should be restricted to NETWORK access only.

The delivered password for the DECnet account should be changed as mentioned earlier. In addition to modifying the password within the DECnet account, it must also be modified in the DECnet-VAX volatile and permanent databases.

File Access Listener (FAL) Account

The system administrator should create the FAL account to provide authorized access to the file system of a DECnet node on behalf of processes executing on any node in the network. When a FAL account is needed, the system administrator should create a restricted account and assign the FAL network object to that account. The guideline gives suggestions for creating this account.

Task 0

The default DECnet account and the TASK 0 object together enable an outsider to become a non-privileged user on your system [8]. The TASK 0 object can be accessed using the syntax: NODE::"TASK=com_file" or NODE::"0=com_file", where com_file is a command procedure on the remote node. A command procedure on the remote node can be activated using the TYPE command with the above specification. A user can get a command procedure to the remote node using the COPY command with a node-name specification. Under the standard setup, this means a user can COPY a command procedure to a remote node and immediately cause it to execute with the TYPE command. This method has been used in the past for virus attacks.

To prevent remote access to your system using the TASK 0 object the task object should be removed from the system. The command used to remove TASK 0 must be issued after the network has been started since TASK 0 is

recreated every time DECnet starts. This command can be included in the startup file to be performed automatically when DECnet is started.

Many system administrators find TASK 0 to be very useful (e.g., for managing multiple systems). If you do not remove TASK 0, there are steps that can be taken to increase its protection. These steps are described in the guideline.

Proxy Accounts

Proxy accounts are provided in VMS as an alternative to direct DECnet access which requires giving user name and password information in the DECnet command line which, in turn, travels across the network in clear ASCII form. Proxy login permits a user who is logged in at a remote node to be logged in automatically to a specific account at a local node, without having to supply any access control (e.g., user id/password) information. The remote user must have a proxy account on the remote node that maps to a local user account. The remote user assumes the same file access rights and default privileges of the local account. The local account for the remote proxy user should only have normal privileges (i.e., NETMBX and TMPMBX) to limit access.

Though the use of proxy accounts are the recommended method for allowing user/process access to remote nodes, the system administrator should be aware that if an intruder gains access to a system with proxy accounts, it is possible to gain access to multiple systems through the use of proxy accounts on each system. Restricting proxy account access and minimizing privileges granted to proxy accounts should minimize this threat.

Ethernet Connections

System administrators should be aware that all systems connected to an Ethernet are susceptible to the monitoring of all traffic, including cleartext passwords, across the Ethernet. This can be accomplished by putting a system connected to the Ethernet into "Promiscuous" mode.

Maintaining Data Integrity

There is an undocumented CHECKSUM command [4] that provides the means for verifying the integrity of a file. The checksum calculated by this command is not a cryptographic checksum and therefore it is possible to modify the file in such a manner that the checksum is not changed. However, if a cryptographic checksum procedure is not available this command should be used on sensitive files to provide a means for detecting file modification. The value provided by the CHECKSUM command should be encrypted if maintained on the system or recorded and maintained off the system (in a secure location).

ADDITIONAL SECURITY CONSIDERATIONS

This section describes other security features that are useful in certain environments, depending upon the system configuration.

Restricting Logins

Logins can be restricted by using a secure terminal server or system password. A "secure terminal server" can be used on VMS terminals to protect against password grabbing programs. The purpose of the secure server is to ensure that the VAX/VMS login program is the only program able to receive a user's login. The secure server can be invoked on any terminal but is especially necessary for terminals, directly wired or remote, located in unsecured areas. Once the terminal has been set up in this manner, the user must press the BREAK key followed by the RETURN key to initiate a login. The login then proceeds as usual. Some uses of terminals may be incompatible with the secure server. Some applications that use the terminal as a communications line may need to use the BREAK key for their own purposes. For example, terminal servers are incompatible with autobaud handling used

on switched or dialup terminals. The modem handling on such terminals performs the equivalent of secure server functions.

Most VMS systems provide password control at the terminal/port level (except LAT-11 Digital terminal concentrators) through the use of a system password. This additional protection is typically used to provide more stringent access control for publicly accessible ports/terminals. The commands necessary to set the system password for a terminal require the SECURITY privilege to execute.

Protection of User Owned Files

It is important that proper file protection attributes be associated with user-created directories and files. A system administrator can define default protection access control list entries (ACE) [3] that are associated with the directory within which the files are created. Since there may be more than one entry for a directory or file, an Access Control List (ACL) [3] of all entries is used. In an ACL, users are specified by identifiers that can be: (1) UIC, (2) an identifier established by the system administrator, or (3) system-defined identifier. "Identifier" ACEs can be used to restrict access for a particular user or group of users. In addition, using a default ACE on a specific user directory ensures that the UIC, identifier, and alarm protection attributes are associated with all files created within the given directory. Though it is important to protect user files, *ACLs should not be used indiscriminately since they require additional processing time and dynamic memory.*

Device Protection

In many environments it may be important to restrict user access to certain devices (e.g., disk packs, printers, tape drives, terminals). User access to these devices may be controlled via the use of ACLs and identifier ACEs. To use identifier ACEs with objects other than directories or files the /OBJECT_TYPE qualifier must be used in the SET ACL command.

VMS C2 Security Features

Some of the security features provided by VAX/VMS are directed toward the requirements designated by a DOD Class C2 rating for computer systems [6]. Currently, only VMS Version 4.3 has been evaluated by NCSC and therefore is the only version given the C2 rating. A system maintaining C2 protection must provide discretionary access control, individual accountability, auditing of security-related events, and resource isolation [6]. C2 protection for VMS systems can be accomplished through the use of user identification, user authentication, protected audit trails, and object protection and reuse. In order for the system to provide C2 protection, all of the necessary mechanisms must be properly implemented.

VMS provides discretionary access control mechanisms for access to named objects in the form of UIC-based protection and ACLs. Individual accountability is provided by enforcing restrictions on user accounts. These restrictions include: use of unique UICs, use of a password for each account, assignment of unique user accounts, and restricting the use of the autologin feature since it associates an account with a terminal instead of a user. Auditing is provided on VMS systems through the use of ACLs and the SET AUDIT command. The audit trail produced is written to the operator.log file and can be protected through the use of an ACL. Object reuse is not allowed in C2 systems. Reuse of system memory pages is protected by the memory management subsystem. Reuse of disk blocks is protected by the highwater marking and erase-on-delete features. The guideline provides further description of the mechanisms used for C2 protection and gives the necessary commands to provide this protection.

DEC Security Updates

This section of the guideline describes three security updates that have been released over the years to fix security problems found in different versions of VMS. If your system is operating under one of the VMS versions mentioned in this section of the guideline, the corresponding security update package should be installed immediately. If you have not received the updates appropriate for your system, you should notify your DEC

representative immediately. If the required security updates are not installed, your VMS system may be at risk to unauthorized use.

SUMMARY

This paper gave only brief descriptions of important VMS security issues. However, the guideline is available from the author:

Debra L. Banning
213/542-6090
dlb@sparta.com

The guideline provides all command sequences necessary to implement the security features described within this paper. It also provides references to VMS documents to obtain additional information about broad topics (e.g., developing an ACL).

REFERENCES

- [1] "Guide to VAX/VMS System Security," VAX/VMS Version 4.2, Digital Equipment Corporation, Maynard, Mass., Order #AA-Y510B-TE, July 1985.
- [2] Memorandum from M. Morgan, DEC, to P. Siebert, DOE HQ, Subject: VMS Security Action Plan - Additional Guidelines, dated 16 February 1988.
- [3] "Guide to VMS Security," VAX/VMS Version 5.0, Digital Equipment Corporation, Maynard, Mass., Order #AA-LA40A-TE, April 1988.
- [4] "Security for the New VAX Manager," Robert Hansen, DEC Professional, June 1988, pp. 58 - 63.
- [5] "Digital Software News and Views - VMS Security Patch #3," Allan Van Lehn, SDE NewsLetter, Lawrence Livermore National Laboratory, Dec. 1988.
- [6] Department of Defense Standard, "Department of Defense Trusted Computer System Evaluation", DOD 5200.28-STD, December 1985.
- [7] U.S. Department of Energy, Washington, D.C., Office of Safeguards and Security, "Classified Computer Security Program," Order 5637.1, dated 1-29-88.
- [8] Sharp, Nigel, "Default DECnet Accounts - A Boon for Security?," Views on VAX, 1989.

SNEAKERNET: GETTING A GRIP ON THE WORLD'S LARGEST NETWORK

Captain James B. Hiller
Chief, Network Software Security
Space and Warning systems Center
Stop 32
Peterson AFB, CO 80914-5001

Abstract

This paper explores the issues of SneakerNet (S-Net), the term often used to describe the transfer of removable media from system to system. It offers a description of S-Net, examples of how and why it exists, and types of problems which can result. S-Net identification, threat analysis and negation, and documentation is developed. Finally, results and conclusions of a case study using a single large computer facility are shown.

Introduction

Our computing environments have come a long way in ten years. We've left the safety of the large computer center and placed virtually the same computing power of an older mainframe on the desktop in every type of workplace - office, home, laboratory, airplane, tractor trailer, tent. Electronic networks have blossomed and created a spider's web around the globe.

The security community has responded to the challenge by identifying, analyzing, and reducing every risk imaginable, normally just after each one is exploited for the first time. This is another application of Murphy's law.

We have a complex array of countermeasures predicated on identification, from the Orange book right on down to add-on packages that can make a microcomputer just about impenetrable. Network security products and measures have also begun to proliferate, and we're tackling the issues of international security standards. It looks like we're on top of the problem.

Are we?

What about that little urchin, the floppy disk? A review of products shows that we have a myriad of solutions, from colored disks to theft-deterrent shipping pouches. Unfortunately, none of these high-tech solutions will protect against a low-tech security problem: SneakerNet (S-Net).

What Is SneakerNet?

S-Net is the virtual network that links every computer in the world via an infinite resistance, abundantly available medium-air. the link between systems is completed and broken very fast, in the time it takes to insert a piece of media and read data.

S-Net exists in just about every conceivable computing environment, provided the environment allows use of some type of removable media. Media type is unimportant. Floppies, tapes, removable disks and cartridges, and the CD format all provide the opportunity for a computer to use S-Net.

Specific Examples

Mainframes - Any large system which has a removable disk or tape drive. One frequent situation: software is developed and used at two different sites. It may be tested at either site, or at both to some degree. Chances are, even if no testing is done at the operational site, compilers and editors are available "just in case." This provides opportunities to leave trojan horses that can use data imported by the S-Net connection. Consider too that most large systems are cold-booted from tape drive or floppies at the console.

Minicomputers - Most minicomputers or workstations have the same types of I/O devices as mainframes. This is the system of choice for most uses where even limited funds are available for computing resources. Electronic networks are used when possible as they are faster than media transfer. However, these systems still have some capability to use removable media. This is often the source of "extra" (pirated) and malicious software.

Microcomputers - S-Net is normally the first network used with a new micro. It is cheap and available. The last time you moved a file to another system to use its printer, S-Net connectivity was achieved. What about the freeware you brought to the office from home? Or the vendor-provided source disk with the latest word-processing package? Each is an example of S-Net.

To sink the point, let's look at a hypothetical situation. While imaginary here, odds are strong that it exists somewhere.

Hypothetical S-Net Scenario

Ajax Software uses a commonly available suite of mainframes from Wasp for software development and testing. Upon development acceptance of each version, the executable code is bonded, sealed with a checksum, and taken to the user site via removable disk by two employees. Once there, it is loaded on the operational Wasps for acceptance test and user implementation.

Ajax uses a contractor, SureSoft, as well as in-house personnel to develop and test its software. They have a versatile, properly controlled development facility. Modem use is prohibited. SureSoft personnel come to this facility to do their work alongside Ajax employees.

Several factors have converged over the years to make this impractical. Software change requirements have escalated rapidly. Software engineering has emerged which requires Ajax and SureSoft to spend as much time on design documentation as on actual code. Both have a larger staff than in the past to handle this. Local travel is a huge expense. A tightening budget forces Ajax management to reconsider ways of doing business.

Modem and direct lines are rejected due to the obvious threat of malicious mischief and deterioration of configuration control. The best answer which preserves the almighty air gap is for SureSoft to develop source in its facility and bring it to Ajax.

SureSoft buys an Analog Equipment minicomputer, popular for its universal market and vendor support. The company already has a large investment in ABC microcomputers. These will be used by employees rather than Analog Equipment terminals, most of which are more costly than inherently flexible microcomputers. Word processing software is suitable for code development. SureSoft will install a local area network in the future, but currently employees can bring floppies to the Analog Equipment machine and run a utility to transfer source developed on the ABC Micros to the Analog equipment system. The code is then provided as a deliverable to Ajax, which has a transfer utility for the Wasps.

As Ajax' contractor, SureSoft has a standard security program and policies. Most of these are collecting dust, but employees still wear their badges and supervisors watch their staff closely. SureSoft has a profit to make as well, and encourages staff creativity and cost-cutting. While pirated software is prohibited and employees are aware of viruses through education by the security staff, SureSoft permits employees to use personally owned systems in its facility. Controls on these systems are encouraged but not required. Modems are prohibited.

Since many employees are using their own machines and can work around the clock, it is virtually impossible to enforce controls. Built-in modems add to this problem.

While unlikely, it is quite possible for someone with knowledge of this set-up to exploit it using S-Net. The obvious way would be for the individual to place malicious code on each of the machines in the process. This is simple to do and can be disastrous for Ajax' customer. As the code runs across systems, it only transfers code which will run on the current machine plus the systems down the road. When the post-mortem is conducted, it will take months, if not years, to trace the problem.

There's a very big hole in the process which can be easily plugged. Before proposing a solution, we should examine the factors that lead to use of S-Net and categorize the problems which result from its use.

Reasons for S-Net

Reasons people transfer media among machines cover a broad spectrum. This includes operational necessity, perceived security, flexibility, compatibility, logistics, cost, availability of other media, and simplicity. The number of possibilities is endless.

The hypothetical situation is a good example of operational necessity. Two different companies with individual platforms need to transfer information routinely but have not developed a wire connection. Media transfer is the most expedient solution.

Perceived security is another reason that media transfer is preferred. The "air gap" has always been viewed as a superb protective measure. We will see that this is not always true.

Flexibility, especially in the microcomputer environment, is a large contributor to the use of S-Net. Invariably, all but the simplest of micro collections have an array of peripherals with different capabilities. In lieu of hard connections, either due to cost or poor planning, S-Net is a fast, economical way to move data between machines to take advantage of various peripherals. It is also used to overcome incompatibilities between connected systems. In most cases, media formats are identical. Copying is the fastest approach.

Compatibility between systems, or lack thereof, is another inspiration for S-Net. While microcomputers boast a high degree of compatibility in communications, there are many instances where this is not the case. For example, when several classes of systems, such as micros under MS-DOS and minis under UNIX, exist in the same office, it may be easier to transfer media than to purchase software and interconnections. It may only take a few minutes to write a format conversion routine, while it could take months to acquire communications assets. The more heterogeneity in a mission environment, the more this becomes true.

Another factor may be logistics issues. Considering the Ajax example, if the development and user site are miles apart and the transfer rate between the two is low or infrequent, it may take years of travel reimbursement to two people to equal the cost of installation of a secure, dedicated line between the two sites.

Cost is an underlying theme. The cost of electronic connections, regardless of medium, escalates very quickly in proportion to the distance covered. Physical media transfer costs are relatively low in comparison, regardless of the distance. The primary factor which increases physical transfer costs is frequency. When combined with distance, high use rates may make the electronic path more cost-effective. However, this only occurs when both distance and frequency are high. In many cases, this will not be the case.

Availability of other media is similar to cost as a causative factor. Connectivity requirements are often overlooked in planning or simply not recognized until a system is in use. New requirements are developed during the operational phase. Until the requirements become a real installation, physical transfer is often the only available means to get the job done. This is accented by the need to "get the job done NOW." While such needs are often overstated, they appear to be real at the time. Physical transfer is used as the only method available.

Simplicity may be the biggest cause for use of S-Net. Most users fall into the novice category. While a vast array of connectivity capabilities may be available, most novices will hesitate to use them, preferring instead to issue the almighty "copy" command as the path of least resistance. Until a novice is trained in electronic transfer, physical transfer will often be the method of choice.

Problems Caused by S-Net Use

Various studies have shown that most system problems stem from unintentional human actions, such as keystroke errors or poorly implemented procedures. Since S-Net is merely a non-real-time, off-line version of a network interface, human error can exploit a system as easily via S-Net as it can via an electronic connection. Any type of problem that can occur from a mistake during on-line use can also affect the system via S-Net. Fortunately, many errors are caught by error-handling mechanisms.

SureSoft buys an Analog Equipment market and vendor support. The company microcomputers. These will be used by terminals, most of which are more costly. Word processing software is suitable for local area network in the future, but the Analog Equipment machine and run ABC Micros to the Analog equipment deliverable to Ajax, which has not

As Ajax' contractor, SureSoft of these are collecting dust, and watch their staff closely. SureSoft creativity and cost-cutting are aware of viruses through the use of personally owned software encouraged but not required.

Since many employees clock, it is virtually impossible problem.

While unlikely, it is dangerous to exploit it using S. Net. The malicious code on each terminal is disastrous for Ajax' code which will run on the post mortem is conducted.

There's a very big problem in recognizing a solution to categorize the problem.

Reasons people try to include operational availability of cost and employees.

The hypotheticals of different companies routinely but have no different solution.

Received security has a low cost and a low cost.

Documentation

Accreditation for each system should document all data paths which do not use electronic connection. Those that do use electronic means are usually already documented. A format for this would include a path description, medium, medium of connection, and a list of procedural controls.

The main benefit of this information is to assist the security officer in recognizing problems when trouble develops. Thus, the investigation will not overlook these areas. This would be easy to do when they are not identified. The documentation should be included in the accreditation package, filed for historical purposes, and updated as it changes.

The information required is fairly straightforward. In addition to documenting the existence of the path, the information can be used by the security officer to identify problems to be resolved prior to occurrence of a problem. Examples include omissions in procedures and future problems that may occur.

One item which could be added to the worksheet is an approval block. This would require management to be aware of and approve each path. We have seen several instances in which the user chose any means available to complete a task, and management was pleased as long as the product was suitable. Management understanding never included the task method. When management was shown how the task was done, it found that the media use was unacceptable due to security or configuration control concerns.

Media Control

To ensure that all paths are identified, and to enforce media control overall, a central point on the room or facility perimeter should be used for passage of all media.

The next step is to develop a written form upon which media transfer approval is requested and approved by the security officer responsible for the system on which the media will be used or from which it was removed. To provide complete control, one approval should only be valid for a single pass by the entry point. Allowing a single round trip can reduce the administrative workload. However, there should be NO allowance for media to be transported as desired over a period of time. In essence, the responsible security officer should have to approve every transit of media past the control point. This way, complete control and cognizance over activities is maintained.

Use of the form and required approval by the security officer serves several purposes. It validates identified S-Net paths and allows new ones to be discovered. Further, security policy often requires the security officer to review and approve all system changes, including software changes, for security impact assessment. Since the security officer is often not in the loop as changes are made, the paper trail provides a way for him to be aware of activity on the system. The paper trail also helps identify maintenance activities which routinely include vendor diagnostic routines that the technician introduces via media.

Intentional acts can also use S-Net as an exploitation medium. The primary example is the virus. This can be generalized to include any anomolous software or data that causes a system to behave in ways other than those expected.

The primary difference between malicious activity using an electronic network and using S-Net is that, with S-Net, there is no network monitor which can track activities leading up to and occuring after the anomaly is introduced. The only indicator might be an audit record which shows the mounting of a media volume. On microcomputers, it is unlikely that this type of event will be audited. Even on systems that do record media use, it would still be virtually impossible to trace the source of the problem unless media use had been very low prior to the problem. An aggravating factor is that the problem may be brought to bear over a very long time, such as is the case in viral infections. Had an electronic path been used, the perpetrator would have been likely to cause the problem over a short period of time. Use of media can be made piece-meal over an extended time frame, avoiding detection. When combined with the fact that media use is often "userless," the result can be an anonymous disaster.

SneakerNet Solution

S-Net is not a tough problem requiring a highly technical solution. The most effective solution is probably one of the least expensive measures available. Our solution includes three activities--identification, threat analysis and negation, and documentation. To enforce requirements for media transfer, a method of positive control has also been devised.

Identification

During system accreditation, each authorized data path should be identified. For systems with a documented system security policy, the paths should be identified in the policy and documented to users through operating instructions. The security officer should also make this data part of initial user training so that each user is aware of what is and is not permitted. Training should include the possible impacts associated with S-Net to motivate the user to comply with procedures. For smaller systems, it may suffice to have the user identify paths that will be operationally necessary.

Threat Analysis and Negation

Once each path is identified, the security officer can view details of the path and its use. Specific countermeasures, usually procedural, can be developed and employed.

For example, media transfer of baselined software from the test facility to the operational site may require absolute assurance of integrity of information from creation through final loading. Thus, a checksum may be needed for data preservation and change detection. If the software or media is classified, appropriate precederal controls may also be needed. In most cases, documenting the presence of the path is sufficient. Measures should be based on the sensitivity and criticality of the information being transferred and of the systems involved.

Documentation

The accreditation for each system should document all data paths which do not use an electronic connection. Those that do use electronic means are usually already covered. A format for this would include a path description, medium, medium classification, and a list of procedural controls.

The main benefit of this information is to assist the security officer in recognizing paths when trouble develops. Thus, the investigation will not overlook these sources. This would be easy to do when they are not identified. The documentation should be included in the accreditation package, filed for historical purposes, and updated as it changes.

The information required is fairly straightforward. In addition to documenting the presence of the path, the information can be used by the security officer to identify other issues to be resolved prior to occurrence of a problem. Examples include shortcomings in procedures and future problems that may occur.

One item which could be added to the worksheet is an approval block. This would force management to be aware of and approve each path. We have seen several instances in which the user chose any means available to complete a task, and management was pleased as long as the product was suitable. Management understanding never included the task method. When management was shown how the task was done, it found that the media use was unacceptable due to security or configuration control concerns.

Positive Control

To ensure that all paths are identified, and to enforce media control overall, a central point on the room or facility perimeter should be used for passage of all media.

The next step is to develop a written form upon which media transfer approval is requested and approved by the security officer responsible for the system on which the media will be used or from which it was removed. To provide complete control, one approval should only be valid for a single pass by the entry point. Allowing a single round trip can reduce the administrative workload. However, there should be NO allowance for media to be transported as desired over a period of time. In essence, the responsible security officer should have to approve every transit of media past the control point. This way, complete control and cognizance over activities is maintained.

Use of the form and required approval by the security officer serves several purposes. It validates identified S-Net paths and allows new ones to be discovered. Further, security policy often requires the security officer to review and approve all system changes, including software changes, for security impact assessment. Since the security officer is often not in the loop as changes are made, the paper trail provides a way for him to be aware of activity on the system. The paper trail also helps identify maintenance activities which routinely include vendor diagnostic routines that the technician introduces via media.

Since entry controllers are often not aware of who is responsible for each system, a list of security officers can be developed and given to the controller for reference. This way, the authority of the approving official can be verified. In time-critical environments, there may be a need to have provisions for others to approve the transfer. This reduces the likelihood of required, spur-of-the-moment activities being aborted but still provides a trail for review after the fact.

For contractors, we have also required that their sponsoring government activity request the transaction. We have many contractors working for other agencies who must use our systems. We have no contractual capability to ensure that the activities occurring are necessary or valid. We normally do not know what kinds of activities have been required of the contractor. Thus, we are able to put the onus on the proper activity to ensure the propriety of the contractual action. This has also been useful as we have found that many times, other agencies do not maintain rigid control over the activities of their contractors. This forces them to maintain a higher level of awareness.

The final step is to collect the forms as the transactions occur. They are validated by the entry controller to ensure compliance with content requirements. They are periodically sent to the organizational computer security manager for general quality control checks and analysis, and finally forwarded back to the responsible security officer for his use and filing.

S-Net Identification Case Study

After beginning to use the techniques described, it became clear that a phenomenal amount of media was traversing our entry points. In reviewing the transaction forms, we also found that agencies conducting transfers were not always aware of the need to protect output media at the same level of classification as the system on which it was created. We also found that input-only media was not being physically write-protected.

To determine the detailed nature of the media flow past the entry points, we commissioned a study by the organization responsible for the flow. The results were eye-opening.

This environment involves a contractor facility for development and analysis; our own three facilities (referenced as "our facility"), used for development, integration, and development testing; and another related facility which is the customer of all developed software and data. The source of media implied in the results is the contractor facility.

Within our facility we have a multitude of systems, from microcomputers and test devices to mainframes. The mainframes use about two dozen removable and fixed disk drives along with about a dozen magnetic tape drives. Several of the larger systems are directly connected to one another. All can be interfaced through local patch panels.

System interface devices range from dumb terminals to fully-configured minicomputers. These include an array of fixed and removable media devices.

By design, S-Net connections exist between most of these systems. Connections also exist with external development and testing agencies as well as with the end user of software systems developed, integrated, and tested in our facility.

The case study includes only one of several agencies with which we interface. This particular study involves media associated with only five of our projects handled by that agency.

Case Study Results - Data Paths

1. Tapes containing complete source code modules or source code changes are brought into our facility for installation on our systems. The source code is generated on microcomputers, transferred to a minicomputer using vendor or quasi-public domain software, and finally transferred again to the target system.

2. Blank tapes are loaded in our facility. Source code records, system performance data, and system analysis scenarios are copied and removed from our facility for transfer to and use on mini and microcomputers.

3. Tapes containing mission scenarios are generated, moved to our facility, used for analysis, and then may be moved back to the origination point.

4. Data generated on tape for use in stress testing is brought to and used in our facility. These tapes are degaussed.

5. Floppy disks (3.5" and 5.25") are brought into our facility as a consequence of other business. These are not intended for use on our systems, but such use is not precluded.

6. Floppy disks (3.5" and 5.25") containing source code are generated in our facility, taken out, and then returned to the same or another facility for analysis on systems within.

7. Non-standard format floppy disks (3.5") are generated and brought into our facility for use on test equipment. These normally contain source and executable code as well as data.

8. Floppy disks (3.5", 5.25", and 8") containing system documentation, system data used for analysis, and system data resulting from analysis are brought into our facility.

9. Floppy disks (5.25") and removable disk cartridges (10 MB) which contain backup files, executable developed software, data files, vendor software, and system documentation are brought into our facility for system testing and final version release.

10. Floppy disks (5.25") containing various types of data are received from world-wide locations, sent to another facility, brought back to our facility after analysis, and redistributed to the original locations.

11. Floppy disks (5.25") containing executable communications software and related data are brought into our facility for use in testing and then transported to another related facility for testing and eventual operational use.

12. Tapes containing modified executable code are generated in our facility and transported to another related facility for operational use.

13. Tapes containing mission analysis and system performance data are generated in a related facility and transported to our facility for use in analysis. These tapes are then degaussed.

14. Tapes and removable disk packs are transported both ways between our facility and a non-related facility for use in disaster recovery testing. This occurs very infrequently.

15. Laptop computers used in system analysis, software development and testing, and system documentation are brought into our facility and removed.

Case Study Results - Numerical Study

A complete study of individual transactions to determine specific trends and problems is ongoing. However, a brief overview of the individual transactions associated only with the flows shown above, covering one month, finds that 55 media transportation requests occurred, moving 272 pieces of media past the entry point.

These figures do not show the amount of computer equipment which may contain data storage devices, such as EPROMs, laptop computers, hard disk devices, and other forms of permanent storage. They also do not show processing devices of any sort, a basis for an entirely different study.

Assuming a 50% frequency of 5.25" floppy disks and a 50% frequency of 6250 bpi, 700 foot magnetic tapes, this is approximately 1256 MB of data or storage capacity moving past the entry point in one month. This does not account for extremely high density media that also moves to and from the facility.

How much data moves in and out of your facility or office?

Analysis

The numbers alone show an urgent need for some sort of identification and control of media flow. More than that, the diverse types of media flow discovered, related to only a fragment of the operations in our organization, show several situations worthy of closer examination:

Path 1 is similar to the hypothetical situation. It is a potential path of attack for malicious or inadvertently fouled software or data. Paths 2, 3, 6, 10 are examples of how security procedures are ignored when operationally expedient methods are developed. Since there is no convenient way to verify that media created which is intended to be non-sensitive is in fact non-sensitive, it may be at risk in a less vigilant environment where all data is assumed to be non-sensitive. Conversely, path 4 shows that security measures may be properly exercised when expediency is not a concern.

Paths 7-10 exemplify a direct opportunity for introduction of unknown data or software. Path 5 shows that fortuitous sources of potential problems routinely transit the facility. These are normally not exploited due to lack of intent rather than to prevention measures. Paths 11, 12, show it is possible for problems encountered in our facility through introduction from other sources to be transferred to other related facilities. Finally, path 15 is a situation that requires very special attention in any case.

Conclusion

The high volume of media transfer results in the likelihood of serious problems should unknown or unintended information be transmitted through these channels. Lack of attention to this problem can result in stymied attempts to prevent or detect malfunctions. In a software production environment, or any situation in which there is a reliance on computer systems, it is extremely important to recognize this problem and solve it. Positive entry control techniques are crucial to the effort.

While there may be useful, automated solutions to this problem in the future, there does not seem to be one now. The easiest, most economical solution currently available is the identification, documentation and control technique. While not perfect, use of this method promises to reduce unforeseen problems to a minimum. In addition to providing a catch for unintended problems, the method establishes a means for identification of related security issues and system changes which should be evaluated from a security perspective. Finally, the very presence of a viable procedure will serve to deter malicious activity in much the same way that even a simple car alarm will cause the thief to move along before ruining your day.

A SOCIO-TECHNICAL ANALYSIS OF A U.S.A. NATIONAL COMPUTER SECURITY CONFERENCE¹

Stewart Kowalski



**Project for System Integrity and Information Security
Dept. of Computer and Systems Sciences
University of Stockholm & Royal Institute of Technology
Electrum 230
S-164 40 Kista Sweden
Phone : +46-8-161611 Fax : +46-8-703 90 25
e-mail: stewart@dsv.su.se**

Abstract

The United States computer security community is one of the most influential forces in the international computer security arena. It has achieved this position by investing massive amounts of resources. Just the man/hours spent attending their annual national computer security conference often amounts to a yearly 35 man/years investment by the community. This massive investment has produced a socio-technical infra structure that is dynamic and vibrant but also very confusing for observers outside the United States. This confusion is often detrimental for the acceptance of U.S. standards by the international computer security community. This paper attempts to shed light on the socio-technical infra structure of the United States computer security community by analyzing the proceedings of the 12 th. United States National Computer Security Conference using the SBC socio-technical analysis methodology. The papers presented at the conference are classified using the SBC national-supranational analysis methodology. Once the papers are classified the SBC methodology is used to suggest trends and tendencies within the United States' computer security community socio-technical infra structure.

Introduction

The annual U.S. National Computer Security Conference has become the "mecca" for individuals involved in computer security. The conference covers a broad range of computer security topics and the 12 th. conference was organized into five tracks:

- a) Research and Development
- b) Systems
- b) Management and Administration
- c) Education and Ethics
- e) Alternate Papers

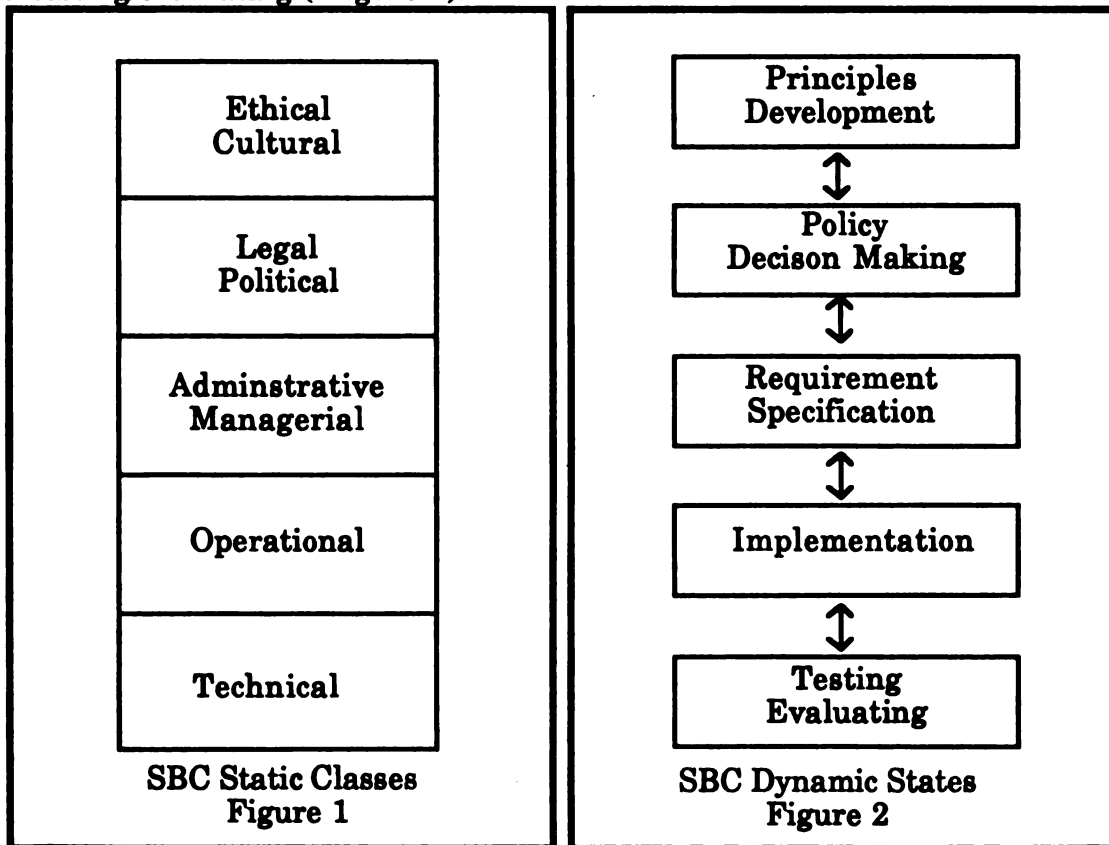
¹ A version of this paper was originally published in Computers & Security Volume 10 No 3 1991 and is being published here with the full permission of the publishers, Elsevier Advanced Technology, Mayfield House, 256 Banbury Road, Oxford, OX2 7DH, UK. This paper has been funded by the Swedish IT4 Programme.

The 12 th. conference attracted close to 2300 delegates. The delegates come from a broad selection of different interest groups involved in computer security issues in the United States. Computer security vendors and computer security users from both the public and private sector attended. About 5 % - 10 % of the delegates at the 12 th. conference came from outside North America.

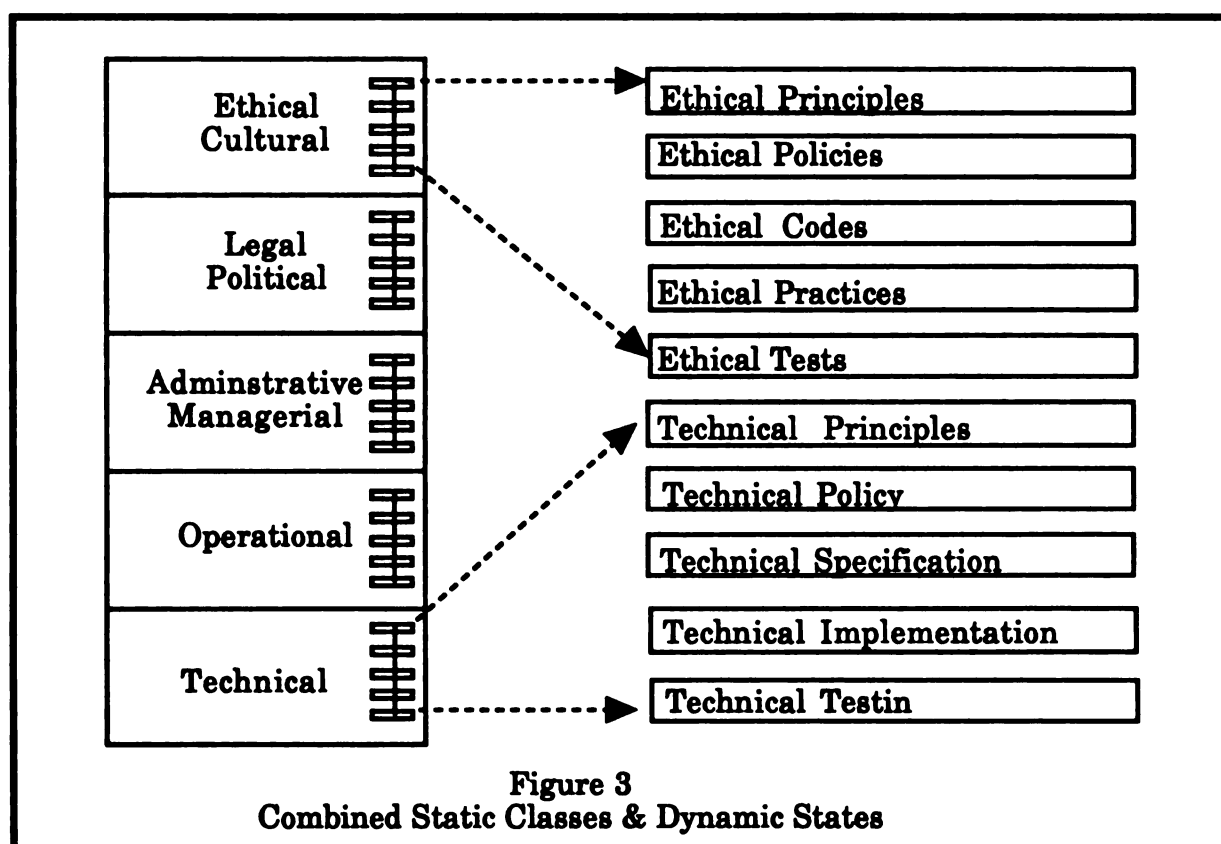
To many observers the National Computer Security Conference is a bit like a four ring circus [4]. The problem is that a great deal is happening in the United States in the area of computer security and it is very difficult to put all of it together at one conference. Consequentially it is also very difficult to understand what direction the United States computer security community is taking after attending one of these conferences. To attempt to overcome this difficulty the conference was analyzed using the SBC socio-technical analysis methodology being developed at Stockholm University [9,10,17].

SBC Analysis Methodology.

The Security By Consensus (SBC) socio-technical analysis methodology consists of a dynamic and a static classification scheme. The static scheme divides the computer security problem and solution space into five classes, or subsystems. These five subsystems are; ethical, legal/political, administrative/managerial, operational and technical (Figure 1). The dynamic classes, or class states, are taken from the traditional design life cycle model and include; principles development, policy decision making, requirement specification, implementation and testing/evaluating (Figure 2).



The dynamic model and static model are then integrated together (Figure 3).



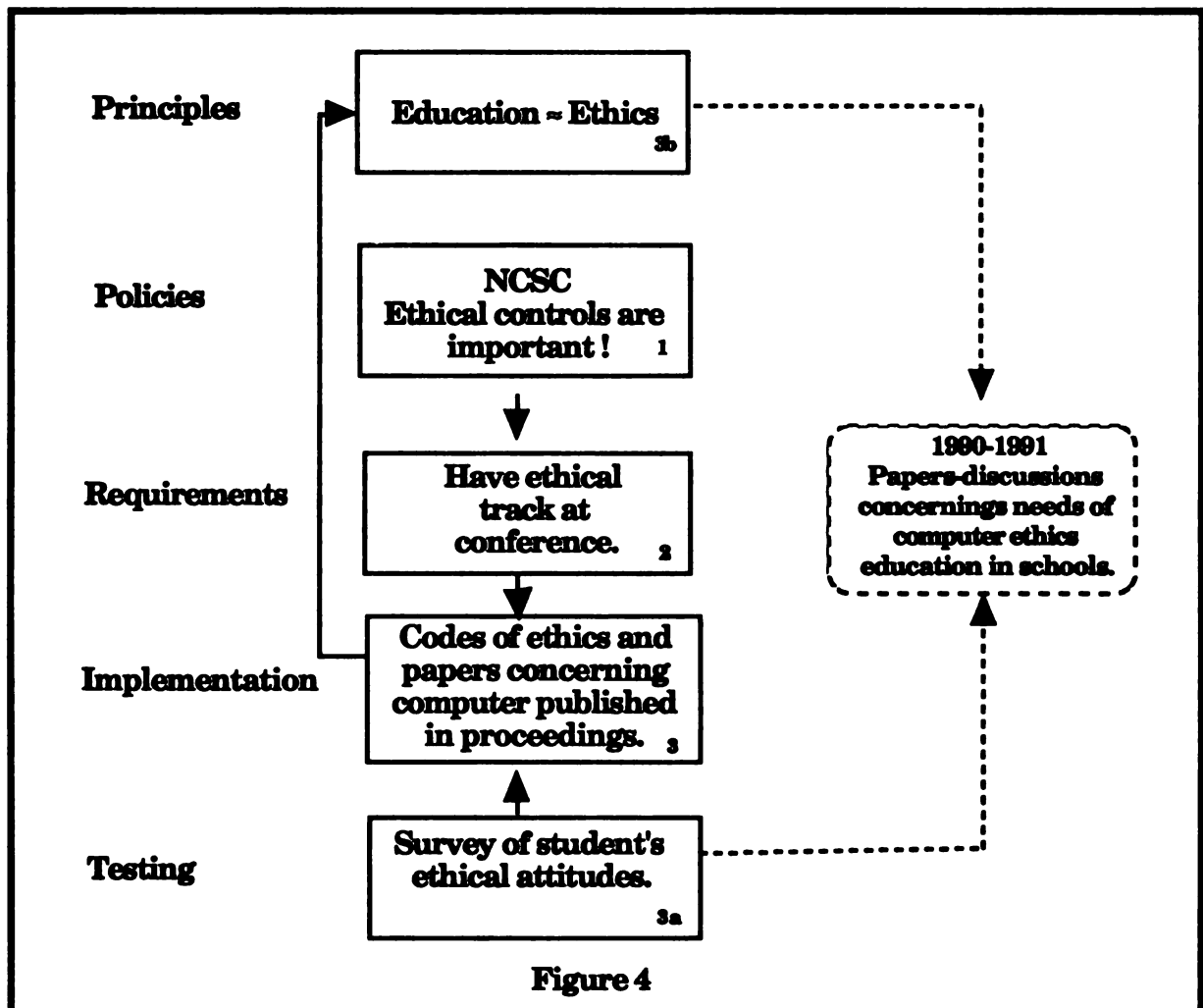
The general thesis of the SBC analysis methodology is that problems at the national and supranational level can be understood by examining the delays and inconsistencies in the material and information feedback cycles between the different classes or subsystems. For a more detailed explanation of the SBC analysis methodology at the national and supranational level please see reference [9,10]. What follows is a SBC analysis of the conference using the static classes of ; ethics, legal political, administrative and managerial, operational, and technical as headings.

Ethical Layer

Although it is not the first time that ethical papers have been presented at the conference [7] it is the first time that ethics have been presented in a separate track. It should be noted that in a 1988 SBC analysis of the United States [10], the ethical subsystem in the US was their slowest and most undeveloped subsystem.

It appears that from a dynamic perspective the computer ethical subsystem in the United States seems to be moving from the policy state into the principle and requirement states. That is to say that there has been a general policy decision in the United States that ethical controls are important and this has caused activity in the principles and requirement subsystems.

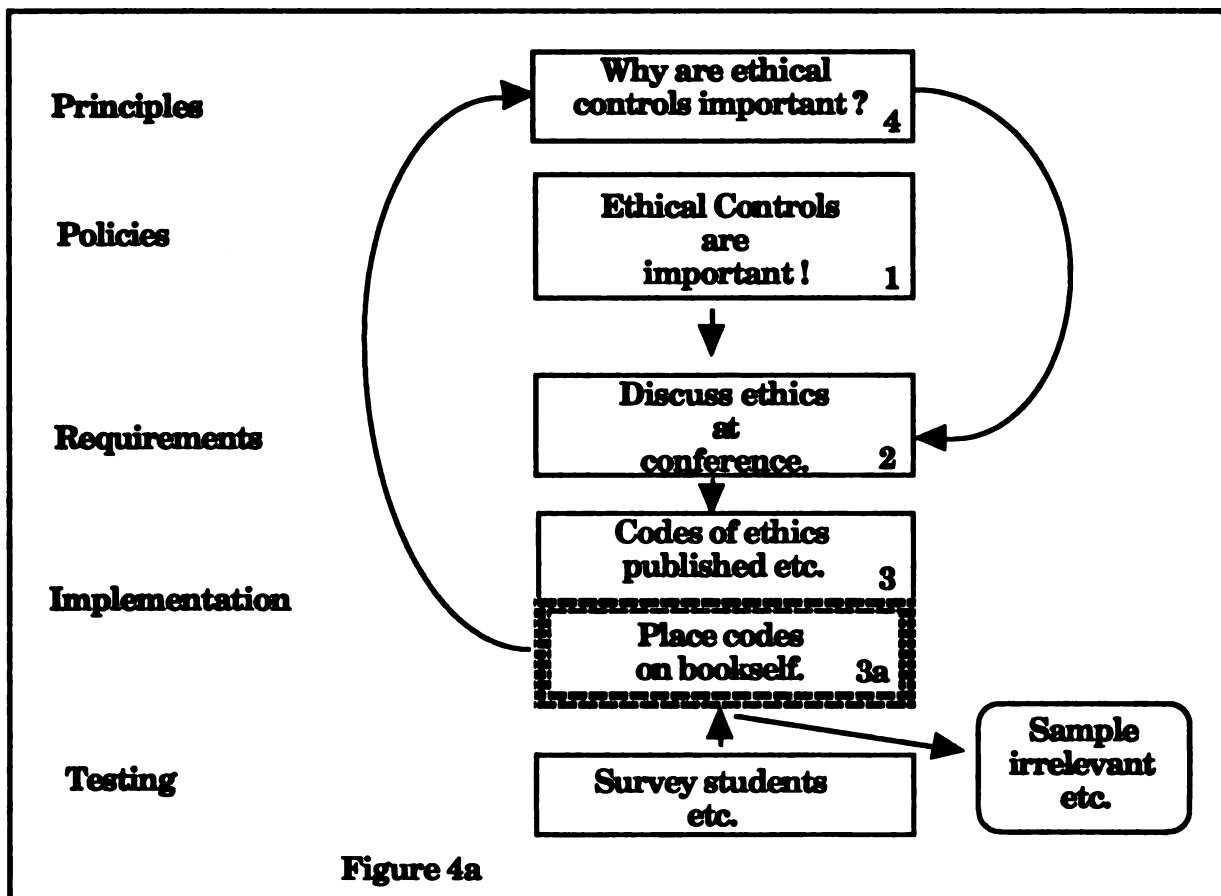
Figure 4 is a SBC flow diagram of the computer ethical subsystem of the United States from a review of the reports presented at the conference.



In figure 4 the policy decision that ethical issues are important (Step 1) is shown leading to an ethical track at the conference (Step 2), which in turn leads to research and debate on ethics and ethical principles (Step 3a,3b). Unfortunately not much new in the way of principles were presented at the conference. Two of the papers presented results from surveys of ethical attitudes among students [3,11]. The surveys were similar in nature to SIIS survey of Swedish students [8] but were however relatively small in size with only 100 to 120 subjects. It is difficult to say how relevant these surveys are for a US population of over 220 million. The Swedish students surveyed by the SIIS project do not appear to differ from the American students. However, there is not enough information in the papers presented at the conference to make a comparison of the Swedish and American students.

What was of interest in the presentations of the papers given at the conference was the strong coupling of educational principles and ethical principles. Also in figure 4 there is an indication that in the U.S. ethical control subsystem there are some activities in the requirement/implementation and testing states. In the presentation *"Information Ethics, A Practical Approach"* [6] the issue of computer security ethics is examined from an implementation perspective. Thus it appears that the ethical control subsystem is moving through the different life-cycle stages and there are individuals trying to take the codes of ethics and use and test them. In figure 4 the dotted lines point to an hypothesis that was made when this paper was drafted in January of 1990 that the issue of education and ethics would become a "hot topic" in the United States. The fact that an ethical track was not included in the next years conference and that education became a session at the conference rather than a track indicates that this hypothesis has been proven false.

Figure 4a is an explanation using the SBC methodology as to why this may have occurred. There is a tendency in the ethical control subsystem that policy decisions do not have enough momentum to complete the full lifecycle process. Codes of ethics are written and discussed in universities and perhaps every now and then at conferences but they are never really implemented or tested in the larger system.



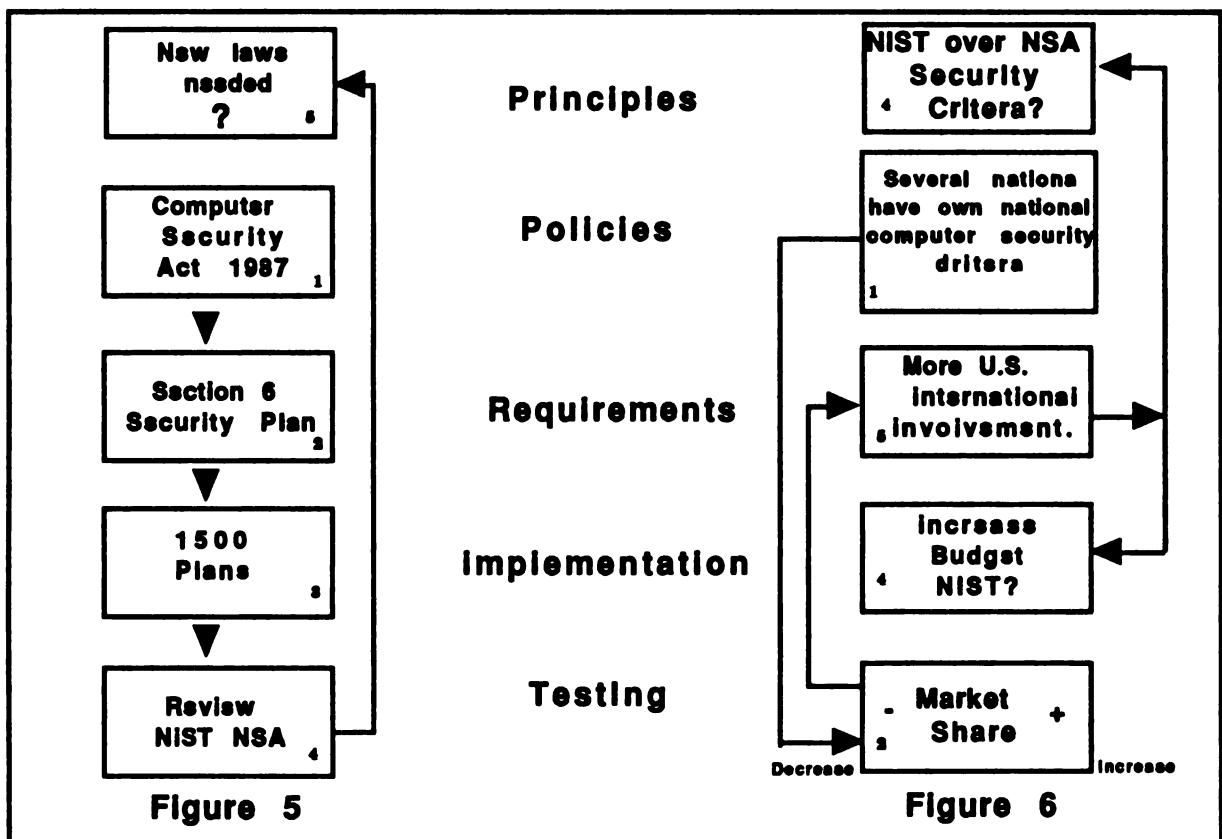
At the 12 th. annual conference the flag had been raised in the United States on the importance of ethical control to the computer security problem. Only time will tell if it has been raised high enough for the individuals in the computer security community to take notices and bring the ethical control subsystem into stability.

Legal/Political Layer

The computer security Act of 1987 is beginning to have some effect on the computer security situation and many of the speakers referred to the act in their presentations. Of particular importance are section 5 and section 6 of the Act. Section 5 of the act requires that federal agencies, processing classified and above material, have mandatory periodic training on computer security for their personal and section 6 requires that federal agencies develop a computer security plan.

The legal/ political subsystem is clear in the implementation and testing state (Figure 5). The security plans of over 15000 federal agencies have been and are being reviewed by the computer security section of NSA and NIST. It will be interesting to see what will be the result of the reviews.

It is difficult to say how long this control subsystem will be staying in the testing state. In the legal/ political subsystem the state cycle shift is influenced to a great extent by the news media and popular opinion. If computer viruses and worms no longer make headlines there is a good chance that the politicians will not introduce new laws.



There was a great deal of complaining among the computer security vendors attending the conference. The computer security Act has created a great deal of interest in computer security in the United States but not necessary a great deal of demand for computer security products. Computer security vendors are not sure that the C2 in 1992 bandwagon will be large enough to carry all of them. The United States vendors want and need to be able to market their security products and knowledge outside the U.S. markets. This message was also emphasized by a congressman's keynote speech which urged for greater internationalization of the computer security effort.

Figure 6 is a SBC flow diagram of the possible future political situation (legal situation excluded). In step 1 several nations create their own computer security criteria. This creates concern among U.S. vendors over possible loss of market share in the international arena (Step 2). For example, vendors based in Britain will dominate the British market, French based vendors will dominate the French market etc. This has lead to market requiring that the U.S. have more international involvement in computer security criteria development (Step 3). The figure suggests that then next step, step 4, will bring about a change in principles and implementation in the subsystem. That is, the general principle that United States National Institute of Standards and Technology rather than the United States National Security Agency will play a more active role in defining national/international computer security standards. As to whether the political subysytem will stay in this configuration will depend on to what extent the United States computer security vendors have succeeded in maintaining and penetrating international markets with their computer security products and knowledge or inversely to what extent internationalization of security standards will enable non United States vendors to penetrate the U.S. markets.

In summary it could be said that the information obtained at the conference when presented in the SBC framework suggests that the legal and political control subsystems for computer security in the United States is currently very dynamic and stable. The concept of dynamic stability within the SBC analysis methodology means that there is sufficient material and information flow for the system to maintain simultaneously all life cycle states from principles to testing. It will be interesting to see how internationalization of the computer security effort will effect this stability.

Administrative Management Layer

The papers presented in this conference track clearly indicates that this control subsystem is in the implementation and testing state. Federal agencies in the United States have a long list of requirements that they must fulfil. The requirements can be found in such documents as the Orange book and the National Bureau of Standard's , (now NIST's), Guide-lines for Computer Security Certification and Accreditation.

The tone of the papers presented in this track are not radical in the sense that there are no statements indicating that the Orange Book and the rainbow series are sometimes difficult to use in the administration and management of a secure computer system. In general the attitudes appear to be that everyone is aware of the short comings of the Orange Book and rainbow series for administration and management of security but it is better to use these models until something better comes along. One of the major short comings of the Orange Book and Rainbow

series that received much attention is that they did not deal explicitly with the problem of software development [1,13,15]. Figure 7 is a SBC block diagram of the papers presented in the administrative and managerial track. In the block diagram the size of the block is proportional to the number of papers that discussed some aspect of that subsystem dynamic class state. For example 7 of the 11 papers in this track presented experiences from the implementation of management models and thus the implementation block is roughly 65% of the total block.

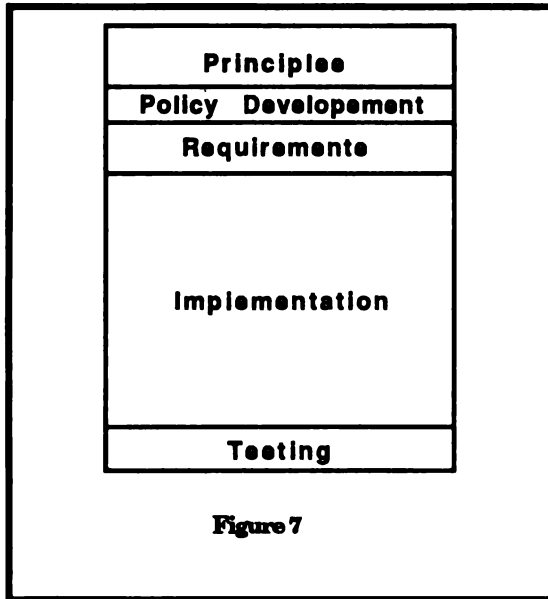


Figure 7

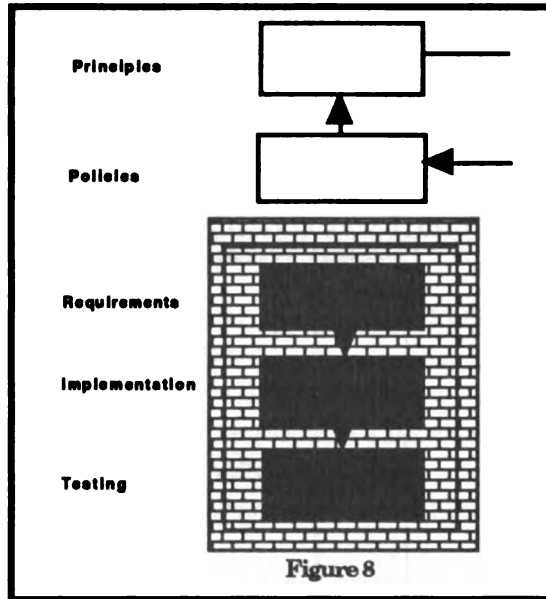


Figure 8

Operational Layer

The conference did not have a specific track dealing with operational practises. But there were a number of papers in the "Systems" track which dealt with some very practical issues of computer security. William Neugent's paper on "Guidelines for Specifying Security Guides [12]" and M.H. Brothers' paper "A How to Guide for Computer Virus Protection in Ms Dos"[2] are good examples. Most of the papers that dealt with security from an operational perspective are of the cook book approach. That is to say they list recipes for secure operational practises. It is interesting to note that of those papers that dealt with the operational problem most of their cited references are from the dates 1986 to 1988. What this shows, in a SBC analysis, is that the operational subsystem in the United States is in a dynamic state. A quotation from Neugent's paper seems however to indicate that even though the system is dynamic it does not seem to be operating properly.

At least four major efforts to produce guards for the military have failed, in the sense that the guards were not used operationally [12].

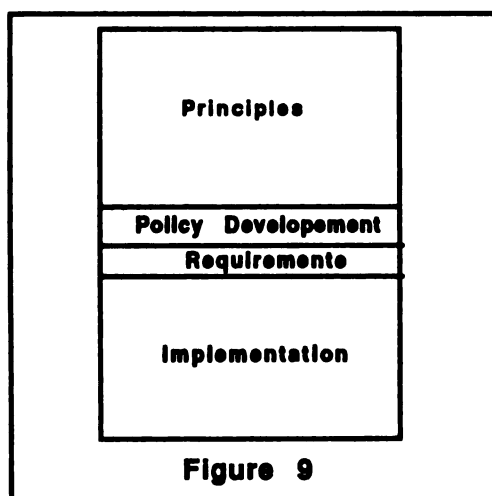
What this would indicate in a SBC framework analysis is that the principles and policies are poorly coupled to the requirement and implementation subsystems (Figure 8 above). A weak coupling between principles and policies, and requirements and implementation, means that what is written on paper at headquarter or in the head office is not practised in the field.

Technical Layer

The conference and the entire computer security community in the USA is basically oriented to the technical problems and aspects of computer security. To cite from the first page of the conference proceedings:

The dominant theme in the literature appears to be entirely dependent on technology to provide security. But long before there were computers we had management controls, principles of good system designing, and procedural security [14].

It is beyond the scope of this summary to do a complete SBC analysis of the technological nature of the papers presented at the conference. A rough analysis indicates that of the 17 papers presented in the "Research and Development" track, 8 were classified as belonging to the principle state, 7 were classified as belonging to the implementation/testing state, 1 was classified as belonging to the requirement state, and 1 was classified as belonging to the policy state (Figure 9).



The emphasis in the technical papers presented at the conference was on problems in database security, communication security and distributed systems security. It appears that the United States computer security community believes that they have good technical bases in operating system security and are starting to attack other technical problems in security from this base. Many papers contain expressions that can be paraphrased as "given a Trusted Computing Base this security function can be developed" [5,17]. Trusted Computing Base (TCB) is Orange Book terminology for secure operating system. This rather primitive SBC analysis seems to indicate that the technical subsystem in the United States is relatively stable. That is to say that activities are going on in all states from principles to testing. The United States has the technical expertise and well developed research infrastructure to deal with computer security as a technical problem.

Conclusion

In this paper the SBC analysis methodology has been used to classify and analyze over six hundred pages of documentation, approximately 24 hours of presentation and approximately 24 hours of informal discussions. The objective for the classification and analysis of such a large quantity of data was to see if any major trends within the U.S. computer security community could be observed.

The SBC analysis of the 1989 conferences indicates three major shifts within the United States computer security community. These are;

- 1) A great acceptance/awareness of the importance of non technical solutions to the computer security problem.
- 2) A shift away from the military computer security perspective.
- 3) A great acceptance/awareness of the need for international cooperation in dealing with the problem of computer security.

References

- [1] Benzel Vickers, Integrating Security Requirements and Software Development Standards, Proceedings 12th National Computer Security Conference, 1989.
- [2] Brothers, M. H., A "How to" Guide for Computer Virus Protections in MS-DOS, Proceedings 12th National Computer Security Conference, 1989.
- [3] Bloombeck Buck J, Trends in Computer Abuse/Misuse, Proceedings 12th National Computer Security Conference, 1989.
- [4] Carroll , John M., Conference Report, Computers & Security ,Vol 9 , 45-49, 1990
- [5] Danner, Bonnie P, Initial Approach for a TRW Secure Communications Processor, Proceedings 12th National Computer Security Conference, 1989.
- [6] DeMaio, H., Informaton Ethics, A Practical Approach, Proceedings 12th National Computer Security Conference, 1989
- [7] Denning ,Dorthy, et al , Social Aspects of Computer Security, Proceedings 10th National Computer Security Conference 1987.
- [8] Kowalski S., Computer Ethics and Computer Abuse: A longitudinal Study of Swedish University Students, IFIPS TC11 Conference , May 1990.Elsevier Science.
- [9] Kowalski S.,Creating Confidence Through Consensus:Using the SBC Model as Framework for Security In Open Systems, IFIPS TC11 Conference ,May 1991.Elsevier Science.o
- [10] Kowalski S., Cybernetic Analysis of National Computer Security, Computers & Security, Vol 10, No 3.,1991, Elsevier.
- [11] Miller James E., Computer Abuse: An Academic Perspective,Proceedings 12th National Computer Security Conference, 1989.
- [12] Neugent, William, Guidelines for Specifying Security Guards, Proceedings 12th National Computer Security Conference, 1989.
- [13] Novell, William, Integration of Security into the Acquisition Life Cycle, Proceedings 12th National Computer Security Conference, 1989.
- [14] Smith,Gary,W, Going Beyond Technology to Meet the Challenges of Multilevel Database Security, Proceedings 12th National Computer Security Conference, 1989.
- [15] Theofanos, Mary, A Systematic Approach to Software Security Evaluations, Proceedings 12th National Computer Security Conference, 1989.
- [16] Wong Raymond et al The SDOS System: A Secure Distributed Operating System Prototype, Proceedings 12th National Computer Security Conference, 1989
- [17] Yngström, L., et al, Förstudie Grundläggande Informationssäkerhet Modeller, SIIS Report 89-2, DSV, University of Stockholm, Sweden

STANDARDIZED CERTIFICATION

Captain Charles R. Pierce

**Air Force Cryptologic Support Center (AFCSC/SRM)
San Antonio Texas 78243-5000**

ABSTRACT

The purpose of this paper is to discuss the process of computer security certification. It begins with an overview of the inconsistencies of certification as encountered in the Department of Defense and indeed throughout the Federal government. It discusses part of the variety of certification definitions, a general overview of the certification process, some prevalent problems with the process and some recommendations for possibly alleviating these problems. The paper's intent is to provide a basis for creating certification standards for computer systems developed or acquired by Program Management Offices (PMO) within the US Air Force. It also intends to inform Designated Approving Authorities (DAA) who place those systems into operational environments about what certification should be. By systems, I mean a computer component or subsystem, either that typically understood to be an Automated Information System or one embedded as part of a larger system. This does not include larger, more widely dispersed computer entities commonly called networks. In some cases the system will stand by itself, such as a stand-alone mainframe system. In others, it will be part of a larger complex system, such as the Advanced Tactical Fighter (ATF) Information Management System, which is part of the overall ATF Weapon System. This paper focuses on the stand-alone or sublevel component system. Policy, standards and implementation guidance for more complex systems is still under development and it is not yet feasible to provide definitive guidance in that arena. That does not mean that the principles in this paper do not apply if the situation warrants.

WHAT IS CERTIFICATION?

Certification and accreditation are parts of a process that leads to the secure implementation and operation of a computer system in a specific environment. Certification is usually understood to be a technically oriented process while accreditation is a management function. Certification has as many definitions as there are agencies that issue certification guidance. Perhaps the most often used, and involving the broadest audience, is that in DOD Directive 5200.28:

"The technical evaluation of an AIS's security features and other safeguards, made in support of the accreditation process, which establishes the extent that a particular design and implementation meet a set of specified security requirements." [1]

The military services and the National Computer Security Center (NCSC) have their own somewhat different, yet similar, definitions [2,3] as do other DOD agencies and the government's civil sector, the primary one being the Department of Commerce's National Institute of Standards and Technology (NIST). Most definitions agree that the system is to be evaluated in some manner as to how well its security measures meet a set of security requirements or specifications. They usually agree that certification supports accreditation.

In reality, discussions involving system developers and security experts on certification usually center on how well technical security measures, particularly those in operating systems, have been implemented to meet stated requirements. Other areas requiring certification, such as facilities and applications software, are not usually discussed during development unless the system's users become involved or they are a part of the development.

Although there are also language differences in the definitions for accreditation, there is consistent agreement that it is a management process that uses certification results and risk acceptance to issue the approval to operate a system in a designated environment. [2,3] The problem, and resulting corrective goal, is the implementation of a certification process that accurately evaluates a system's security posture and provides reasonable assurances that security is sufficient to its accreditor, the Designated Approving Authority (DAA).

CERTIFICATION FLOW

The general flow from the beginning of a system's development to accreditation is not standard but somewhat parallels the following typical steps.

Requirements

A user who needs the system defines security requirements based on an analysis of mission capabilities and shortfalls in these capabilities. If security is a critical issue, that information and security's possible effects on mission performance is included in the system's Mission Need Statement (MNS). Risk analysis (sometimes called threat analysis) is used to define threats and vulnerabilities to the system. The risk analysis is initiated early in the system's life cycle and continues throughout the system's existence. Risk analysis is the first phase of an overall risk management program, with certification and accreditation being the other phases. [2] The requiring user will use the downward directed requirements from existing policy (e.g., DODD 5200.28 [1], DOD 5200.28-STD [4], and AFR 205-16 (AFR 56-30 and AFR 56-31 [5])) and other mission particular operational requirements for initiating the risk analysis process. The output of this first round of risk analysis should be a set of security requirements, including the proposed Trusted Computing Base (TCB) level.

Mission Need Statement Preparation

The MNS preparation, review, and validation can include defining deficiencies in mission performance, including those due partially to computer security and existing hardware and software provisions. The MNS states acceptable performance of mission tasks and functions, but does not state specific hardware and software solutions. The system developer's review of the user's MNS will include, as much as possible, an assessment of program technical risks, including those induced by security. Finally, the MNS will be evaluated and validated if existing systems or improving an existing system cannot meet the user's requirement.

Policy Definition

The major intent of the requirements analysis, needs definition and first round of risk analysis is to produce a security policy at the system level. The security policy will be the "guiding light" or road map for all subsequent development actions. This system security policy should not be confused with the security policy described in DOD 5200.28-STD. That policy is explicitly for the TCB, not the entire system. The system policy will describe requirements for computer security, communications security, physical security, etc.

Security Development

Countermeasures are implemented during the system's development life cycle to meet the requirements generated by risk analysis results, just as other features meet other requirements. The various reviews and audits described in MIL-STD-1521B [6] include security much as they would any other operational issues. As a minimum, the risk analysis should be reviewed or reperformed at each formal development milestone. Any identified deficiencies or shortcomings must be evaluated for their impact on the intended security implementation. Serious problems could cause changes affecting proposed operating modes, TCB levels, or other security requirements and residual risks. In the later phases of development, various tests and evaluations determine if countermeasures meet requirements. The system developer, usually a contractor, performs Developmental Test and Evaluation (DT&E) as the system is being built to determine if the design has been properly implemented. DT&E normally looks at the system from a somewhat isolated, technical view and minimally considers the operating environment. Some organization independent of the system developer, contractor, or the user performs Operational Test and Evaluation (OT&E) to evaluate the system in its operating environment. OT&E considers active users and the system's final facility. Security Test and Evaluation (ST&E) is defined as an examination of the system's security measures. [2] Since both DT&E and OT&E can also evaluate security, there may be significant overlap between them and ST&E. ST&E's requirements may be completely included in DT&E and OT&E or it may be performed solely as a stand-alone process. In any case, ST&E is the final step in the risk analysis process and therefore the last opportunity before certification to identify residual risks.

Certification

After the system is developed, when it is installed at the users facility, and after the risk analysis is completed, the program manager certifies the system. If there is no program manager, as when a system is bought off the shelf, the purchaser has certification responsibility. In any case, whoever delivers the product to the user is usually the certifier. The completed risk analysis is the primary input for the certification package. The risk analysis package may actually be composed of multiple risk analyses, the primary one being that discussed above, i.e., for the system. Other analyses may examine the operational facility, various applications software, collocated equipment (such as secure network gateways), or other interfaced systems. The level of detail, or amount of information in the package, must be agreed upon by the certifier and the DAA.

Accreditation

The DAA takes the certification package and any other recommendations into consideration with the operating environment and makes an accreditation decision. The DAA may approve final operations as recommended, a more restrictive mode of operation, or disapprove operations until residual risks are reduced. The DAA may also provide an interim approval to operate the system as is with risk reducing measures required within a set time period. Recertification and reaccreditation are required on fixed schedule, e.g., three years, upon system modifications, or when security deficiencies are discovered.

Existing systems which did not go through the entire risk management process use as much of it as needed to reach certification and accreditation. Since they do not go through the development process, the early stages of risk management do not easily apply. The resulting accreditation decision usually involves more acceptance of risks.

CURRENT PROBLEMS

Terminology

Although similar certification definitions are provided in multiple policy directives, their process application is inconsistent. Many people confuse a NCSC term, "evaluation," as meaning certification. In fact, NCSC sometimes uses "certification" when they speak of evaluation. Specifically, DOD 5200.28-STD states that evaluations can be delineated into two types: (a) an evaluation can be performed on a computer product from a perspective that excludes the application environment; or (b) it can be done to assess whether appropriate security measures have been taken to permit the system to be used operationally in a specific environment. The first type of evaluation is that done by NCSC on a commercial product. The second type, done to assess a system's security attributes in a specific environment, is known as a certification evaluation [4]. Some civilian organizations use "certification" when they mean "accreditation." This is a minor point because there are many pressing problems that need attention more than developing a standard definition. These problems continue to occur regardless of who's definition is used and would not likely be solved by a common definition. Nevertheless, definition commonality would be an improvement on the road to any standards.

Responsibilities

Certification responsibilities are not firmly defined. Whatever agency or individual is responsible for certification can depend on such variables as the particular type of system being developed or the type of agency doing the development. Multiple versions of a "standard" system consisting of primarily off-the-shelf software and hardware are usually certified by a "Standard System Manager." Systems composed of developmental software or hardware usually have Program Managers who certify what they develop against its stated requirements. Responsibility problems are even more compounded when a developmental system is partially built of off-the-shelf items. Systems built of components acquired by multiple developers can have an equal number of certifiers who may feed a wide variety of inconsistent certification products to a single DAA.

Responsibility Transfer

Currently, developmental system certification responsibility resides with the system acquisition or development agency until the system management transfer (SMT) to the system's supporting agency or user. The supporting agency's (e.g., Air Force Logistics Command) life cycle responsibilities are usually not clearly understood and are seldom implemented properly or consistently. Where standard acquisition or development guidance is lacking, supporting or maintenance guidance is almost nonexistent. Agencies, such as the original evaluators, cannot be tasked to maintain those evaluations as parts of certification throughout the life cycle. For example, once NCSC completes a TCB evaluation its agreement for maintaining that evaluation is with the system's commercial developer, not its purchaser or user. If economically feasible, and the TCB level is low enough, a commercial developer can enter NCSC's Rating Maintenance Phase (RAMP) [7] to ensure a product's TCB rating is maintained when system changes are made. The RAMP is a relatively new program with little experience and bears monitoring. It also does not address higher level TCBs, i.e., B2 and higher. In any case, reevaluations will probable not coincide with recertification schedules, in fact it is unlikely the user system will be reevaluated, but instead replaced.

Certification Sharing

Resources for maintaining central certification activities, such as the Air Force Cryptologic Support Center (AFCSC), NCSC or NIST are not available. Therefore no central repository of certification lessons learned or experience information exists. Nor is there a central pool of certification skills. NCSC does maintains a pool of evaluators and AFCSC, for example, is developing this capacity at its Product Assessment and Certification Center (PACC). A program management office must form and educate a new certification team for each system. The education process will probably not be able to benefit from any other system's lessons learned, partially because of this lack of central sharing. Broad level mission or organizational realignments to free resources to provide a central certification capability seems unlikely in the near future. Required capabilities range from the current advice and assistance provided by the Air Force, to expertise centers or central certifying agencies and information repositories. Currently provided advice and assistance is much like that provided by an Independent Verification and Validation (IV&V) agent or contractor.

Unclear Requirements

A normal system certification is typically based on a subset of the potential users' requirements. If users were surveyed before development began, it is unlikely that all potential users provided requirements. A system originally planned for a single user, such as the Air Force Space Command, may have addition users identified, such as the Strategic Air Command, while it is still under development. Any number of new factors, e.g., access controls, could induce stricter requirements than those against which the system is design to be certified. Unclear requirements are difficult to combat when the system is acquired or developed under a "requirements contract." If the available "requirements contract" system meets a user's stated requirements, the user must purchase that system and not one specifically tailored to his needs. The certifier for this type of system is faced with gathering a

complete set of requirements that could possibly satisfy all potential users. This is neither practical or reasonably possible. Invariably the "standard" certification for the available system will leave residual risks for some users. DAAs will now be faced with undesirable risk acceptance, acquiring additional security measures or justifying the system's non-suitability based on security deficiencies. On the other end of the scale some users will have excess security measures to implement, e.g., security label management, even though the measure exceed their requirements.

Excessive Security

Should a system developer try to implement a complete set of security measures to meet all possible environments or user requirements the cost would be prohibitive and seldom justifiable. In most cases where more than one sensitivity level of information is to be run, a requirements analysis would probable indicate the need for a B2, B3 or A1 TCB. [4] Properly applied risk analysis principles can prevent over-specification, but only if valid requirements are available. The DAA must decide on the most cost-efficient implementation of security measures. There is no guidance available on how to obtain equivalent levels of security by trading administrative or procedural security measures for technical security features. Additionally there is little or no life cycle cost experience or guidance available for implementing trusted features.

Embedded Systems

One major shortcoming of current certification methodologies is that they do not apply well to embedded weapon systems. Certifications are not often done for embedded systems or else they typically overlook embedded components of larger systems. Standard criteria like those for TCBs [4], do not exist for embedded systems. Many of the features or assurances of the TCB classes in DOD 5200.28-STD are not relevant to embedded systems. For example, there is little need for an audit trail feature on a tactical missile with embedded processors. Besides, who would analyze it after the missile is fired?

DAA Capability

Many senior accreditors (DAA) do not have sufficient knowledge or capabilities to make credible approval decisions. They seldom have been involved in computer security during their careers. In the DOD environment they typically spend the major portion of their careers in the prime area of operations for their parent service, e.g., ship captains. Even if they have participated in system development or operations, it has not been with secure systems or from a security point of view. Security guidelines for the DAA have not been completed.

Integrity and Service Assurance

Current certification methodologies do not adequately address the critical issues of service assurance and data or system integrity. This is natural because there is little available policy in this area. There are also no standards such as DOD 5200.28-STD or DOD-STD-2167A [8] that address security relevant criticality in systems or software development. The Trusted Network Interpretation of DOD 5200.28-STD [9] does minimally address network integrity

Responsibility Transfer

Currently, developmental system certification responsibility resides with the system acquisition or development agency until the system management transfer (SMT) to the system's supporting agency or user. The supporting agency's (e.g., Air Force Logistics Command) life cycle responsibilities are usually not clearly understood and are seldom implemented properly or consistently. Where standard acquisition or development guidance is lacking, supporting or maintenance guidance is almost nonexistent. Agencies, such as the original evaluators, cannot be tasked to maintain those evaluations as parts of certification throughout the life cycle. For example, once NCSC completes a TCB evaluation its agreement for maintaining that evaluation is with the system's commercial developer, not its purchaser or user. If economically feasible, and the TCB level is low enough, a commercial developer can enter NCSC's Rating Maintenance Phase (RAMP) [7] to ensure a product's TCB rating is maintained when system changes are made. The RAMP is a relatively new program with little experience and bears monitoring. It also does not address higher level TCBs, i.e., B2 and higher. In any case, reevaluations will probable not coincide with recertification schedules, in fact it is unlikely the user system will be reevaluated, but instead replaced.

Certification Sharing

Resources for maintaining central certification activities, such as the Air Force Cryptologic Support Center (AFCSC), NCSC or NIST are not available. Therefore no central repository of certification lessons learned or experience information exists. Nor is there a central pool of certification skills. NCSC does maintain a pool of evaluators and AFCSC, for example, is developing this capacity at its Product Assessment and Certification Center (PACC). A program management office must form and educate a new certification team for each system. The education process will probably not be able to benefit from any other system's lessons learned, partially because of this lack of central sharing. Broad level mission or organizational realignments to free resources to provide a central certification capability seems unlikely in the near future. Required capabilities range from the current advice and assistance provided by the Air Force, to expertise centers or central certifying agencies and information repositories. Currently provided advice and assistance is much like that provided by an Independent Verification and Validation (IV&V) agent or contractor.

Unclear Requirements

A normal system certification is typically based on a subset of the potential users' requirements. If users were surveyed before development began, it is unlikely that all potential users provided requirements. A system originally planned for a single user, such as the Air Force Space Command, may have addition users identified, such as the Strategic Air Command, still under development. Any number of new factors, e.g., acc could induce stricter requirements than those against which design to be certified. Unclear requirements are difficult. A system is acquired or developed under a "requirements available "requirements contract" system meets a user's the user must purchase that system and not one speci needs. The certifier for this type of system is

complete set of
This is the
certification of
users. DAA
additional
on security
excess
though

Excessive

Should a
measures
be prohibitive
sensitive
probable
analysis
requirements
implementation
obtain equivalent
security measures
little or no
trusted features

Embedded Systems

One major
do not apply
done for
of larger system
embedded system
5200.28-STI
need for
Besides, when

DAA Capabilities

Many senior
capabilities
involved in
they typically
operations for
participated
systems or
not been

Integrity

Issues of
because
standard
related
Integrity

d
.
e
h
e
as
on
s-
ide
of
not

be
ere
s of
both,
such

ch as

agency
Force
other
ficult
valence

d, such
ntation
uments
r cases
lication

cribe the

and service assurance, but does not provide firm criteria such as those for stand-alone systems. Some security features are described but their evaluation is based on qualitative estimates of effectiveness. What other applicable guidance there is provides for safety issues, typically nuclear or medical safety.

EXISTING REMEDIES?

Standard Certification Process

It may be advantageous to produce a "standard" certification process. This process would contain ALL potential stages and actions required for the most complex SYSTEM certification. An all inclusive process would be very large and applicable in whole to only the most complex system. By necessity the process would include tailoring directions, with examples, for various types of systems, e.g., micros, stand-alones, networked, embedded, complex combinations, etc. Not all steps in a tailored process would necessitate following a "standard" step, thus there must allowances for unique variations. When used, risk analysis methodologies could vary. For example, a package such as the Automated Risk Evaluation System (ARES) could be used for a one time run for a small or large scale system. A multi-disciplined approach could be used for a complex system, particularly if developmental components mix with those off the shelf. Other unforeseen elements could also have major impacts.

Life Cycle Guidance

The Air Force is producing a complete set of life cycle oriented guidance as Air Force System Security Instructions and Memoranda (AFSSIs and AFSSMs). These provide guidance leading to system certification and accreditation plus other services. Most are currently under development. The topics involved include Security Policy Generation [10], acquisition guidance [11], Source Selection guidance [12], ST&E [13], applications software development [14], DAA Guide [15] and more. NCSC is also providing life cycle guidance applicable to the certification process, such as its procurements guide. [16]

Certification Consulting

The Air Force also provides life cycle oriented consultation services to both program managers and standard off-the-shelf system managers. These services include: defining security requirements, as derived from mission and downward directed requirements; developing security specifications, including those for TCBs, certification and accreditation supporting documents; providing Contract Data Requirements Lists (CDRL) and Data Item Descriptions (DID) for security deliverables; ST&E assistance; and operational guidance. The level of involvement in a particular program varies from telephone and correspondence to nearly full time "hand holding," if the security implications of the project merit it.

Certification Analysis

The Air Force's Product Assessment and Certification Center (PACC) evaluates computer security products for their applicability to Air Force acquired systems. The PACC's assessments are not the equivalent of NCSC's TCB evaluations, but intend to determine if products work as advertised on Air

Force systems, currently small systems such as the Z-248 microcomputer. These assessments are useful for determining if an available product can help meet a certification requirement or reduce a residual risk. They are not centralized "certifications" but can be referenced or included in a certification package much as can be a NCSC evaluated product report. Reviews of product assessments are published in the Air Force Assessed Products List (APL), not to be confused with NCSC's Evaluated Products List (EPL).

RECOMMENDATIONS

A Standard Certification Process

Because certification is a complicated endeavor, an initial action should be to formulate a strategy for developing a standard certification process. The strategy should include plans for developing the process, acquiring the resources to implement it, education and training for personnel with certification responsibilities, proficiency standards for some of those personnel, and operational implementation guidance. An effort in this vein was recently begun under the auspices of the Computer Security Implementation Management Panel (CIMP) of the Joint Commanders Group for Communications-Electronics, a Joint Logistics Commanders subgroup.

The process should encompass all types of systems. It must include variations and subsets for small, embedded, and other "unique," etc., types of systems. If these are not included, users may decide the process does not apply to them.

The process should also highlight when user or developer decisions must be made as to strictly applying the process or that a point has been reached where a user risk assumption decision must be made. Embedded and like types of systems must be considered.

It must clearly define who each step applies to (user, developer, both, etc.). The process should contain some specifics not currently addressed, such as logistics and maintenance of trusted software.

The process must describe where within itself various guidance such as AFSSIs and AFSSMs or NCSC Technical Guidelines is to be applied and how.

It must describe how, why, when, and where to use and accept other-agency evaluations and certifications, such as NCSC evaluations, Air Force assessments, or certifications from other military components done under other regulations, e.g., Army Regulation 380-19. This may be quite difficult considering there are also no standards for measuring certification equivalence between agencies.

The process must also define when such outside products are required, such as NCSC evaluated TCBs. Also included would be advice as to what documentation to use or require from those other evaluations. The commercial documents produced as part of a NCSC TCB evaluation may be sufficient. In other cases system unique documents may be required, particularly if dedicated application software is involved.

Although it appears obvious on the surface, the process must describe the

applicability of DOD documents, service regulations and technical guidelines. This is particularly critical if the system development is contracted. If a document's applicability is not stated specifically in a contract it normally will not be legally enforceable. It will be impractical for all documents to apply all the time. For example, a network guide would be impractical for a stand-alone system.

The resources for implementing the process must come from those currently available. It is unlikely that any new personnel or any increased funding will become available to formalize what we are already suppose to be doing. Current security people must become more expert in the non-computer security disciplines and more customer service oriented. The inability to be fully DOD customer oriented has become a perceived failing of the NCSC. The process must not be so complicated that these existing resources will be overloaded to the point of duplicating this perception.

The process must be mandated as a standard for all systems. Include this mandate in agency policy and regulations. Include the process in certification and accreditation management and technical guidelines. [17] Each program, standard system, etc., must include a system-tailored process description in their security plan. The plan must define roles and responsibilities, including those in other organizations such as AFCSC. It must also include rules for including external agency evaluations.

Other guidance must be completed. This would include the applicability or inclusion of other agency (e.g., NCSC, AFCSC, etc.) evaluations as supporting certification documentation, the acceptability of EPL or APL reports without further testing, and responsibilities for recurrent life cycle reviews or recertifications. Standard criteria for evaluations beyond DOD 5200.28-STD are desperately needed. This involves embedded systems, complex systems, real time systems, the Trusted Database Interpretation (TDI) [18], the Trusted Network Interpretation (TNI) [9], and applications software. A subsequent activity is to complete translation of these criteria into acquisition specifications formats and operational implementation guidance.

As service organizations, the services and comparable agencies must improve their capabilities to provide consultive support to both existing systems and those under development or acquisition. A first step must be to develop a program to educate all applicable personnel in multiple security disciplines, i.e., COMSEC, TEMPEST, etc. A NIST or NCSC personnel certification program could be possibility, or perhaps one developed by industry or the educational community.

Finally, if possible, the process should include as many of the recommendations of the National Research Council's "Computers at Risk" [19] panel as practical. Not all of the panel's recommendations can or will be implemented in the DOD environment. However, we must seek the maximum commonality between the DOD as commercial communities, if for no other reason than cost savings. The certification standard should be flexible enough to do this.

REFERENCES

1. Department of Defense Directive 5200.28, Security Requirements for Automated Information Systems (AIS), 21 March 1988.
2. AFR 205-16, Computer Security Policy, 28 April 1989 (To become AFR 56-30, same title).
3. NCSC-TG-004, Glossary of Computer Terms, 21 October 1988.
4. Department of Defense Trusted Computer System Evaluation Criteria, 26 December 1985.
5. AFR 56-31, Security Policy and Requirements in the Development and Acquisition of Computer Systems, (Draft), 8 January 1991.
6. MIL-STD-1521B, Technical Reviews and Audits for Systems, Equipments, and Computer Software, 4 June 1985.
7. NCSC-TG-013, Ratings Maintenance Phase Program Document, 23 June 1990.
8. DOD-STD-2167A, Defense System Software Development, 29 February 1988.
9. NCSC-TG-005, Trusted Network Interpretation of Trusted Computer System Evaluation Criteria, 31 July 1987.
10. AFSSM 5001, System Security Policy Generation Guide, (Draft) 21 December 90.
11. AFSSM 5024, Computer Security Considerations in the Acquisition of Computer Systems, (Draft), 10 June 1990.
12. AFSSM 5002, System Selection Technical Evaluation, (Draft).
13. AFSSM 5025, Security Test and Evaluation (ST&E) Guide, (Draft) 2 November 1990.
14. AFSSM 5011, Computer Security in Software Development, (Draft), 23 Jan 1991.
15. AFSSM 5003, DAA Guide, (Draft), 31 October 1989.
16. Using the Department of Defense Trusted Computer System Evaluation Criteria in DOD Procurement, (Draft), NCSC, 13 May 1991.
17. AFSSI 5026, Certification and Accreditation Guide, (Draft).
18. NCSC-TG-21, Trusted Database Management System Interpretation of Trusted Computer System Evaluation Criteria, 22 August 1990.
19. National Research Council, Computers at Risk: Safe Computing in the Information Age, Computer Science and Technology Board, National Academy Press, Washington, DC, 1991

A STRATEGIC FRAMEWORK FOR INFORMATION SECURITY MANAGEMENT

**Rolf Moulton, CDP, CISA, CSP
Senior Regional Information Security Representative
BP America
200 Public Square, Suite 6-K
Cleveland, OH 44114**

**Santosh Misra, DBA
Associate Professor
Computer and Information Science Department
Cleveland State University
2400 Euclid Avenue
Cleveland, Ohio 44114**

Abstract

This paper proposes a reference framework to help improve the effectiveness of information security management. The framework is intended as a standardized vehicle that can be used by both business and security managements to identify and prioritize information security requirements on the basis of the intended use of the information, the value priority of the information, and the critical security factors for that information. The paper also suggests that a serendipitous benefit resulting from the use of the proposed framework could be a better understanding of the present and potential use of an organization's information assets for competitive advantage.

Keywords

CRITICAL SECURITY FACTORS, INFORMATION SECURITY MANAGEMENT, INTENDED USE OF INFORMATION, RISK MANAGEMENT, SECURITY PLANNING, VALUE PRIORITY OF INFORMATION

Acknowledgement

The authors would like to express their appreciation and thanks to Dr. Bruce Baker, Robert Courtney and Donn Parker for their comments and suggestions during the preparation of this paper.

A STRATEGIC FRAMEWORK FOR INFORMATION SECURITY MANAGEMENT

BY

ROLF MOULTON, CDP, CISA, CSP

SANTOSH MISRA, DBA

1. INTRODUCTION

There is no standard definition of information security, nor are there generally accepted criteria for measuring or prioritizing information security requirements. There is a wide variance in understanding security priorities, vulnerabilities, threats and safeguards among information users, providers and regulators. [4,6,19] And, there are significant differences in the sources of security concerns expressed by managers in the United States, as well as by managers in other countries.[26] Consequently, an organization's management may have considerable difficulty in its efforts to define security requirements and to prioritize resource allocations for security.

Information value has been a major factor for developing priorities as part of some security management programs. However, defining the value of information for the purpose of setting security priorities continues to be as difficult as defining its value for competitive advantage.[4,27,28] That may be the result of examining information value in too limited a context.

This paper seeks to place information into the broader context of intended use, value and the factors which may have an adverse impact on it. It examines principles associated with information security management (ISM) and proposes a strategic framework for ISM that has three major components. They are:

- . Intended Use of Information (IUI)
- . Value Priority of Information (VPI)
- . Critical Security Factors (CSF)

The remainder of this paper is organized into four sections. Section 2 examines some existing methods of ISM. Section 3 discusses the components of the framework that is proposed in this paper. The paper concludes in Section 4 with suggestions of future research.

SECTION 2.

METHODS OF ISM

The most widely known criteria for managing information systems security may be those defined by the U.S. Department of Defense (DOD) in its "Orange Book." [29] The DOD approach emphasizes information confidentiality, but does not stress integrity, availability or authenticity of information, all of which are significant for business users.[3] Many security professionals, especially those in Europe, also find the security criteria specified in the Orange Book lacking in terms of the needs of a networked society [26]. Consequently, several countries and organizations, individually and collectively, have begun efforts to harmonize their criteria for information security as a means to define and prioritize their information security requirements.[8]

Some security professionals favor a quantitative risk assessment method(s) to help prioritize information security requirements. Using this method, the value of information loss is quantified as the probable frequency of loss due to adverse action occurrences. [10] This approach appears to be better accepted by government agencies than by private industry.[12]

Establishing information security priorities with qualitative risk assessments is another technique used by security professionals.[21] This method may be used to establish baselines of risk and prudence, which in turn lead to the prioritization of security needs. Qualitative techniques do not usually develop a monetary value for information; according to some security professionals, qualitative, relative ranking of information value for security purposes is perhaps quite sufficient for business needs.[23] Variations of both quantitative and qualitative approaches have found favor in some organizations, but concerns have been expressed about the work effort required to obtain results that are meaningful.[5,12]

Assigning a value to information either for determining security priorities or for calculating a return on investment has proved to be difficult.[27,28] Without general agreement on the basis for establishing the value of information, or other measures to be used in place of value, there is little surprise that risk assessment advocates have yet to come to terms with each other, or with the problem of defining security priorities. The U.S. National Institute of Standards and Technology (NIST) has established a forum and procedural mechanism which, it hopes, will lead to a standardization of terminologies and techniques for information management.[2,13,15,18]

SECTION 3. A FRAMEWORK FOR INFORMATION SECURITY MANAGEMENT

It is difficult to assess the 'essentiality' of information to an organization unless the relationship of the information to the organization's functions is clearly understood.[7] Building on this contextual nature of information, a three dimensional framework is proposed for use within ISM. These dimensions respectively provide definition for the location, value and structural context of information security requirements. Each is discussed below.

3.1 IUI

Information may be located within four overlapping general strata that are based on the intended use of the information. The strata are STRATEGIC, TACTICAL, TRANSITORY, and CHATTER.

STRATEGIC information is used by an organization's executive management to develop major business strategies and decisions, such as acquisitions, mergers and new business ventures. Strategic information is likely to be acquired and managed with a great deal of attention to the needs of the executives who use it, rather than on the basis of standard cost/benefit considerations. There may not be a great volume of this information maintained on a regular basis; it may or may not be handled within executive support information systems. Some of this information is likely to be very valuable and would require a high degree of protection, while other strategic information may be in wide public use with relatively low overall protection. Most of the high-risk strategic information and its related protection mechanisms would probably not be subjected to formal accounting controls audits.

TACTICAL information is primarily created and used by the operating and administrative managements of an organization. It may include operational information from various organizational functions, such as sales, finance, production, marketing, research and human resources management; it would also include the backup and archival copies of this information. Tactical information may be provided to executive management in detailed or summarized form as strategic information. The volume of tactical information is likely to be large. Cost/benefit ratios and regulatory compliance would be key considerations in the acquisition and management of tactical information. Tactical information can be expected to comply with financial or other standardized accounting auditing practices.

TRANSITORY information may be considered as information in the processing pipeline. When consolidated and evaluated it may become tactical or strategic information. It may include

information from the multitude of computer programs (including undocumented information bases and spread sheets) that people develop and use to analyze information from internal and external sources. Transitory information may or may not be subject to cost/benefit controls or standardized audits.

CHATTER is the remainder of information that flows through an organization, with or without management's knowledge, consent or control.

3.2 VPI

VPI is defined as a value ranking assigned to information by its owners and users within the context of the intended use of the information. The VPI may be established using a rank order scheme, or it may be a quantitative monetary value, or it may be set using some other system that is based on the requirements and practices of the information owners and users. The VPI is clearly subjective unless the users are able to establish a quantitative real dollar value. The full value priority of its information to an organization can then be considered as a weighted sum of individual VPI values.

VPI, as envisaged in this paper, helps move the current security valuation emphasis beyond tactical information to other categories of information. The VPI concept of information value deviates from classic approaches to information valuation. For example, value of information has been presented using descriptions such as NORMATIVE [14,16], REALISTIC [17,11,9], and SUBJECTIVE [22,25]. While those methods are theoretically interesting, they have limited practical application from a security perspective.

VPI and IUI may vary considerably within an organization. As an example, information that is critical to an organization's executive management may be significantly different from information that is used principally by the organization's operating and staff managements. This use difference may also require a different scale basis (monetary, subjective rank, or other) to define a value priority that is acceptable to the organization. The Information Systems Security Association (ISSA), Newport Beach, California, USA, has initiated a long term study of information valuation that may be extremely helpful in this regard.

VPI can also be used to address opportunity costs and losses associated with information, including information that the organization plans to obtain at some time in the future. This would help to resolve a limitation of risk management strategies that focus on existing information use, but ignore future information opportunities.

3.3 CSF

The third dimension of the proposed strategic framework for ISM examines the structural context of information. This structure of information helps to establish critical CSFs that are needed for effective ISM. It is suggested that the six CSFs be used to help determine the level of information risk. They are modified extensions of Donn Parker's five security attributes,[23] which are expanded to include the factor of TIMELINESS.

The CSFs are defined as follows:

AVAILABILITY is the state of being present, accessible, or obtainable for a specific purpose.

UTILITY is the state of being useful or fit for some purpose. Utility can be lost, yet availability preserved, when information is encrypted and the intended user is not provided with the decryption key.

TIMELINESS of information refers to the state the information at an instant in time. The relevancy of timeliness of information to the utility of information is well established. However, timeliness is isolated as a CSF because of the rapidity with which information may gain or lose its real or potential utility value, and hence become or cease to be a security concern. As an example of extremely rapid transition of utility, a company's confidential quarterly earnings information could be extremely valuable information to a person(s) wishing to invest in that company up to the time at which it is released to the public. Once the information is made public, instantly, it then no longer requires protection from disclosure, modification or availability. (There is some disagreement with the authors' use of timeliness as a CSF.[1])

INTEGRITY of information exists when all information is present and accounted for. It does not represent that the information is correct or is otherwise a true representation of some condition. It is consistent with the ISO (International Standards Organization) communications concept that information is received as sent, with nothing added, deleted or modified.

AUTHENTICITY of information refers to its extrinsic correct or valid representation of that which it is intended to represent. As an example, a program is authentic if its pedigree can be traced back to include the original copy and all changes have been properly authorized. An electronic mail note is authentic if it can be demonstrated that it was sent by the sender, who in turn can not repudiate having sent the note. An inventory quantity is authentic if it accurately represents the actual number of items on hand or available for sale.

CONFIDENTIALITY of information refers to the information being maintained as secret or private to only those permitted to know it or have access to it.

SECTION 4.

CONCLUSION

The goal of effective ISM is simply

.... to lessen either the probability that something undesirable will happen (or the frequency with which it is known to be happening) or the severity of the consequences when it does happen, or both.[7]

Meeting this goal requires considerable knowledge of the assets that are at risk and the undesirable events that may occur to those assets. And, it requires that both the business and security managements of an organization take prioritized actions to achieve a prudent level of comfort with regard to them. Towards meeting this goal, the proposed strategic framework for ISM establishes terms of reference for defining information security requirements in a context that would facilitate the use of varying risk management strategies to help prioritize the allocation of security resources.

There may be a supplemental, perhaps even serendipitous, benefit from the use of the proposed strategic framework for ISM. The framework may be directly applicable to setting priorities for overall information management, as well as possibly deriving supplemental productivity improvements during the process.[20] It addresses both the current and future information requirements and opportunities of the organization. It builds on the critical success factor approach used to identify information needed by chief executive officers to support the attainment of organizational goals [24], and such an approach to both information management and ISM could be developed with future research.

The authors would welcome comments and suggestions towards developing a model to validate the proposed framework. Please send them to:

Rolf Moulton
Senior Regional Information Security Representative
Regional Center Security
BP America
200 Public Square, Suite 6-K
Cleveland, OH 44114

REFERENCES

- [1] Baker, Bruce and Parker, Donn, comments on draft of this paper, 4/8/91
- [2] Browne, Peter, "A Descriptive Risk Management Framework-Draft," Computer Security Risk Management Model Builders Workshop, May, 1989
- [3] Clark, D. & Wilson, D., "A Comparison of Commercial and Military Computer Security Policies," Proceedings of the 1987 IEEE Symposium on Security and Privacy, IEEE Computer Society, 1987
- [4] Competitive Intelligence Gathering: Friend or Foe, Cris R. Castro, SRI International, September 17, 1991
- [5] Computer Security Handbook: Strategies and Techniques for Preventing Data Loss and Theft, Rolf Moulton, Prentice Hall, 1986
- [6] Computers at Risk, National Research Council, National Academy of Sciences, 1991.
- [7] Courtney, Robert, Robert Courtney, Inc, Port Ewen, NY, limited circulation draft on risk assessment, and personal interview on 2/6/91.
- [8] Cutler, K. & Jones, F., Commercial International Security Requirements, Draft 15 January, 1991, I4 Forum #12, January, 1991
- [9] Edstrom, O., Man-Computer Decision Making, Gothenberg Studies in Business Administration, 1973, Gothenborg, Sweden,
- [10] FIPS PUB 65: Guideline for Automatic Data Processing Risk Analysis, National Bureau of Standards, 1979
- [11] Hedberg, B., Man-Computer Decision Making, Gothenberg Studies in Business Administration, 1973, Gothenberg, Sweden
- [12] I4 Forum #10, SRI, International, discussion of risk management tools and strategies, May, 1990, London, UK
- [13] Katzke, Stuart, "A Framework for Computer Security Risk Management," National Institute of Standards, May, 1989 (Computer Security Risk Management Model Builders Workshop)
- [14] Marschak, J., "Economics of Information Systems" Journal of the American Statistical Association, 66, 1971, P192-219

- [15] Mayerfeld, Harold, "Framework for Risk Management," Computer Security Risk Management Model Builders Workshop, May, 1989
- [16] McGuire, C.B. & Radner, R, Decision and Organization, North-Holland, 1972, Amsterdam
- [17] Mock, T.J., "The Evaluation of Alternative Information Structures", PhD Dissertation, University of California, Berkley, 1969
- [18] Mosleh, Ali, "Mapping Between a Risk Management Methodology and the Proposed Conceptual Framework", Computer Security Risk Management Model Builders Workshop, May, 1989
- [19] Moulton, Rolf, "A Survey of User Attitudes and Practices Towards Information Ownership and Protection in an End User Environment", ISPNews, (July/August '91 - tentative)
- [20] Moulton, Rolf, "A By-Product of Effective Security- Improved Organizational Productivity," Computer Security Journal,V5 #1. 1988
- [21] Moulton, Rolf, "Data Security is A Management Responsibility", Computers & Security, vol 3, 1984
- [22] Munro, M.C. & Davis, G.B, "Determining the Manager's Information Needs", Journal of Systems Management 29, #6, 6/78, 34-39
- [23] Parker, Donn B., "Restating the Foundation of Information Security," SRI International, Applied Research Note 10, October,1990 & revised 1/91)
- [24] Rockart, John F, "Chief Executives Define Their Own Data Needs," Harvard Business Review, Mar-Apr 79.
- [25] Ronan, J. & Falk, G, "Accounting Aggregation and the Entrophy Measure: An experimental Approach," The Accounting Review 28, 10/73
- [26] Schwartau, Winn, "Terrorism, Privacy & Standards: Marketing Infosecurity in the New Europe," ISPNews, Jan/Feb, 1991
- [27] Terdoslavich, William, "Payback Puzzler", Computer Systems News, 6/11/90, p14
- [28] Tinsley & Power, "Why IS Should Matter to CEOs."(Datamation, 9/1/90, v36, p85)
- [29] Trusted Computer Systems Evaluation Criteria, Department of Defense, 1985.

A SYSTEM SECURITY ENGINEERING PROCESS

J. D. Welss

AT&T Bell Laboratories
Whippany, New Jersey 07981

ABSTRACT

This paper describes a formal MIL-STD-1785-compliant, tool-supported System Security Engineering (SSE) process that can be used in a variety of government and commercial environments. The objective of SSE is to derive a cost-effective system security architecture and integrate it into the system design process. The security architecture, like other system attributes, must be evaluable and justifiable. AT&T SSE is also designed to provide a well-defined framework for security requirement evaluation and justification.

INTRODUCTION

There are a number of areas that compete for budget dollars in the design of any system. Security, performance, reliability, interoperability, and a full range of other engineering concerns impose requirements that must be addressed from the pool of resources allocated to system design and development. For each of these engineering areas, analyses are required to demonstrate that the resources associated with meeting the requirements are well spent. Analyses generally show the costs and effectiveness of the associated requirements versus their alternatives across the system lifecycle.

Techniques for analysis in some areas of system design are more established than in others. In the area of communications system performance, for example, one can calculate bandwidth required for message communication paths by establishing message attributes (i.e., size and frequency). Candidate networking technologies that support the necessary bandwidths may then be identified and their trade-offs analyzed [1].

In the area of *security*, however, few techniques are available to provide analytical support for requirements. Many current systems base their security requirements on global policies (e.g., the federal government's Orange Book [2]), previous experience in other environments, and/or the advice of knowledgeable security experts. While these requirements may be argued to be effective and economical, such arguments may only be made on an intuitive level. Still other systems do not address security in their designs at all, opting to retroactively apply protections as the systems are broken. Security often becomes an uncontrolled expense in such cases.

The purpose of this paper is to present a uniform process for providing analytical support for system security requirements. The defined process is AT&T's System Security Engineering (SSE) approach. SSE is being applied on a variety of government, and commercial systems. The sections that follow will provide background and an overview of SSE, while subsequent sections will discuss the individual SSE operations. Finally, a description of a prototype tool-set will be provided, and the paper will conclude with a discussion of SSE obstacles.

BACKGROUND

The SSE process was originally designed by AT&T Bell Laboratories for use on the Strategic Defense Initiative (SDI) System Engineering and Integration contract. SSE was established to be a formal implementation of MIL-STD-1785, "System Security Engineering" [3], and has been successfully used to evaluate key SDI subsystem architectural alternatives. SSE is also being applied on current AT&T products and services.

Security Vulnerability Analysis (SVA) is the analytical engine of SSE. SVA has its technical foundations in:

- risk management theory [4],
- structured analysis [5],
- fault tree constructs used in reliability engineering [6], and
- empirical risk formulas widely applied within AT&T [7].

In addition to SVA, SSE consists of an automated toolset and a security requirements integration process.

SSE OVERVIEW

The SSE process is designed to apply finite resources to mitigate those vulnerabilities that represent the greatest risk to the system. Figure 1 illustrates the goal of SSE: to identify security architectures that fall on the curve of optimal reduction of security risks¹ (vulnerabilities) for applied security dollars.

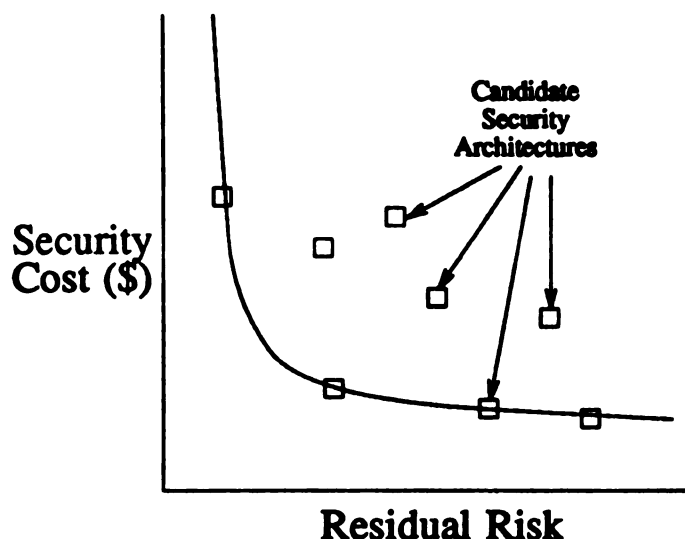


Figure 1. SSE Goal: Security Dollars Spent on Key Security Risks

In order to build the curve in Figure 1, the following ten steps are required:

1. **Baseline Architecture Identification** — The baseline requirements and components of the system to be analyzed must be characterized from a security perspective as the basis for all steps to follow.
2. **Threat Identification** — The assets at risk in the system, and the overall objectives of attacks to which they are subject (threats) must be identified.
3. **Threat Analysis and Decomposition** — High-level threat objectives must then be broken down into intermediate objectives, and, ultimately, into the individual activities that comprise an attack scenario.
4. **Risk Assessment** — Risks must be calculated for the various objectives and activities defined in the previous steps.

1. A system's "residual risk" represents a combination of its individual security risks.

5. **Prioritization of Vulnerabilities** — Areas of vulnerability must be prioritized based on the calculated risks. The key risk drivers for the system must then be selected for application of security resources.
6. **Identification of Candidate Safeguards** — For the selected key vulnerabilities, a set of candidate safeguards must be selected from a variety of disciplines.
7. **Safeguard Trade-off Analysis** — Candidate safeguards must then be assessed against the baseline system architecture for effectiveness and lifecycle costs.
8. **Security Architecture Selection** — Based on the trade-off analysis, an optimal set of safeguards must be identified, along with an assessment of their effectiveness and costs.
9. **Security Architecture Integration** — The selected safeguards must then be integrated into the system design to become part of the new baseline architecture.
10. **Iteration** — The SSE process may be repeated until residual security risks versus dollars spent are within the desired thresholds.

The above steps are supported by the SVA model in Figure 2. The following sections will describe the individual steps of the SSE process and will relate them to the elements of the SVA model.

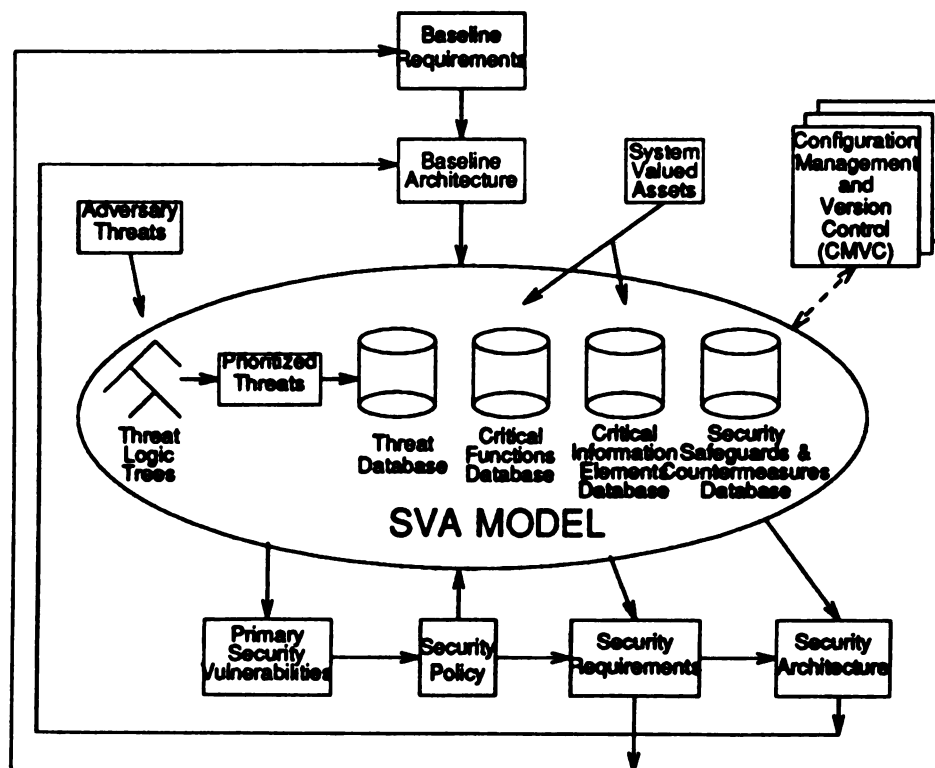


Figure 2. SVA Model

1. Baseline Architecture Identification

Ideally, SSE should commence from the very beginning of a system definition activity. However, more often than not a system architecture baseline already exists that needs to be secured. The system architecture is generally defined from a performance perspective, with system security characteristics (if any) interspersed among other system design details.

The first step in SSE is to identify the elements and components of the baseline system architecture that are security relevant. These elements and components constitute the "system valued assets" in Figure 2. The characterization of system valued assets is used in SSE for identifying and evaluating their existing levels of protection against potential threats, as well as the viability and costs of analyzed safeguards.

System valued assets are represented as critical functions and information elements of the system and their attributes (see Figure 2). Attributes of functions include their purpose, criticality, where they are performed, inputs, outputs, initiators, and subfunctions. Attributes of information elements include size, criticality, using functions, subelements, where they reside, and how they are communicated. Note that facilities, documents, communication links, software, and personnel may all be represented within this framework.

Established system analysis and specification techniques (e.g., structured analysis) are recommended to ensure that all security relevant elements have been represented. For specific SSE efforts, modeling or specification tools may be used to represent the valued assets of the system. SSE on SDI, for example, used the Requirements Driven Design (RDD)[®] tool provided by Ascent Logic Corporation to model a portion of the system architecture. On other commercial efforts, simple prose descriptions have been deemed sufficient.

2. Threat Identification

Once the valued assets have been extracted from the baseline architecture, potential threats to those assets may be identified. By "potential threats" we mean those adversary-initiated occurrences that can adversely affect the system through compromise of valued assets. Compromise can occur in the form of loss, disclosure, modification and destruction of system elements, or denial of system services.

3. Threat Analysis and Decomposition

High-level potential threats serve as the starting point for further decomposition. Threat decomposition is performed using "threat logic trees," a structure required in [3] and similar to decision trees in other forms of risk management [4] and reliability engineering [6].

Figure 3 provides an example of a threat logic tree for System V/MLS, an AT&T B1 secure UNIX[®] System design effort.

In this computer-oriented example, the overall threat objective is to obtain administrative privileges on a UNIX system. This high level objective breaks down into alternative objectives of obtaining the administrative password or gaining physical access to the system console. The intermediate objectives decompose further, until eventually we reach the set of individual steps to achieve the primary objective.

Note that in each stage of the decomposition, an intermediate node is either the "AND" of its children, or the "OR." An "AND" node is an objective that requires the successful completion of *all* of its children (sub-objectives) in order to be achieved. An "OR" node requires the successful completion of *any* of its children to be successful. As we shall see in the next section, an "AND" node and an "OR" node inherit risks from their children in different ways. The notions of Risk, System Weighted Penalty (SWP), and Level of Adversary Effort (LAE) will be defined in the next section.

For each node in the threat logic tree, there is a corresponding entry in an SVA threat database (see Figure 2) that defines the threat in greater detail. The attributes of threat information include a description, its objective, its targeted assets, success criteria, type (Signal Intelligence, Sabotage, etc.), and risk attributes (Risk, SWP, LAE). The output of this database and the threat logic trees serve to document the results of the threat analysis. The database also serves as a repository of accumulated threat knowledge that may be applied to future SSE efforts.

4. Risk Assessment

When high-level threat objectives have been sufficiently decomposed, the next step is to assess the risks associated with the threat. Traditional formulas employed in risk management are based on the probability of a particular event times the loss associated with that event [4]. The difficulty in applying

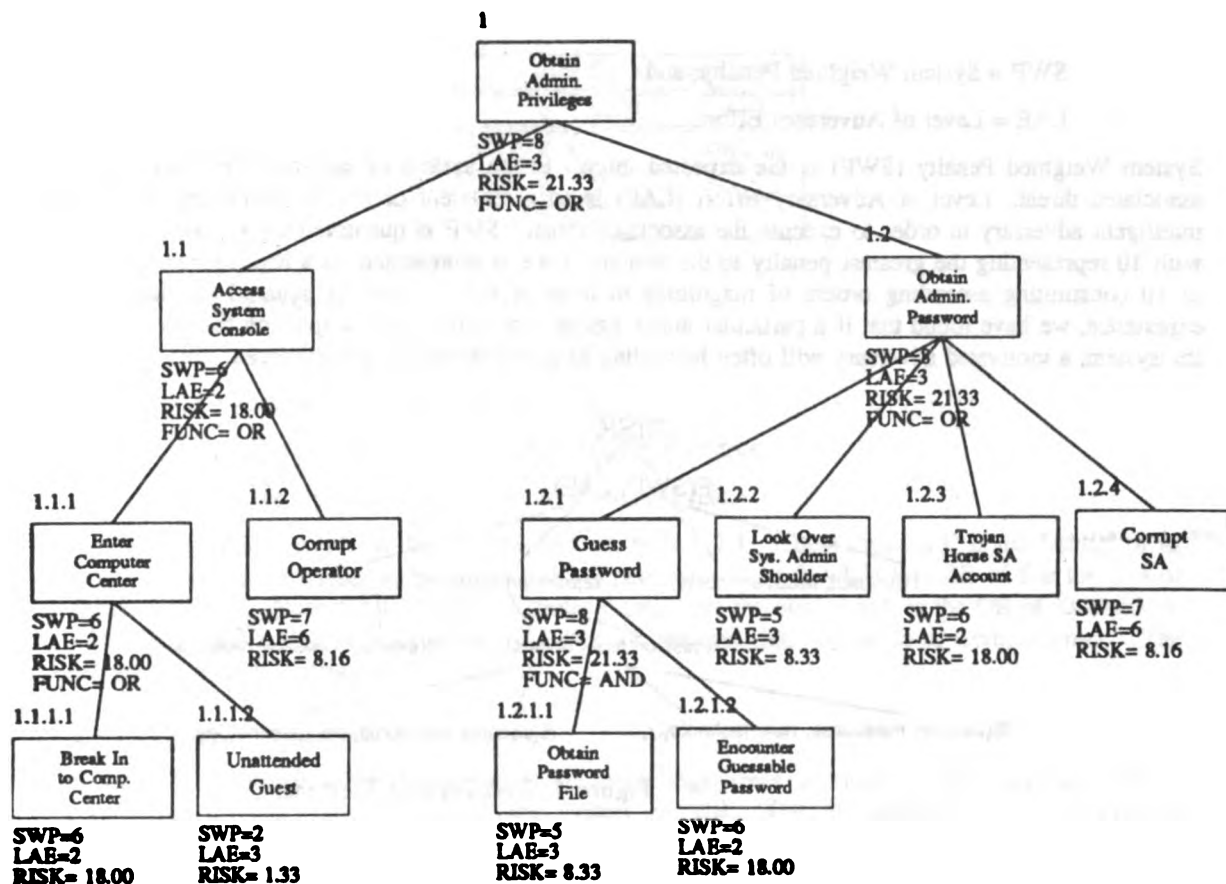


Figure 3. Threat Logic Tree Example - UNIX System Design

such a formula to SSE is that security "losses" are generally associated with adversary actions. Probability of loss is thus the product of the probabilities of attack and attack success. Probability of attack is often impossible to estimate for the following reasons:

- **Unknown Adversary** - A system designer does not necessarily know who will be trying to subvert his system. Thus, it is exceedingly difficult to predict the types, frequencies, and degrees of motivation that comprise the probability of attack.
- **Unknown Attributes of Adversary** - In cases where the adversary is known, the system designer often still lacks insight into the capabilities, dispositions, and resources available to the attacker. Within the federal government, for example, to the extent that this information is available at all, it is protected to the highest levels of secrecy within multiple compartments, and is difficult to integrate into the system design process.
- **Unknown Future** - Time favors the attacker. An adversary can succeed by exploiting a single weakness, while the defender must protect against all avenues of attack. When a technological advance adds new potency to a particular attack, the old risk assessments no longer apply. Furthermore, an adversary who is currently unmotivated to employ a particular attack may later become motivated on the basis of opportunity. Thus, it is difficult to predict if and when a low probability attack will become much more likely and effective.

For the reasons described above, it was necessary to derive a risk formula that does not require so accurate an assessment of the adversary psyche. Work within AT&T on product and service security assessments [7] was adapted to yield the following empirical formula:

$$\text{Risk} = \text{SWP}^2 / \text{LAE}$$

where:

SWP = System Weighted Penalty, and

LAE = Level of Adversary Effort.

System Weighted Penalty (SWP) is the expected impact to the system of successful execution of the associated threat. Level of Adversary Effort (LAE) is an assessment of the resources required by an intelligent adversary in order to execute the associated threat. SWP is quantified on a scale of 0 to 10 with 10 representing the greatest penalty to the system. LAE is represented on a logarithmic scale of 1 to 10 constituting ascending orders of magnitude in level of effort. SWP is squared because in our experience, we have found that if a particular attack has an especially severe impact on the operations of the system, a motivated adversary will often be willing to spend the additional resources.

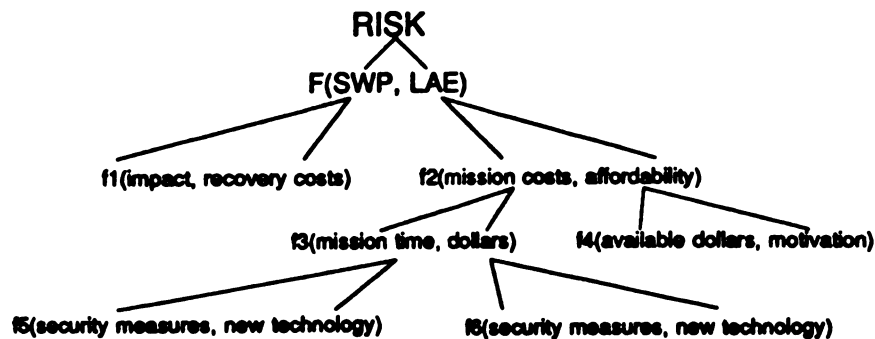


Figure 4. Risk Formula Elements

Intuitively, this formula states that an intelligent adversary is interested in achieving a maximum negative impact on the system for a minimum number of attack dollars spent.

An advantage of this formula is that by default, it assumes the worst case scenario of an adversary who applies available resources intelligently. Thus, there is no need to suppose specific adversaries and assess individual motivations. Furthermore, since SWP is the primary driver in the risk formula, a severe vulnerability that is difficult to exploit today will still be addressed, with the assumption that its presence will induce a future threat.

While the SSE risk formula does not depend on availability of data about the adversary, if such data are available, it may be applied toward an accurate calculation of LAE. Figure 4 shows a breakout of subcomponents of SWP and LAE. Estimation of these subcomponents on a scale of High, Medium, and Low have been applied in specific SSE efforts to derive SWP and LAE. For these specific efforts, a calculus for combining the subcomponents has been represented in tabular form.

The assumption of an intelligent adversary has not only influenced the derivation of the risk formula, but has also dictated the means by which risks propagate up the threat logic tree hierarchy. In the previous section, we discussed the two types of threat logic tree intermediate nodes, the "AND" and the "OR"; these nodes demonstrate different behaviors in inheriting risk attributes from their children.

For an "AND" node, risks are not directly inherited from a child. Instead, the LAE of the parent represents the sum of efforts of the children. That is, the effort associated with achieving an objective that requires all of a set of subobjectives to be achieved is the sum of the efforts of achieving all subobjectives. Because LAE is on a logarithmic scale, the sum of efforts for a parent is reflected as the maximum of its children's LAEs. The SWP for an "AND" node must be assessed and input independent of the SWP of its children, since it is often the case that the penalty of a set of successful actions is greater than the sum of the individual pieces.

The risk associated with a parent "OR" node is the maximum of the risks associated with its children. This means that we assume our adversary will choose the alternative attack that offers the greatest return on the dollar in achieving a higher-level threat objective.

The following table summarizes the risk calculations for parent nodes in the threat logic tree:

Risk Calculations		
	AND	OR
SWP	I	swp_{maxR}
LAE	$\sum_{i=1}^n lae_i$	lae_{maxR}

where:

I : independently assessed value;

swp_i : system weighted penalty for child i ;

lae_i : level of adversary effort for child i ;

n : number of children of the parent mission objective;

$maxR$: the child with the maximum associated risk.

In our UNIX system example in the previous section, node 1.2.1, "Guess Password" is the "AND" of its children. Therefore, its SWP has been assessed independently to be 8, while its LAE of 5 is the sum of its children's LAEs. Node 1.1.1, "Enter Computer Center" on the other hand, is the OR of its children. Its Risk, SWP, and LAE are those of its child with the greatest risk, "Break In to Comp. Center." The values are 18, 6, and 2, respectively.

5. Prioritization of Vulnerabilities

With the risks quantified in the framework of the threat logic trees, summary reports may be produced that rank the driving vulnerabilities of the system by risk. For the purpose of this process, a vulnerability may be thought of as a high-risk threat. A vulnerability analysis summary report for our UNIX system example would look as follows:

Vulnerability Analysis Summary Report				
ID	Threat Name	Risk	SWP	LAE
1.2.1	Guess Password	21.33	8	3
1.1.1.1	Break In to Comp. Center	18.00	6	2
1.2.1.2	Encounter Guessable Password	18.00	6	2
1.2.3	Trojan Horse SA Account	18.00	6	2
1.2.2	Look Over Sys. Admin Shoulder	8.33	5	3
1.2.1.1	Obtain Password File	8.33	5	3
1.1.2	Corrupt Operator	8.16	7	6
1.2.4	Corrupt SA	8.16	7	6
1.1.1.2	Unattended Guest	1.33	2	3

Note that intermediate "OR" nodes (1 - "Obtain Admin. Privileges", 1.1 - "Access System Console", 1.1.1 - "Enter Computer Center", and 1.2 - "Obtain Admin. Password") are not included in this summary report. These nodes are left out because their risk values depend directly on the values of their descendent "AND" and leaf nodes. Thus, as the risks of the "AND" and leaf nodes are managed, the risks of the associated "OR" nodes will be automatically reduced.

We are now ready to take each vulnerability in priority order and apply safeguards.

6. Identification of Candidate Safeguards

Safeguards for selected vulnerabilities are chosen from a safeguard and countermeasures database (see Figure 2) that represents a variety of security disciplines. These disciplines include:

- Computer Security (COMPUSEC),
- Communications Security (COMSEC),
- Physical Security (PHYSEC),
- Operations Security (OPSEC),
- Personnel Security (PERSEC), and
- Administrative Security.

Often, a particular vulnerability may be addressed by alternatives that span disciplines. For example, if we look at the highest-risk vulnerability in our example, "Guess Password" there is a range of alternatives for mitigation. We can apply a COMPUSEC solution of implementing machine generated passwords that are difficult to guess (as in [8]), or we can apply a more PERSEC or administrative approach of developing a training program for system users on the selection of unguessable passwords. The ultimate choice of safeguards, as will be shown in the next section, depends on relative effectiveness versus cost of the alternatives.

7. Safeguard Trade-off Analysis

The effectiveness of a particular safeguard candidate in a given environment may be quantified as an effect on the risk value of the safeguard's targeted vulnerability or vulnerabilities. Its costs may be quantified through standard cost estimation techniques (e.g., previous project data, cost models, simulation, prototyping). The following fields of information in the safeguards and countermeasures database support assessments of effectiveness and costs:

- safeguard description,
- safeguard type (e.g., COMPUSEC, COMSEC),
- safeguard alternatives,
- implementation costs,
- special life-cycle cost concerns (technology risks, reliability, maintainability, survivability, etc.).

A project-specific SSE management program must establish the means and interfaces through which costing of candidate architectures may be done in concert with other specialty engineering activities. This will be addressed in greater detail in step 10.

8. Security Architecture Selection

The output of the analysis process is a recommended security architecture, an assessment of associated costs, and a list of remaining vulnerabilities and their associated risks. The format for specification of the security architecture is dependent on the conventions of the project in which SSE is being applied. The security architecture may include a security policy, requirements specification, and/or design document (see Figure 2).

SVA outputs may be reviewed by the system providers or users to determine whether residual risks are acceptable or additional reductions through SSE iteration is required.

9. Security Architecture Integration

Integration of the recommended security architecture into the system design is a key issue in SSE. Integration is primarily a management and planning function, and requires allocation of suitable resources and identification of organizational structures and interfaces. Figure 5 illustrates some of the engineering area interfaces required for incorporation of security requirements into overall system design. In this diagram, security engineering has been placed in the center as the focus of this paper. In reality, a similar diagram could be generated around any engineering specialty area of emphasis.

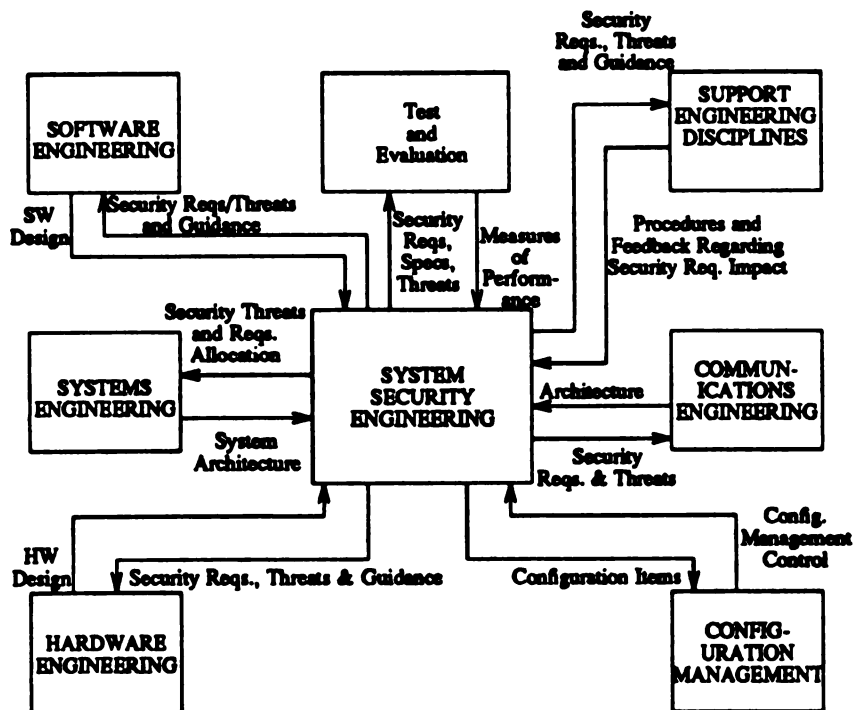


Figure 5. Example System Security Engineering Interfaces

In order to implement the integration function and interfaces represented in Figure 5, it is necessary to establish a detailed SSE management plan. The SSE management plan should address staffing, schedules, budget, avenues of interaction (e.g., meetings, working groups), and points of contact within related disciplines. The SSE management plan must be endorsed and supported within the framework of an overall system design effort in order to be effective.

10. Iteration

Like any other engineering task, SSE needs to be applied continuously throughout the system design process. As architectural changes are made, SSE must be applied to assess the impact of those changes on the security attributes and vulnerabilities of the system. Strict configuration management and version control of the databases, threat logic trees, and outputs of the SVA model must be maintained to track shifting architectures, allow "what-if" analyses, and return to a previous baseline.

AUTOMATED TOOLSET

A prototype Automated SSE Toolset (ASSET) has been developed by AT&T Bell Laboratories to implement the SSE process described above. It runs on an AT&T 630 Multi-Tasking Graphics terminal and supports the following features:

- efficient threat logic tree generation and management,
- automated risk calculation and recalculation capabilities,
- risk parameter and subparameter input forms,
- automated report generation of hardcopy threat logic trees and summary reports,
- automated threat and safeguard databases,
- integrated configuration management, and
- on-line help capabilities.

Future capabilities for the tool include:

- critical risk path highlighting for threat logic trees,
- generation of wall chart threat logic tree reports,
- incorporation of threat subtree libraries,
- automated safeguard trade-off analyses,
- integrated cost models,
- integrated system modeling capabilities, and
- support for non-expert users.

CONCLUSIONS

The SSE process described in this paper is designed to provide analytical support for security requirements. It applies sound engineering and risk management principles to administer security resources effectively. The toolset and underlying databases of the SSE process are evolving as the methodology is applied to a broader problem set.

Our primary obstacle in widely implementing SSE commercially, internally, and within the government has been in our inability to quantify the costs of NOT applying SSE. It is generally understood that security constitutes a risk to the current state of the art in information systems, and there has been anecdotal evidence of break-ins and viruses and their adverse effects. There remains, however, a general skepticism in the market that security is an important element against which design resources must be applied.

This is the primary issue to be resolved in broadening the application of SSE.

ACKNOWLEDGEMENTS

This paper represents the efforts of members of AT&T Bell Laboratories' System Security Engineering Group: Cheri Dowell, Don Gazzale, Dan Goddard, and Lima So. The author also wishes to acknowledge the thoughtful review of Ed Amoroso, Thu Nguyen, Howard Israel, Dave Schreiber, Pete Dinsmore, and Bill Leighton.

REFERENCES

- [1] Kleinrock, L., *Queueing Systems*, John Wiley and Sons, 1975, Volumes I and II.
- [2] Department of Defense, *Department of Defense Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD, December 1985.
- [3] Department of Defense, *System Security Engineering Program Management Requirements*, MIL-STD-1785, June 20, 1988.
- [4] Boehm, B. W., "Software Risk Management: Principles and Practices," *IEEE Software*, January 1991, pp. 32-41.
- [5] DeMarco, R., *Structured Analysis and System Specification*, Prentice Hall, 1978.
- [6] Musa, J., Ianino, A., and Okumotu, K., *Software Reliability Measurement, Prediction, and Application*, MacGraw-Hill Company, 1987.
- [7] Chancer, R., Charney, J., Kolchmeyer, P., and Mayer, W., "Security Assessment of Services, Products, and Architectures," AT&T Bell Laboratories Technical Memorandum, 55131-87008.01TM, October 8, 1987.
- [8] Amoroso, E., Heiland, D., and Israel, H., "Unified Password Generation," *Proceedings of the 3rd Annual Canadian Computer Security Symposium*, Ottawa, Canada, May 1991.

TEACHING COMPUTER SYSTEMS SECURITY IN AN UNDERGRADUATE COMPUTER SCIENCE CURRICULUM

ALFRED W. ARSENAULT

and

CAPTAIN GREGORY B. WHITE, USAF

Department of Computer Science,
U. S. Air Force Academy
USAF, CO 80840

arsenaul@usafa.af.mil
white@usafa.af.mil*

ABSTRACT

This paper examines how Computer Security should be taught in an undergraduate Computer Science curriculum. We will examine: (i) why a Computer Security course should be offered as an elective to undergraduate Computer Science majors; (ii) what should be the prerequisites for that course; and (iii) what should be the content of that course.

INTRODUCTION

As Higgins described at this conference in 1989[HIGG], and as an informal survey conducted by the authors this year seems to confirm, most university Computer Science programs do not offer a course in Computer Security designed for undergraduate Computer Science majors. We believe that this is an oversight that will only result in the continual problems with Computer Systems Security we see today. All undergraduate students majoring in Computer Science (and related fields) should have the opportunity to be exposed to topics and issues in Computer Security.

Many of the institutions that do offer a Computer Security course concentrate on the 'non-technical' aspects of the field. Others combine cryptography with other topics in their Computer Security courses. This paper will show that a course covering the non-cryptographic technical aspects of Computer Security -

*The development of this paper was supported by the United States Government. Sponsoring Organization: United States Air Force HQ USAFA/DFCS

what we refer to as Computer Systems Security - would be beneficial for students, for universities, and for the computing community as a whole.

This paper consists of four sections and two appendices. We first define our terminology - what we mean when we describe a course in 'Computer Systems Security'. We then describe why we believe a course in Computer Systems Security should be taught, and then describe what the prerequisites for such a course should be. Finally, we discuss the content of the course itself.

The first appendix describes the results of an informal survey conducted by the authors as to what Computer Science departments offer undergraduate Computer Security courses, and some of the details (textbooks, prerequisites, etc.) of those courses.

The second appendix proposes a course schedule for a one-semester, fifteen-week Computer System Security course.

WHAT IS 'COMPUTER SYSTEMS SECURITY'?

Before we can discuss why a course on Computer Systems Security should be taught in any curriculum, we must define our terminology. According to the NCSC's Glossary of Computer Security Terms, Computer Security is synonymous with Automated Information System (AIS) Security, which is

"Measures and controls that protect an AIS against denial of service and unauthorized (accidental or intentional) disclosure, modification, or destruction of AISs and data. ... includes ... all hardware and/or software functions, characteristics and/or features; operational procedures, accountability procedures, and access controls ...; management constraints; physical structures and devices; and personnel and communication controls needed to provide an acceptable level of risk for the AIS and for the data and information contained in the AIS...."[GLOS]

A course which addressed in detail all of the topics indicated by this definition would include too much material to cover in a one-semester, undergraduate Computer Science course.¹ Additionally, much of Computer Security is either site-specific (e.g., physical security at a particular facility) or too system-specific (e.g., proper administrative procedures for specific releases of operating systems) to adequately cover in more than

¹ Higgins suggested that a survey course in Computer Security be offered as a first course, to be followed by courses in systems security and cryptanalysis.[HIGG] The course we are describing combines the survey course with the systems security course, since we do not believe it likely that many departments will develop three separate courses on security.

passing depth. It seems appropriate, then, to limit the scope of the course, and to cover a subset of the topics of 'Computer Security' in more depth.

We refer to the subset of topics to be covered as 'Computer Systems Security,' and concentrate on the hardware and software aspects of Computer Security. We also include sections on Risk Analysis as it relates to computer systems, and some other short security-related topics. This is not intended to indicate that other areas of Computer Security are not worthy of study, but only to narrow the scope of a course to that which is appropriate for a single semester.

WHY SHOULD A COURSE ON 'COMPUTER SYSTEMS SECURITY' BE OFFERED?

We believe it is important for people receiving Bachelor of Science (or equivalent) degrees in Computer Science to have the opportunity to become familiar with the field of Computer Systems Security. Computer Systems Security is a key part of the overall effort to develop more trustworthy computer systems. As the recent publication Computers at Risk: Safe Computing in the Information Age ("the NRC report") stated:

"Security, Safety, and Reliability together are elements of system trustworthiness - which inspires the confidence that a system will do what it is expected to do."[COMP]

Computer Systems Security must be thoroughly intertwined with all aspects of system and software development if we are to reach the point where we have a reasonable level of confidence that our computers are doing only what we want them to do.

The NRC report also proposed a research agenda for Computer Security - a list of areas in which research is desperately needed if our Computer Security posture is to improve. Whether the posture needs to improve or not should no longer be a subject for debate. Quoting from the NRC report again:

"Without more responsible design and use, system disruptions will increase, with harmful consequences for society. They will also result in lost opportunities from the failure to put computer and communications systems to their best use.

In order to reach the point where "more responsible design and use" of computers is realized will require an effort by everyone involved in the computer field--especially the colleges and universities which have traditionally been excellent places to conduct research.

Potentially significant amounts of research funding seem to be available for work in computer systems security.[VAUG] The NRC report calls for significantly increased Government funding of computer security research.[COMP] It is to the advantage of departments interested in doing research in this field if their

graduate students have some familiarity with Computer Systems Security. If these students have taken courses in Computer Systems Security as undergraduates, they will constitute a base on which further research can more easily proceed.

It is in the computer industry's best interests to have software and system developers knowledgeable in computer systems security techniques and fundamental concepts. In fact, it is reasonable for them to assume that a system developer they hire understands the fundamentals of system security. Security has been shown many times to be significantly cheaper and more effective to design into a system than to try to add on later (see, for example, [NEUM]). Unfortunately, this is a lesson that has too often been omitted in the education of graduates from most computer science programs. Is it not logical to have colleges and universities offer Computer System Security courses that will teach this basic lesson before their graduates enter the work force?

The students themselves will find it advantageous to study Computer Systems Security since it is an area that can greatly affect their careers. Developers and designers of future software and systems will need to understand the roles of security in developing reliable, trustworthy systems and software. Additionally, since viruses, Trojan horses, worms, and other malicious logic are becoming more common, it is imperative that all graduates of computer science programs be familiar with techniques to detect, prevent, and/or limit the damage that such malicious logic can cause.

Given this, we believe that Computer Science departments should offer courses in Computer Systems Security. Unfortunately, as the results of our informal survey showed, this is not the case. In fact, the norm is probably what one of the respondents to our survey stated--that no undergraduate Computer Security Course was offered because the topic is addressed in other courses such as Operating Systems, Data Base, and Networks. While at first glance this might appear to be sufficient, it does not provide the in-depth study necessary to further research. Additionally, we do not believe that incorporating the fundamentals of Computer Systems Security into other courses will cover the subject in sufficient detail. There is a need for a focused course that looks at the history, the experiences that we have learned from, the design techniques, work examples, and other topics. Quoting from the NRC report again:

"Working on secure software requires yet more skills. Most notably, one must be trained to understand the potential for attack, for software in general and for the specific application domain in particular."([COMP], p. 117)

While we agree that other courses should discuss security as it relates to that course's material, it is not enough to rely on them solely in order to obtain the skills mentioned in the NRC report. A useful analogy to illustrate this point can be drawn

between Computer Systems Security and Operating Systems. Operating systems are addressed in many courses; however, most institutions offer at least one course dedicated to Operating Systems. The material covered in other courses usually relates to how an operating system is used to support other computer programs. A course dedicated to operating systems usually addresses how they work, and describes many of the design issues involved in writing an operating system. In a similar manner, computer security, when addressed in other courses, deals with how security affects that topic, while a course dedicated to Computer Systems Security would address the fundamental design issues and the details involved in security. What then are the topics that would be covered and what are the prerequisites of such a course?

WHAT SHOULD BE THE PREREQUISITES FOR SUCH A COURSE?

The course content (described in detail in the next section) is fairly technical. Many of the ideas extend foundational concepts established in other courses. Therefore, it is appropriate that students enrolled in the course be expected to have a good working knowledge of computers and software before the course begins.

Detailed knowledge of at least one high-level programming language should be required. Pascal, C, C++, and Ada are all acceptable; others would be as well.

Knowledge of operating systems and how they work (including some familiarity with computer architecture) should be required prior to taking this course. (Some instructors may find it acceptable if students are taking an Operating Systems course concurrently.) Much of the course is illustrated by showing how operating systems protect (or fail to protect) their various resources; it is therefore necessary for students to understand the basic concepts being relied upon.

Knowledge of database management systems (DBMS) and/or networks should not necessarily be required, although it is helpful in covering those units if students have some general familiarity with the topics. Additionally, knowledge of software engineering and software specification and verification would be helpful.

WHAT SHOULD BE THE CONTENTS OF SUCH A COURSE?

This section will discuss in detail what we believe the contents of a Computer Systems Security course should be. (Appendix II of this paper lists a suggested class schedule that describes how and in what order we believe the topics should be covered.) Again, this is not intended to be a 'hard and fast' syllabus, to be followed by everyone, but instead a set of topics that we feel, based on our experiences, are appropriate for a

course of this type. We welcome suggested changes and other comments.

We begin with a caveat about cryptography. Cryptography is an important topic, as it is an important mechanism in Computer Security. It would be very helpful if students in the course had an understanding of how this mechanism worked, and what its uses and limitations were. We believe, however, that Cryptography is such an important topic and should be addressed in such detail that it should not be covered in this course. It should instead be covered in a separate course, devoted to Cryptography and related topics. (One of the authors has experience with such a division, and was very pleased with the results.)

Therefore, in the rest of this section, we will assume that Cryptography is offered as a separate course. It will only be covered here where it is an appropriate mechanism, and then only in the context of its uses to provide security services, without providing details of 'how Cryptography works'. If it is not the case at a particular institution that Cryptography is offered separately, then it should be added to the materials listed in this section, and other material may have to be deleted.²

(It is not necessary for students to have taken a Cryptography course by the time they enroll in Computer Systems Security. For many students, Cryptography can/will be best described as a 'black box' - something goes in, something else comes out, and we can reasonably assume that certain security services are provided. We are much more interested in the uses and limitations of Cryptography in this course than in the technical details of implementations.)

Note that, at present, there is no textbook that exists that matches the course we describe. There are only a few Computer Security textbooks in print (they are identified in Appendices I and II). Each book has its strengths and weaknesses; however, we do not believe that any of them cover all of the topics we suggest in sufficient detail.

As with most courses, we believe that it is appropriate to start off with an introduction to the course and an overview of the material to be presented. Thus, we recommend spending some time early in the course describing Computer Systems Security and why it is important, and then discussing the three goals of Confidentiality, Integrity and Availability of data. A discussion of some of the major privacy concerns is appropriate at this time, as is at least a brief discussion of computer ethics.

² We leave it to the instructor's discretion as to what material to delete. Some may choose to compress other topics, so that all suggested topics are covered, but in less detail; others may choose to delete one or two topics altogether.

We think that it is important to place much of the course material in its proper context. Thus, we recommend that Risk Analysis be the next major topic covered. This discussion should include a general discussion of threats to computer systems, vulnerabilities in computer systems, and countermeasures available to thwart some of the threats and close some of the vulnerabilities. If possible, the instructor may want to discuss threats/vulnerabilities/ countermeasures relating to specific systems, such as those the university's academic computing center uses. We have found that this tends to raise students' interests, since it is something to which they can directly relate.

Once this unit has been completed, the next topic should be a discussion of a specific threat: malicious code. Addressing this topic early in the course provides motivation to the students. They can see and understand some of the specific problems, they can relate it to what they have been seeing (or maybe even experiencing), and they can relate each countermeasure discussed during the course back to this topic, to help determine the effectiveness of the countermeasure.

There is admittedly a potential problem here. We do not wish to provide students with a roadmap describing how to break into any specific system. On the other hand, there are several articles in the technical literature that describe generic (and sometimes specific) vulnerabilities in systems. Many times these vulnerabilities have not been fixed in university-owned computer systems. Thus, the instructor will have to make a decision about how much detail to go into - here and throughout the course. The authors' best recommendation is to try to gauge the level of the students, and discuss material in a depth appropriate for the particular class.³

We recommend next describing when and how to build 'secure' computer systems. The instructor should cover the importance of deciding what security measures are important, given the intended uses of the system. When technical security measures are

³ One of the authors was faced with an interesting situation because of this issue. The 'dictionary attack' on UNIX(Tm) systems has been discussed in the literature many times. A 1979 paper by Morris and Thompson[MORR] describes it in sufficient detail, as do many of the papers that have been written about the 'Internet worm' of November 1988 (e.g., [EICH]). After discussing some of the technical literature during the semester (replete with admonitions to 'not try this yourself'), the author was presented with a short program, written in C, and a list of user identifiers and corresponding passwords for one of the university's computer systems, obtained by using a dictionary attack.

Tm UNIX is a registered trademark of UNIX System Laboratories, Inc.

appropriate, the instructor should address the importance of designing security into a system from the beginning - including the benefits of doing so, and the consequences of not doing so. A description of the differences between security mechanisms and security assurances should follow next and then a description of each of the important security mechanisms. We recommend beginning with authentication, then moving onto access controls and information flow controls, and finishing up with auditing. It is important to discuss each of these mechanisms in the context of providing confidentiality, integrity, and availability of data - we do not think that any of the three should be singled out as 'most important'.

The next major unit is a discussion of security assurances. This material should be tied in with software safety and reliability, and discussed in the context of developing trustworthy software.⁴

The specific topics we recommend covering include: program and system correctness; minimization of security-relevant hardware and software; security models; system, subsystem, and program specification; consistency among models, specifications, and implementations; and the reference monitor concept.[ANDE]

After completion of the security assurance unit, we recommend that the instructor take time to cover one or two case studies. These case studies would be discussions of the security (or lack thereof) provided by specific operating systems. The choice of which operating systems to cover would be up to the instructor; ideally, they would be systems with which students are familiar.

The final major unit of the course should cover network and applications security. The network security lectures should address security issues relevant to the International Standards Organization's Open Systems Interconnection Protocol Reference Model, as well as other protocol reference models. The applications security unit could include such topics as database management system (DBMS) security, virtual machine monitors, and embedded systems. If DBMS security is chosen as an appropriate application to study, the unit should address differences between operating system security and DBMS security, as well as the problems of inference and aggregation. Although there is not a great deal of detail that can be provided in these areas, students should at a minimum be made aware of the problems, and some limited solutions.

We recommend concluding the course with one or more case studies of system security (as distinct from operating system security). Appropriate topics might include the Internet worm or other malicious code attacks; systems that have been designed to

⁴ Note that, although we concentrate on software in this course, we recommend discussing hardware and firmware where appropriate.

provide security, and how well they provided it; and networks with which the instructor is familiar.

If time permits, the instructor may also wish to provide an overview of some of the computer security efforts ongoing in both the government and private industry. Appropriate topics here might include the DoD Trusted Computer System Evaluation Criteria [TCSE], the European Information Technology Security Evaluation Criteria (also called the Harmonised Criteria), various FIPS pubs, the IEEE POSIX ('Portable Operating System Interface for Computer Environments') effort, and other security efforts.

CONCLUSIONS

In this paper, we have argued that most undergraduate Computer Science programs should offer at least one course in Computer Security. Unfortunately, as shown by the results of both our informal survey (see Appendix I) and the survey conducted by Higgins in 1989, relatively few colleges currently offer any courses in Computer Security. We believe that this is an error that must be rectified.

We have provided reasons why a course in Computer Systems Security should be offered. We have described what we consider to be appropriate prerequisites for the course. We have also suggested a set of topics to be addressed in the course. (In Appendix II, we provide a suggested schedule for the course.) Again, we emphasize that there are currently no textbooks in print that adequately cover the list of topics we propose.

Our primary reason for proposing this course schedule is to foster discussion. We welcome suggested changes to and other comments on this proposal.

ACKNOWLEDGEMENTS

We would like to thank all of those who helped us in the preparation of this paper, and in the gathering of information in our survey. In particular, we thank Lt. Col. Lawrence Jones, USAFA, Dr. Lionel Deimel of SEI, Dr. John Higgins of BYU, COL Ray Vaughn of the U. S. Naval Academy, and all of those who responded to our informal survey.

REFERENCES

[ANDE] Anderson, James P., Computer Security Technology Planning Study, ESD-TR-73-51, Volume 1, Hanscom AFB, MA, October 1972.

[COHE] Cohen, Fred, "Computer Viruses: Theory and Experiments", in Proceedings of the 7th DoD/NBS Computer Security Conference, Gaithersburg, MD, 24-26 September 1984.

[COMP] National Research Council, Computers at Risk: Safe Computing in the Information Age, National Academy Press, Washington, DC, 1991.

- [DENN] Denning, Dorothy E., Cryptography and Data Security, Addison-Wesley, Reading, MA, 1982.
- [EICH] Eichin, Mark W., and Rochlis, Jon A., "With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988", in Proceedings of the 1989 IEEE Computer Society Symposium on Security and Privacy, Oakland, CA, May 1-3, 1989.
- [GASS] Gasser, Morrie, Building a Secure Computer System, Van Nostrand Reinhold, New York, 1988.
- [GLOS] National Computer Security Center, Glossary of Computer Security Terms, NCSC-TG-004, Version 1, 21 October 1988.
- [HIGG] Higgins, John C., "Information Security as a Topic in Undergraduate Education of Computer Scientists", in Proceedings of the 12th National Computer Security Conference, Baltimore, MD, 10-13 October 1989.
- [LAMP] Lampson, Butler W., "A Note on the Confinement Problem", in Communications of the ACM, Volume 16, No. 10, October 1973.
- [LOBE] Lobel, Jerome, Foiling the System Breakers: Computer Security and Access Control
- [MART] Martin, James, Security, Accuracy and Privacy in Computer Systems
- [MORR] Morris, Robert, and Ken Thompson, "Password Security: A Case History", in Communications of the ACM, Volume 22, Number 11, November 1979.
- [NEUM] Neumann, Peter G., "Computer Security Evaluation", in AFIPS Conference Proceedings, Vol. 47, 1978.
- [PFLE] Pfleeger, Charles, Security in Computing, Prentice-Hall, 1987.
- [SEED] Software Engineering Institute, Software Engineering Education Directory, Technical Report CMU/SEI-90-TR-4, April 1990.
- [TCSE] Department of Defense Trusted Computer System Evaluation Criteria, DoD Standard 5200.28-STD, 26 December 1985.
- [THOM] Thompson, Ken, "Reflections on Trusting Trust", in Communications of the ACM, Volume 27, No. 8, August 1984.
- [VAUG] Vaughn, Ray, private communication, 12 March 1991.
- APPENDIX I: RESULTS OF AN INFORMAL SURVEY OF UNDERGRADUATE COMPUTER SECURITY COURSES

In December, 1990, and January, 1991, the authors conducted an informal survey of colleges and universities to determine what

was being offered in the way of Computer Security courses of the type we describe.

We used several different methods to collect data for this survey. Messages asking for information were broadcast on several Internet newsgroups. We consulted college catalogs available to us in the U. S. Air Force Academy Library, looking for offerings of Computer Security courses. We consulted the Software Engineering Education Directory[SEED], published by the Software Engineering Institute, to determine which universities listed Computer Security Courses as part of their Software Engineering curriculum. And finally, we contacted some universities directly.

Note that, at the time of this writing, the survey is still ongoing. Thus the results below should be regarded as preliminary.

In the survey, we requested answers to several questions:

(1) Does your school offer a course in Computer Security as part of its undergraduate Computer Science curriculum? If so, what is the title of that course?

RESULTS: Five departments reported no course in Computer Security. (Certainly, there are many other departments without Computer Security courses that did not respond to our requests for information.)

Four departments - the U. S. Naval Academy, the University of Maryland-Baltimore County, the University of Maryland-University College, and California State University at Northridge reported Computer Security courses specifically designed for undergraduates.

Ten departments - the University of California at Davis, UC-Berkeley, California State University at Hayward, Brigham Young University, Arizona State University, George Mason University, George Washington University, the University of Seattle, Lehigh University, and Carnegie Mellon University - list graduate courses in Computer Security which may also be taken by qualified undergraduate students.

The University of Toronto offers a graduate course in Computer Security that is not open to undergraduates.

Texas Christian University is developing a course in "Security, Reliability and Safety", with a first offering scheduled for the 1991-1992 academic year.

Names of the courses varied widely, with no discernable trend.

We are not certain at this time why more schools offered Computer Security courses at the graduate level than at the

undergraduate level. Possibly, this is a result of graduate programs being more inclined to focus on the 'research topics' in Computer Science in an effort to make a contribution to the discipline. Certainly there are many unsolved security problems and therefore the security discipline is still worthy of graduate research. However, the discipline is now about twenty years old and enough has been learned and documented to make this an interesting and useful topic for undergraduates [VAUG], and we are somewhat disappointed that there are not more departments offering undergraduate Computer Systems Security courses.

(2) If so, is the course required or an elective for Computer Science majors?

RESULTS: In all cases, Computer Security courses are electives.

(3) What textbook is being used, if any?

RESULTS: (n.b. In several cases, we have thus far been unable to determine what if any textbook is being used for a particular course.) There seems to be no clear consensus here. Many instructors use their own notes to teach, and do not use a textbook. Five different books were each mentioned by at least one department:

Pfleeger's Security in Computing[PFLE];

Gasser's Building a Secure Computer System[GASS];

Denning's Cryptography and Data Security[DENN];

Lobel's Foiling the System Breakers: Computer Security and Access Control[LOBE]; and

Martin's Security, Accuracy, and Privacy in Computer Systems[MART].

(4) What are the prerequisites for the Computer Security course?

RESULTS: Prerequisites generally include upper-division standing, plus at least one programming course. Many departments also require Data Structures, and some require Operating Systems.

Departments whose Computer Security Course includes Cryptography generally also required one or more advanced Mathematics courses, with Number Theory being a fairly common requirement.

(5) Is the course offered once a year, or every semester/quarter?

RESULTS: In all cases where we could verify the offering, the Computer Security course is offered at most once a year. In

many departments, the course is offered less frequently than once a year, and some departments that list a course in Computer Security report going several years without offering it.

(6) Approximately how many students typically enroll in the course?

RESULTS: This varied widely by department, with a low of 10 and a high of about 40 students being reported. (Note: in most instances, student count included both graduate and undergraduate students.)

(7) If your institution does not offer an undergraduate Computer Security course, is there a particular reason (e.g., no faculty interest in teaching such a course; not enough students interested in taking such a course; no room in the undergraduate Computer Science curriculum for another course)?

RESULTS: Few departments responded to this question. Those that did provide reasons included two departments where there is no room in the undergraduate curriculum, one department with no faculty member qualified to teach the course, and one department that believe they adequately cover Computer Security in other courses.

In summary, one can see that there are not many departments that currently offer Computer Security courses. When one combines this with Higgins' findings (only 26 of the 102 departments he surveyed in 1989 offered computer security courses), one sees that there is not yet the significant trend toward Computer Security as a legitimate academic topic that one might hope for.

APPENDIX II: A RECOMMENDED CLASS SCHEDULE FOR A COURSE IN COMPUTER SYSTEMS SECURITY

This Appendix contains a class schedule that the authors believe is appropriate for an undergraduate course in Computer Security. This is a minor modification of a course that has been taught at the University of Maryland-Baltimore County by one of the authors.

This course does not assume the use of any particular book. It can make use of an appropriate textbook (e.g., Pfleeger's Security in Computing[PFLE], Denning's Cryptography and Data Security[DENN], or Gasser's Building a Secure Computer System[GASS]) along with supplemental material, or it can use only material from the technical literature, without a central textbook.

Among the materials which can be used to supplement any textbook are: Anderson's Computer Security Technology Planning Study[ANDE], Cohen's "Computer Viruses: Theory and Experiments" [COHE]; Lampson's "A Note on the Confinement Problem" [LAMP];

Thompson's "Reflections on Trusting Trust" [THOM]; and the DoD Trusted Computer System Evaluation Criteria [TCSE].

Discussion of the Schedule

This schedule starts with an overview of the field of Computer Systems Security, and discusses each of the three major parts: Confidentiality, Integrity, and Availability. It then moves into a short unit on Risk Analysis. Following this, there is a one-week unit on malicious code, and the various types of malicious code that exist. The rest of the time prior to the first examination is spent discussing some of the fundamentals of computer system security.

Between the first and second examinations, we cover several popular security mechanisms. We begin with authentication mechanisms (passwords, biometric devices, etc.), follow with various access control mechanisms, and conclude with audit mechanisms and audit trail analysis.

Between the second and third examinations, we address security assurances, and then move to case studies of two particular operating systems.

After the third examination, we move to other aspects of Computer Systems Security. We begin by addressing network security. We then move to Database Management System (DBMS) security issues, including such topics as inference and aggregation. We conclude the course with two more case studies.

Assumptions

We should point out that there are some assumptions built into this class schedule. First, we assume that the course in question meets 45 times (three meetings per week, for fifteen weeks) over the course of a semester, and that there are 50 minutes per course meeting. We assume that there will be three examinations (in addition to the final) given in the course, and that there will be some time spent reviewing for each of them. And, as stated in the paper, we assume that Cryptography is addressed in a separate course.

Lesson #	Topic to be addressed
	INTRODUCTORY MATERIAL
1	Introduction - Course overview and rules; What is Computer System Security? Confidentiality, Integrity, Availability
2	Computer Systems Security, Privacy, and Ethics
	RISK ANALYSIS
3	Risk Analysis - Threats, Vulnerabilities, and Countermeasures
4	Risk Analysis

	THREATS, VULNERABILITIES, AND COUNTERMEASURES
5	Threats and Vulnerabilities: Malicious code
6	Malicious Code: Trojan Horses, Time Bombs, Trap Doors
7	Malicious Code: Worms and Viruses
	SECURE COMPUTER SYSTEMS: OVERVIEW
8	The Parts of a Secure Computer System
9	Designing a System for Security
10	The Current Security Status of Computer Systems
11	Summary of Material Covered to Date and Review for Exam #1
12	Examination # 1
	SECURITY MECHANISMS
13	Authentication - Spoofing, Trusted Path, Password-based Authentication Systems
14	Authentication - Mechanisms Other Than Passwords
15	Access controls: Discretionary Access Controls, Access Matrices, and 'Safe' Systems
16	Access Control Mechanisms: Access Control Lists, Protection Bits and Capabilities
17	Information Flow Controls: Mandatory Access Controls and the Lattice Model
18	Information Flow Control Mechanisms
19	Information Flow Control Limitations and Covert Channels
20	Extensions to Conventional Controls: Access Controls to Meet Specific Security Requirements
21	Auditing Events
22	Auditing and Audit Trail Analysis
23	Audit Trail Analysis and Review for Exam #2
24	Examination #2
	SECURITY ASSURANCE
25	Security Assurances: Overview and Importance
26	Correctness of Programs and Systems - Minimization of Security Relevant Hardware/Software
27	Security Models: Bell-LaPadula, Clark- Wilson, Goguen-Meseguer
28	System and Subsystem Specifications
29	Consistency Among Models and Specifications
30	Coding the System - Correctness and Consistency Considerations
31	Security Kernels and other Reference Monitor Implementations
32	Hardware Security Requirements and Issues
33	Case Study: Operating System #1
34	Case Study: Operating System #2

35	Other Assurance Techniques and Review for
	Exam #3
36	Examination #3
	NETWORK AND DATABASE SECURITY
37	Network Security - What is a network?
38	The ISO Protocol Reference Model and its
	Security Addendum
39	Other Protocol Suites and Security Issues
40	DBMS Security - Differences from Operating
	Systems
41	Inference and Aggregation
42	Case Study #1
43	Case Study #2
	WRAP-UP AND SUMMARY
44	Government and Other Computer Security
	Efforts - A Summary
45	Course Wrap-up and Review for Final Exam

TOWARD CERTIFICATION, A SURVEY OF THREE METHODOLOGIES

Captain Charles R. Pierce

**Air Force Cryptologic Support Center
San Antonio, Texas 78243-5000**

ABSTRACT

The purpose of this paper is to provide an overview of three computer security certification methodologies and their applicability. The applicability of each methodology to a particular system depends somewhat on the system's stage of development in the computer systems life cycle.

INTRODUCTION

NCSC-TG-004, Glossary of Computer Terms [1], defines certification as, "The comprehensive evaluation of the technical and nontechnical security features of an AIS and other safeguards, made in support of the accreditation process, that establishes the extent to which a particular design and implementation meet a specified set of security requirements." Terms such as "comprehensive," "technical and nontechnical," and "other" imply that there is a large amount of inexactness to the science (art?) of computer security certification. The definition also implies that there likely is no firm certification target, since the goal is to only measure the "extent" to which security requirements are met. Nebulous terms and moving targets. Given this situation, it is quite easy to see why certification may not readily lend itself to standard methodologies and measures of success. Adding to these frustrations is a time factor, i.e., when was the certification begun (before system design or after it was built). This time factor can affect who does the certification, the system developer or system implementor.

There is veritably a different certification methodology for every system, either existing or developmental. There are no DOD standards for performing certification during system development or acquisition. The DODD 5200.28 [2] directs that certification be done but provides no criteria methodology. Each of the military services requires system certification in their implementing regulations but provide no standard. In the Air Force, both the AFR 700 (for AISs) and 800 (for embedded resources) series publications reference AFR 205-16 [3] for computer security certification guidance. AFR 205-16 (soon to be replaced by a series of three other regulations) divides certification requirements into three subsets, hardware and system software, applications software, and the operating facility, but also provides no standard methodology. This can lead one to the belief that either a standard methodology can not apply to certification or else using differing situational methodologies may be the best approach. This paper examines three methodologies with some annotation as their usability, but leaves the reader with deciding "will this work for me?" On the other hand, there are proposals for standardizing certification [4], and the reader can contemplate "will these methodologies fit into any standard?"

MITRE. "MANAGEMENT PLAN FOR COMPUTER SECURITY CERTIFICATION OF AIR FORCE SYSTEMS."

The Mitre Management Plan for Computer Systems Certification [5], done under contract with the Rome Air Development Center for the Air Force Cryptologic Support Center (AFCSC), uses DOD-STD-2167A, Defense System Software Development [6], as its basis. It uses the AFR 800-14, Life Cycle Management of Computer Resources in Systems [7], and AFR 205-16, Computer Security Policy [3], implementations of the DOD standard, including roles and responsibilities, adding the Air Force's certification process. It introduces the term "Certification Manager" to provide a single name for those different positions indicated as "certifying authorities" in AFR 205-16.

The Process

Security engineering tasks and products are placed in appropriate locations alongside standard system reviews (as defined in MIL-STD-1521B [8]) or developmental products. Certification tasks are also matched to the standard system development life cycle. A System Program Office (SPO) or Program Management Office (PMO) structure is assumed for the developing agency and most tasks and products are contractor deliverables. The certification tasks consists of SPO review, evaluation, or validation of these products. The user is limited to providing the initial set of security requirements. Actions reserved for the SPO are performing risk assessments and providing the actual certification. The Plan does not provide any accreditation specific activities that are not part of the certification process.

Since it is primarily concerned with contractor developed systems, the Plan concentrates on the engineering and manufacturing development (EMD) phase of the life cycle. However, for completeness the concept exploration and definitions actions required to generate system requirements (mission need statement (MNS) and concepts of operation (CONOPS)) and the related security products (security CONOPS and security policy) are included. Certification Manager actions include reviewing requirements and planning for certification. All other actions are in EMD except for final certification which occurs during production and deployment.

The tasks are sequential in nature as they are tied to the system development schedule. Each task description includes a list of inputs (from the contractor) required for performing the task, a description of the specific certification task actions, and expected outputs (from the SPO analysts) (see Figure 1 for a sample task). User and Computer Security Working Group (CSWG) interactions are not extensively covered since the Plan's target audience and certification task performers are to be program office personnel (who will normally be members of the CSWG). The end product of the task sequence is the certification. Certification maintenance, to occur every three years, is a part of the operations and support phase of the life cycle and is not specifically addressed. This maintenance would be highly dependent on certification support products developed according to the Plan however. Therefore, maintaining these products, within established risk management boundaries, would constitute a significant portion of certification maintenance. Although not originally written as such, the Plan is a fairly complete life cycle oriented document, from a program office or contractor developmental point of view.

Task 4.3 - Evaluate Detailed Security Design

Task Inputs

Software Requirements Specification
Design Review Presentations
Security Audit Trade Study
Top Level Design Documentation
Detailed Design Documentation

Task Description

This task continues the on-going process of evaluating the security design. It includes an analysis of the security engineering efforts to correlate the requirements, as identified in the system/segment specification and appropriate regulations, with the design provisions as described in the design documentation. Assuming that the security architecture and general framework has already been found acceptable (as a result of Task 4.1), this evaluation is directed towards the design details that will provide the detailed functional and protective requirements. This activity starts at the time of the system Preliminary Design Review (PDR). The analysis and reporting required by this task is significant, and as the design progresses and matures this work also needs to be continued and updated. It must analyze all changes or updates to the SRS/IRS and include a review of the Design Documentation or C-level specifications. As the review progresses, potential security vulnerabilities should be promptly identified and communicated to the developers for corrective action.

The Security Audit Trade Study needs to be evaluated as part of this task, in order to ascertain that it contains the proper trade-off analysis, and it identifies the elements and circumstances that will be audited including the rationale for their selection based on security requirements, performance impacts, costs, etc.

The security Working Group provides a convenient forum in which the certifier can request clarifications and address the issues at requirements interpretations and possible conflicts among requirements.

Task Outputs

Evaluated Security Design
Risk Assessment Results

Figure 1. Sample Task Description

AFR 56-31. COMPUTER SECURITY IN THE AIR FORCE ACQUISITION SYSTEM

The Air Force is developing a series of regulations and guidelines for certifying systems. A new regulation AFR 56-31, Computer Security in the Air

Force Acquisition System [9], defines development life cycle activities that leads to system certification. Primarily, Air Force System Security Memorandum (AFSSM) 5010, Computer Security in the Acquisition Life Cycle [10], takes the DOD-STD-2167A life cycle as its basic foundation and adds to the process those security actions that must occur. The goal is provide full life cycle guidance for certification and accreditation. The AFSSM activities actually begin before the DOD-STD-2167A defined life cycle by including considerations for mission needs analysis. User requirements and the risk management process are the main drivers of the proposed methodology. Each phase of the standard life cycle is then broken down with guidance for including security relevant activities. Key points in the process where Designated Approving Authority (DAA) decisions or interactions are required are specially indicated. Certifying authority decision points (as could likely result from risk analyses) are provided. These points indicate where trade-offs may be required or where the developer should check to see that the system is still being developed to meet requirements. If the process is followed and key certifier and accreditor decisions have been made and documented it follows that both certification and accreditation should be complete, from the management prospective. Viewing both from only management's perspective is not sufficient however. Therefore, detailed guidance on technical activities and product development during each life cycle phase is also provided. Contractor reviews, deliverable products, and schedules with resulting program office actions are covered, but the methodology concentrates on a test and evaluation (T&E) point of view as T&E is the life cycle activity most likely to be performed by purchaser (government) resources. The guideline also provides some experienced based information on pitfalls that may be encountered, some organizational responsibilities within the Air Force, and some tools or methodologies that can be used to aid in certification. Where and how to apply these tools in the life cycle, be they for verification, risk analysis, or testing, is provided. Unlike some other methodologies this one does not end with certification, in fact it does not end at all. The final action described is the reentering of earlier life cycle process phases when recertification and reaccreditation are required. The only final action that occurs in a system's security life cycle is its destruction or disposal without it having been replaced or updated. Complete detailed guidance for each life cycle phase, intended for the action officer level, is not provided. The nuances of each system development are such that detailed guidance for each possible action would fill volumes. This task is left to an entire series of other documents related to AFSSM 5010, two of which are AFSSM 5011 for applications software and AFSSM 5024 for program managers.

AFSSM 5011

Among this implementing series of documents is AFSSM 5011, Security Certification Guideline for Application Software [11], which provides the certifier with a checklist approach for certifying applications software. Individual checklists target software developed on either C2 or B1 TCB systems, meeting AFR 56-31 requirements, meeting DOD-STD-2167A requirements, or evaluating commercial-off-the-shelf software. The methodology is not as thorough as those mentioned elsewhere in this paper because it is intended for the system user community, not system developers. It does not assume the guideline user is either a system or security engineer or analyst. In fact, one of its main proposed users is the individual developing software on a

microcomputer for later use on a larger system. Its results should be added to those provided by the developing program office and presented to the operational DAA for final system accreditation.

AFSSM 5024

AFSSM 5024, Computer Security in Acquisitions [12], focuses primarily on the generation of system specifications for a host of security disciplines. It discusses each security discipline, e.g., computer security, TEMPEST, physical, etc., and provides appropriate Contract Data Requirements Lists (CDRL) and Data Item Description (DID) language that the program could require. The goal is to produce an appropriate Request for Proposal (RFP) that will lead to certification. The process doesn't stop with RFP release, but continues with guidance as to how the contract deliverables should lead to certification, much like the Mitre methodology, and accreditation. The guideline covers the timing of product delivery and the probable size of each, thus enabling planning for the effort required to review and manage each product.

ETA TECHNOLOGIES CORPORATION

The ETA certification effort [13] began after some of the affected systems (for munitions storage and management) were actually installed. The system's developers had begun certification itself much before, but this particular effort was performed by a contractor brought in during the full scale development phase. Rather than continue existing certification actions the contractor, ETA, chose to employ a step-by-step procedure that validated existing products and then continued with needed tasks which would compose the bulk of the certification. The process is composed of four phases closely related to the development life cycle. A series of 18 tasks, spread over the phases, are required. Most of the tasks, see Figure 2, occurred after the time the effort began. Relatively few tasks were required to validate previous work. The basic concept is to certify in stages, not all at once. This concept was also carried over to an accreditation methodology for the system, see Figure 3.

Certification Phases

Each of the certification phases were matched to the basic process of system development; define the system concept based on user requirements, translate these requirements into system development requirements (specifications, statements of work), validate that the system is being built to requirements through the review process, and finally test that the implementation meets the user requirements. The certification phases closely match the first four phases of the development life cycle (Figure 2). Certification during the production/operation phase is the responsibility of the user or maintaining agency and was not part of this particular effort. However, modifying the process should yield a subset of the tasks which the user can then use for recertification maintenance.

Task Phases

Each of the tasks consist of a purpose statement or goal. Actions required to meet goal follow, including the development or publishing of needed support documents. In this case, early tasks consisted of reviewing and validating previous certification activities. When needed activities had not been performed or products not developed, the actions were much like those of later tasks where the required actions were more performance oriented. Each task completion is documented by a validation letter or document to the certifying authority that the task was complete. The formal performance of each task is tracked by a standard set of action items. Each task action was

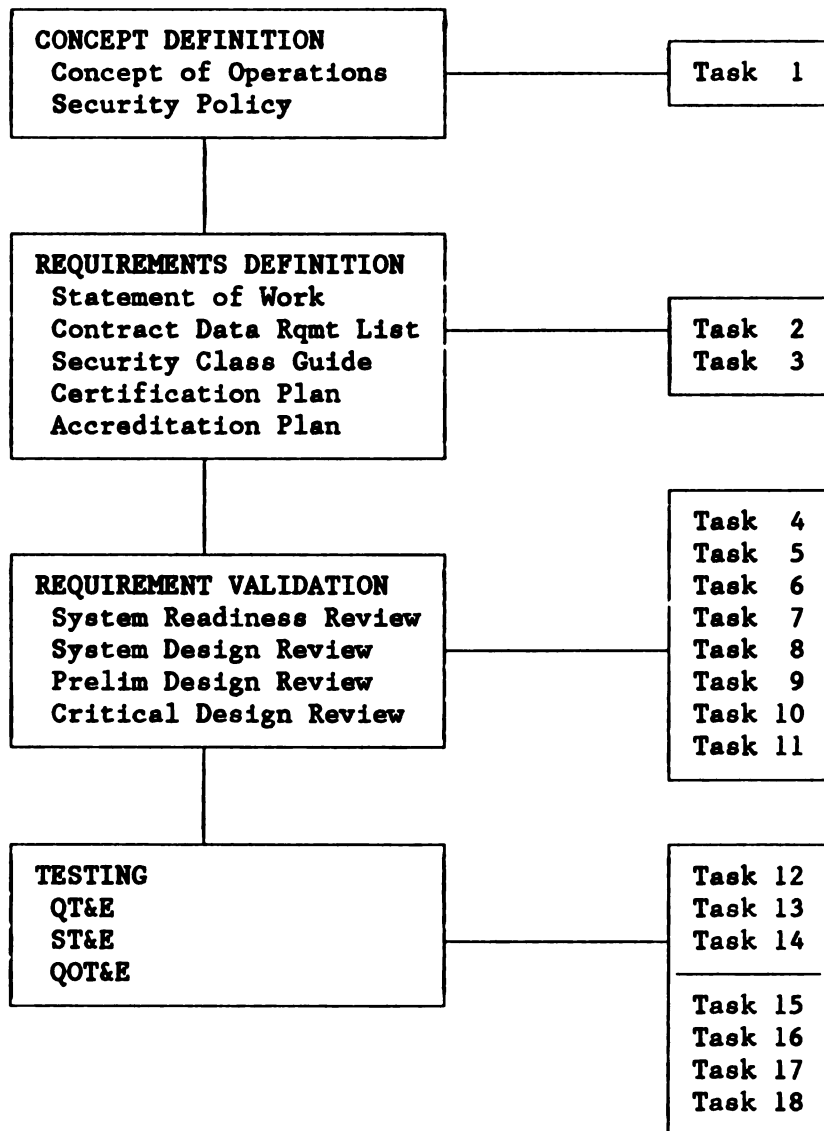


Figure 2. Certification Overview

researched as to its basis in requirements and current status. The status and subsequent actions are analyzed to determine if the task's goal has been met. The results of each task provided either the location of products which meet the goal or needed updates or modifications that would meet the goal. Each task completion requires coordination by every member of the system Certification and Accreditation Working Group (C&AWG), a subset of the larger Computer Security Working Group (CSWG). The C&AWG was set up to specifically work certification and accreditation issues within the overall security process which involved more widespread issues, e.g., TEMPEST, facilities design and implementation, etc. The final task action is certification authority approval of task completion. Thus, when all tasks are

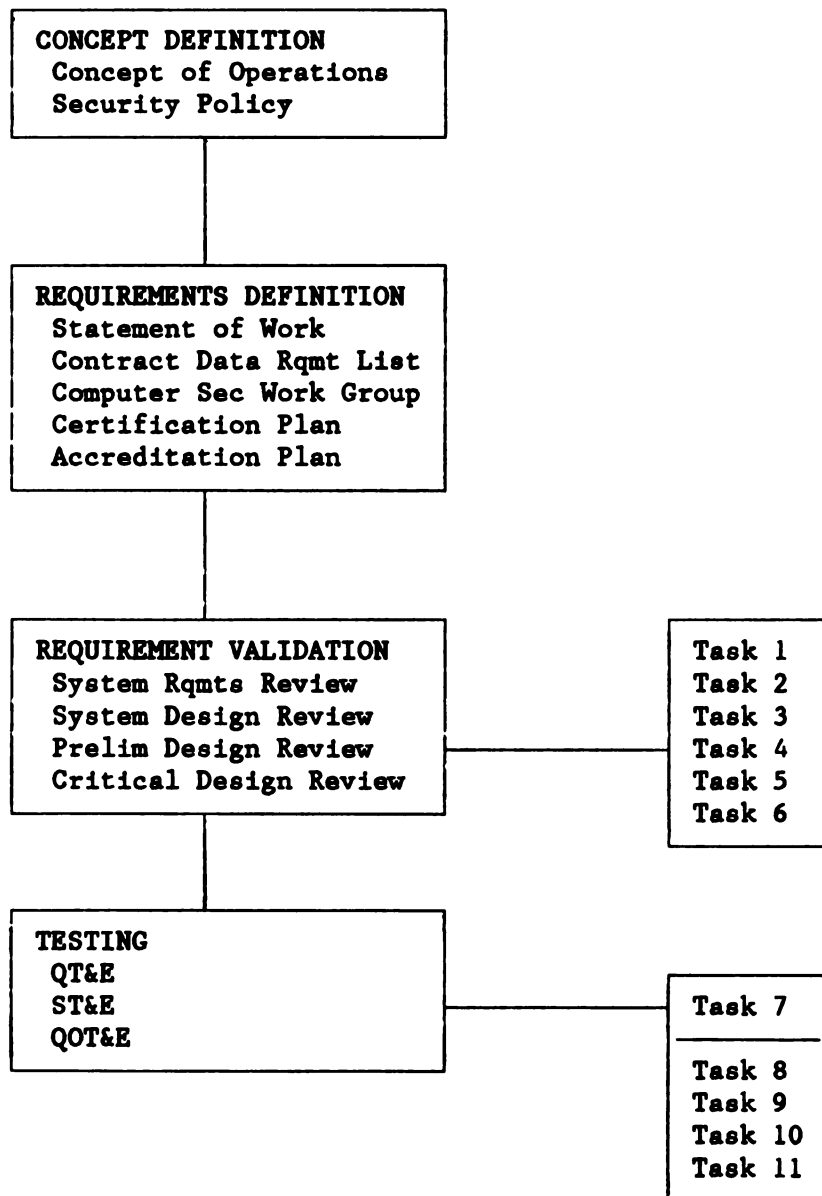


Figure 3. Accreditation Overview

completed, certification is complete without the certifier having to review the entire process at one time.

Observations

Although most of the later tasks involve validating development contractor actions or products, many involve actions and plans to be produced by the system's developing agency. In fact some are products to be produced by ETA themselves in their support role to the developing agency. The C&AWG coordination becomes important as this group is composed of major users of the system and representations from independent certification support agencies. This provides an extra degree of assurance that the certifier has not overlooked anything by being too close to the action.

Tracking the tasks includes producing briefing aids for each task to present each task's status to the certifier or at each C&AWG meeting. A typical task status appears at Figure 4. This particular task involves validating that appropriate security procedures are being developed according to SOW and user requirements. The actions required must ensure preliminary procedures (the system is still in the development phase) allocate security functions based on cost-benefit analyses done in risk analysis. Research and analysis of completed activities of needed actions have been completed and activities

Task 6: Validate Security Procedures

Develop preliminary security procedures based on the evaluation of the security requirements, in accordance with the SOW. Validate these procedures and evaluate the allocation of security functions between technical enforcement and procedural enforcement.

a. Action: Ensure preliminary security procedures adequately allocate security functions and features between technical enforcement (internal controls -- TCB, TCM, Trusted Processes) and procedural enforcement (external controls -- Physical Security, Communications Security, et al).

b. Documentation: Security Procedures Validation Letter.

c. Action Item Status:

- | | |
|----------------------|---|
| - Research | Complete |
| - Analysis | Complete |
| - Resolution | Complete |
| | Trusted Facility Manual Supplement. |
| | Security Features Users Guide Supplement. |
| | Users Manual Revision Needed. |
| - C&AWG Coordination | |
| - Certifier Approval | |

Figure 4. Sample Task Status

needed to resolve the action provided. The resolving activities involve modifying or supplementing vendor provided documents to meet the specific application and the production of a specific product by the development agency. As these are not yet done, C&AWG coordination has not yet begun nor has the certifier approved the task as complete.

At first glance task performance appears to match the development life cycle, but this is not necessarily so. In reality many tasks are being performed at once. For example, Task 17, Conduct Certification Readiness Review, is performed when a task is completed or at appropriate standard development reviews. Tasks 15 and 16 involve vulnerability and risk analysis and are constantly performed and will continue until the system is accredited.

Although much of this certification effort was performed when the system was already in operation, the process itself is easily applicable to a system not yet past concept definition. By fully matching the process to the development life cycle and matching tasks to schedule and performance responsibilities, the largest part of the developing agency's certification plan is provided. Minimal efforts, e.g., inserting the system's security policy, regulatory requirements, product specifics, etc., would complete the plan. For a relatively simple system the process is not overly complicated. For a more complex system, with multiple interfaces or embedded applications, other tasks may be needed and retroapplicability may not be possible.

ISSUES

As you can see, each of these methodologies, as well as many others, provide somewhat thorough but different views of the certification process. Their guidance is at such a level that the user must have some knowledge of the certification process as well as being an experienced system developer. The detailed guidance for each activity in each life cycle phase is being developed at many sources but is not currently available. Some topics which must be addressed include translating user requirements and mission needs into a system security policy (of which TCB policy is only a portion) and then deriving a user/program office produced CONOPS and maintenance policy from this policy. Specification, SOW, and RFP production is an area frequently addressed but how to evaluate responses to them are not. This includes both source selection activities and models or prototype developed during demonstration and validation. Configuration management for prototypes is usually less restrictive than for EMD systems, a situation that can impede certification, particularly for trusted products. Several approaches address certification during system development and early production and deployment when final, baselined products are available for test and evaluation. Security testing guidance, especially for TCBs, is available but not widely. Much work needs to be done in defining how certification is to be transitioned to users or maintainers. Incremental changes, patches, distribution and storage, maintenance interfaces, and a myriad of other issues arise during system operation that did not occur or were not a concern to the developer. Often the user is faced with not having the original certification methodology or the certification tools used. Certification must then be performed by another methodology with no assurance that results similar to the original will be realized. Any certification process that is used must consider the entire life cycle, since the requirement may be to enter the process at any point in the life cycle.

REFERENCES

1. NCSC-TG-004, Glossary of Computer Terms, 21 October 1988.
2. Department of Defense Directive 5200.28, Security Requirements for Automated Information Systems (AIS), 21 March 1988.
3. AFR 205-16, Computer Security Policy, 28 April 1989.
4. Pierce, Charles R., Standardized Certification, Proceedings of the 14th National Computer Security Conference, 1 October 1991.
5. Proposed Management Plan for Computer Security Certification of Air Force Systems, Draft Mitre Technical Report, RADC Project 4610.
6. DOD-STD-2167A, Defense System Software Development, 29 February 1988.
7. AFR 800-14, Lifecycle Management of Computer Resources in Systems, 29 September 1986.
8. MIL-STD-1521B, Technical Reviews and Audits for Systems, Equipments, and Computer Software, 4 June 1985.
9. AFR 56-31, Computer Security in the Air Force Acquisition System, (Draft), 9 May 1991.
10. AFSSM 5010, Computer Security in the Acquisition Life Cycle, (Draft), 3 June 1991.
11. AFSSM 5011, Computer Security in Software Development, (Draft), 23 Jan 1991.
12. AFSSM 5024, Computer Security Considerations in the Acquisition of Computer Systems, (Draft), 10 May 1991.
13. Technical Report, Combat Ammunition System-Base, SETA Contract F11624-88-D-0002, Delivery Order 6K-03, ETA Technologies Corporation.

TRUSTED DISTRIBUTED COMPUTING: USING UNTRUSTED NETWORK SOFTWARE

E. John Sebes, Richard J. Feiertag
Trusted Information Systems, Inc.
444 Castro Street, Suite 800
Mountain View, CA 94041

Abstract

The Distributed Trusted Mach Concept Exploration resulted in a design that extends the Trusted Mach design, to support transparent network communication between several nodes in a B3 trusted distributed system. A key feature of this trusted network communication is the use of existing network protocol software in a manner which allows this software to be untrusted.

Keywords: *Distributed systems, Trusted systems, Network Protocols*

Introduction

The Distributed Trusted Mach (DTMach) Concept Exploration¹ resulted in a design [1] that builds on Trusted Mach (TMach) to provide a trusted distributed system intended to meet the TCSEC B3 trust requirements [2]. TMach [3] [4] [5] is a trusted operating system being developed in conjunction with Mach, to provide Mach operating system services in a manner consistent with the B3 requirements. Mach [6] is a portable multi-programming, message-passing operating system being developed at Carnegie Mellon University.

This paper describes a part of the design of DTMach which deals with providing communication between the various nodes in a trusted distributed system. In particular, this communication is by means of the same trusted inter-process communication (IPC) used on a single TMach node, but transparently extended over the network to other nodes. A key feature of the design for distributed IPC is that existing network communication protocol software is used without modification; and this software is not included in the Trusted Computing Base (TCB).

This is an important result because a more usual approach to trusted networks is to include network protocol code in the TCB, frequently by developing new trusted protocols or by extending and re-engineering existing ones to meet trust requirements [7] [8]. For DTMach, however, the B3 trust requirements make these approaches difficult. Development or re-engineering can be a significant task, particularly in light of the requirements placed on B3 trusted code. Additionally, the development of B3 systems must include significant effort to minimize the size of the TCB by excluding from it non-security-critical functionality. For DTMach, network protocols represent a significant area of functionality that can be so excluded using the techniques here described. Therefore, an important part of the DTMach development will be integrating existing network protocol implementations into this architecture.

This paper first provides an overview of Mach and TMach, followed by a description of the relevant functionality of DTMach. Then we give the architecture and high-level design for this functionality, discussing the various alternative means to implement it.

¹This work was funded by Rome Air Development Center contract number F30602-87-D-0093/0008.

Mach, TMach, and DTMach

Both Mach and TMach consist of a kernel, which implements the basic abstractions of a multi-programming, message-passing system, and a number of system servers which use the kernel primitives to provide most of the conventional operating system services such as devices, directories, files, a name space, and so on.

Among the kernel abstractions are *tasks* and *threads*. The *task* is both an execution environment and also the basic unit of resource allocation. The *thread* is the basic unit of execution in the system; a task may contain multiple threads, executing with common access to the resources in the task's environment.

A task may communicate with another task by sending a *message* over a *port*. This message passing, or IPC, is the primary form of communication not only between tasks, but also between tasks and the kernel. A message is simply a typed collection of data that is sent on a port. A port is a channel for messages.

A task's ability to use a port is governed by possession of *port rights*, which can be sent in messages, in addition to message data. Among the port rights are the right to send a message over a port, and the right to receive a message from a port. There is only one receive right to a port, and the holder of that right is called the receiver of the port. There may be several senders for one port. Possession of a right to a port can also be the capability to create and/or send a right to that port, subject to some restrictions.

Both Mach and TMach use the client/server model of system operation. With respect to a given type of service, a task may be a client (a user of the service) or a server (a provider of the service). In order to obtain a service, the client task and server task communicate via ports; in other words, the port is the entry into a server task for obtaining a service. In Mach and TMach each system object is considered a separate service and is accessed via its own port. The port associated with a particular object becomes the representation for that object. For example, in TMach, the File Server is the task that manages files. For each file it manages, the File Server creates at least one port. It creates rights to send messages to that port, and sends those rights to each client task that requests and is allowed access to the file. The client task's representation of the file is its right to send a message to the port associated with the file.

Ports are therefore the fundamental means for accessing objects, and port rights are the fundamental means for controlling access to objects. The TMach security policy is implemented by controlling the creation and dissemination of port rights. This control is one of the main functions of the TMach kernel: it enforces the security policy by implementing security-relevant restrictions on the use of ports and the passing of port rights between tasks. The TMach trusted system servers also enforce the security policy by implementing access control on the objects represented by ports.

One of the important differences between Mach and TMach (besides the essential addition of trust features in TMach) is that TMach was originally designed to address the trust issues of a single Mach system running on an isolated machine. Mach, on the other hand, supports network communication between nodes, including IPC that is transparent across the network. Thus, one of the two essential purposes of DTMach is to unify the distributed IPC of Mach and the trusted IPC of TMach to make a distributed trusted IPC that can be the foundation of a trusted distributed system. The other main purpose is to adapt the TMach system servers to utilize DTMach IPC to provide true distributed service.

In distributing IPC, DTMach must enforce the kernel's policy on ports: because the use of ports is transparently extended over the network, their use must follow the same rules as local use of ports. Rather than altering the kernel to do this, the DTMach design calls for a new trusted server, the Network Server, to distribute IPC in a trusted manner. In this regard, DTMach follows Mach, which also has a Network Server. The remainder of this article describes the DTMach Network Server, and its use of existing network protocols.

Network Server Overview

This section gives an overview of the basic functionality of the Network Server, without going into particular design details. This basic function is distributing IPC by transmitting IPC messages between nodes, while maintaining the kernel's security policy on ports. The Network Server distributes IPC by holding rights to local ports on each node; as a result, it can associate ports on different nodes to create

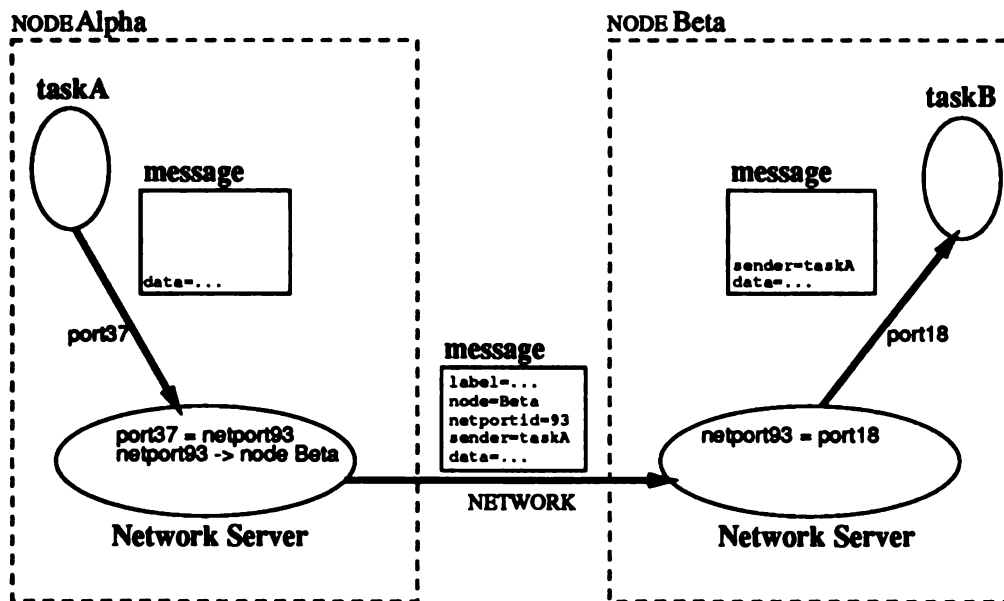


Figure 1: DTMach Message Scenario

a virtual circuit between tasks on different nodes. How this arrangement is set up and maintained it described in detail in [1].

The essence of DTMach distributed IPC is that it allows a task to send a message over a port—and the message will be delivered to a task on another node— without either the sender or receiver being aware of the fact they are on different nodes. This is accomplished by having a Network Server on each node in the distributed system; or to use the terminology of distributed systems, the Network Server is a distributed server with an instance on each node in the system. Each Network Server instance communicates with others via network communication protocols; and each communicates via local IPC with the kernel and with other tasks on its node.

The Network Server's role on each node is characterised by two essential strategems: first, the Network Server is the receiver of every local port with an ultimate receiver on another node; and second, the Network Server is a sender of every local port with an ultimate sender on another node. As a result, the Network Server receives every message bound for another node, and it can locally deliver any message that it gets from another node. Thus, the Network Server provides a global IPC service that transparently transmits messages between ports on different nodes.

This global IPC service typically follows the sequence of events illustrated in Figure 1:

1. A user task **taskA** on node Alpha sends a message to another task **taskB** on node Beta, by sending the message over a port **port37** that **taskA** assumes **taskB** is the receiver for. In fact, the port is networked, so the local receiver for port **port37** is actually the Network Server.
2. The Network Server receives the message on **port37**, already knowing that Beta is the node with the ultimate receiver for messages sent over **port37**; therefore Alpha's Network Server instance sends the message over the network to Beta's Network Server instance.
3. Beta's Network Server instance receives the message from from the network, already knowing that the local port **port18** is the destination for messages originating from Alpha's **port37**.
4. So, Beta's Network Server instance sends the message to the over **port18** to **taskB**, and does so in such a way that it appears the message was actually sent over **port18** by task **taskA**.
5. Finally, **taskB** receives the message from **taskA**, and neither of them knows about the intervention of the two instances of the Network Server.

In order to correctly forward and deliver messages, Network Server instances must have a mechanism for “knowing that Beta is the [receiving] node ... for port37” and “knowing that the local port port18 is the destination” as mentioned above. The mechanism is a set of mappings between local ports and global identifiers called net-port-ids. One such mapping is shown in the lower left of the figure as “port37 = netport93”. This indicates that port37, over which the message came, is associated with the net-port-id netport93; and this net-port-id is recognized by other instances of the Network Server on other such nodes.

Also shown in the lower left is another mapping,² between netport93 and node Beta; this node is the one which has the task that is the destination for all messages on port port37, and all other ports associated with netport93. Note that since a port may have multiple send rights, there may be several local ports which have the same remote destination; and furthermore, since these rights may be sent in messages across nodes, there may be several nodes with local ports all of which have the same remote destination. Each of these is associated with the same net-port-id.

As a result of consulting these mappings, the Network Server instance shown on node Alpha knows which other Network Server instance to send the message to, in this case that on node Beta. Before doing so, however, Alpha’s instance must add various kinds of control information to the message. One of these is the net-port-id, which Beta’s instance uses to determine how to locally deliver the message. So, when the Network Server receives the message on Beta, it consults a similar mapping between net-port-ids and local ports (shown on the lower right in the figure). In this case it finds that port18 is the local port over which to send messages originating from remote ports associated with netport93.

Other control information included in the message is the security label of the local port it was originally sent over, and various data about the sender. This sender information (schematically illustrated as “sender=taskA”) includes the user identity and security label or range of the sending task. This and other data are of vital importance to Beta’s Network Server instance, because it must convey the information to Beta’s kernel, as part of the message sent over port18. The kernel uses this information to mediate the reception of the message, in accordance with the security policy. The transmission and use of this security-critical data is one of the primary trust-relevant functions of the Network Server, in helping enforce the kernel’s policy on ports. In other words, the Network Server acts as the local representative to the kernel for the remote task that sent the message; and in order for the kernel’s policy to be enforced, the Network Server must correctly represent remote tasks.

Besides maintaining port mappings and transmitting security-critical data about each message, the other key function of the Network Server is to track the movement of port rights across nodes, and update and disseminate the changing port mappings that result from such movements. This enables the Network Server to continue to efficiently deliver messages, even as port rights move through the system.

Architecture

The main feature of the architecture of the Network Server is its partition into separate functional components. The reasons for this partition derive from the TCSEC architectural requirements. Primary among these requirements at the B3 level of trust is that of minimality, which mandates significant effort for the removal from the TCB of non-security-critical functionality. It turns out that a large part of the functionality of the Network Server is in fact not security-critical, and can easily be implemented outside of the TCB, in a component called the *NetProtocol Server*. Therefore, the first task in describing the architecture of the Network Server is to describe both the *NetProtocol Server* itself, and also how the Network Server interacts with it. Then, we can describe the further functional split of the Network Server into two components, the *NetMessage Server* and the *NetLine Server*.

NetProtocol Server

The DTMach NetProtocol Server (NPS) implements the network protocols used by the Network Server, including TCP/IP, among several others. If the network protocol software can be controlled so that it is unable to violate the security policy of the system and it cannot compromise the integrity of the TCB, then it is not protection critical and, in keeping with the TCSEC B3 requirement for minimality,

²Both of these mappings are needed, because both can change independently of one another.

can be removed from the DTMMach TCB. This is especially important, in light of the large amount of network protocol code and its intricate nature, which would make it a major undertaking to implement this functionality in adherence with the TCSEC requirements.

Therefore, the entire function of this untrusted server is to take IPC messages from the Network Server, and to create network packets from them (and the inverse); these packets contain the content of the message in a form ready to be sent over the network.

Since the NPS is untrusted, however, there is a set of issues concerning how it can be used by the TCB, while not effecting MAC or DAC, and while maintaining the integrity of the data it handles, particularly TCB data. In other words, the TCB must take measures that ensure the non-disclosure and integrity of the data given to the NPS. This is because of three factors:

- enforcement of both MAC and DAC depend on the integrity of the TCB MAC and DAC data that the Network Server puts in each network message;
- disclosure of TCB MAC and DAC data could result in undermining the enforcement of policy;
- improper disclosure of any data could itself constitute a violation of policy.

The general approach to integrity is for the Network Server to implement an end-to-end integrity check on each message. More details on this, and on the various alternative methods of accomplishing non-disclosure, are given in the last section.

The overall interface between the Network Server and the NPS is this:

1. The Network Server receives a message to be sent over the network, and embeds MAC information, DAC information, and integrity-checking data.
2. The Network Server sends this annotated message to the NPS, which breaks the messages into packets in whatever way is appropriate for the network protocol used.
3. The NPS sends the packets back to the Network Server, which sends the packets over the network.
4. The receiving Network Server instance passes packets to the NPS, which re-assembles the messages and passes them back to the Network Server.
5. When the Network Server receives a re-assembled message, it checks the integrity of the message to ensure that it was received without tampering. The intact MAC and DAC information is passed to the kernel so that it can perform its mediation of the message-receive by the intended recipient.

One last important feature of the NPS is that it is for the sole use of the Network Server. This is not to deny that user tasks may wish to use a service that implements network protocols—such a service would be provided by untrusted servers in many systems. However, the NPS is not such a service; rather it will not be available to any other system component besides the Network Server, so that the latter can use the NPS without any interference from other tasks.

In other words, the NPS is best conceived of as a private part of the Network Server, which is implemented as a separate, untrusted task for trust engineering reasons.

In any case, the NPS is not exactly the service that clients need. Although the NPS's set of protocols includes some that are generally useful, it also includes some that are not, and omits some that are. Therefore, one can envision other untrusted servers, for example a TCP server, a UDP server, and an IP server, and perhaps others for ISO protocols. In such cases, the NetLine Server (see below) would accept the traffic from these untrusted servers at various levels, multiplex it onto the network, and de-multiplex it based on packet labels.

NetLine Server

Besides separating out the functionality of the NPS, the Network is also broken into two further components, the NetLine Server (NLS), and the NetMessage Server (NMS). This separation arises from the fact that part of the Network Server's functionality—moving messages over the network—is unrelated to the functionality of managing IPC, which is the main function of the Network Server. Therefore, this additional, unrelated function is implemented in the NLS, while the NMS implements the bulk of the Network Server functionality as described in the rest of this report.

The separation of the NMS and NLS is only partly motivated by modularity. Certainly, since the functions are quite distinct, modular separation is possible. But another TCSEC architecture requirement

also comes into play— that of least privilege. The NMS requires special privilege from the kernel, but this privilege is not needed by the NLS. Likewise, the NLS's access to network devices constitutes a functional ability not needed by the NMS. If the two were implemented in one task, then the NLS code would have more privilege than it needs, and the NMS code would as well. Therefore, the separation of the NMS and the NLS increases compliance with the least privilege requirement as well as the modularity requirement.

The function of the NLS is, quite simply, to manage the network devices in the distributed system. Its access to these devices is through the use of other TCB components, such as the kernel and the Device Server. Of course, the use of these devices is simply to write packets onto them, and to read packets from them. Since these devices may carry multi-level data, the management of them must be done by the TCB, rather than by components which actually generate the packets, such as the NPS. Therefore, data from IPC messages of all labels flow from sending tasks through the NMS and then the NPS, to the NLS and over the network; and of course, the reverse flow happens as well.

The main trust-relevant task of the NLS is to label each out-going packet. This is necessary for preservation of the label of the data throughout the message transfer. The receiving NLS instance uses the label to ensure that for each packet, the the protocol service that handles it is actually permitted to handle information with the packet's label. Additionally, the NLS must implement an integrity check on each packet, to ensure that it has not been corrupted or damaged in transit— not least to ensure that the label on the received packet is the same as the label it was sent with.

In summary, the NLS is a trusted multi-level device manager which labels the data passing through it, to or from the devices. One further point of note is that the NLS may provide service to tasks other than the NetProtocol Server— for example, an untrusted TCP/IP server will certainly have packets for the NLS to put onto the network.

NetMessage Server

The NetMessage Server implements the main functionality of the Network Server— that of distributing IPC— rather than implementing the underlying network protocols (done by the NPS), or the management of the network devices and data (done by the NLS). In this respect its functionality is essentially similar to the Mach NetMessage Server. In fact, the DTMach NetMessage Server may be based on the Mach NetMessage Server, in many respects.

The overall function of the NetMessage Server can be summarized as follows. It must:

1. receive from local tasks all messages intended for remote destinations;
2. forward each such message to a remote NetMessage Server instance;
3. after receiving such a message from an originating NetMessage Server instance, send it to the correct local task as intended by the sender.

It is in (2) that the services of the NPS and NLS are used. At this level of functionality, the NMS must be trusted to correctly deliver messages, since incorrect delivery might violate security policy. Additionally, the NMS must also uphold security policy by performing these functions:

4. include in each message some information about the sender, which is necessary both for policy checks and correct delivery;
5. co-operate with the kernel by aiding in carrying out IPC policy checks;
6. provide data-integrity for the messages it sends via the NPS, in order to facilitate the exclusion from the TCB of the NPS (see above).

One additional architectural point is it might be possible to split out into an untrusted component much of the detail involved in (2) above, in accordance with the minimality and least privilege requirements.

Design

This section addresses various design issues of the Network Server previously raised. The high-level design of this IPC-related functionality is best given as a detailed description of message transmission. Many of the details of the design of the Network Server are described by explaining the way these three components— NMS, NPS, and NLS— work together to move a message from a client task on one node to another client task on another node. This is illustrated in Figure 2.

The figure shows two nodes, each with the three Network Server components, the kernel, and a client. The shaded box shows the TCB, which includes the NMS, NLS, and kernel of both nodes, together with the network. The dotted box encloses the components of the Network Server; the NPS is shown as instantiated at each level, as in one of the architectural alternatives described below. Each arrow indicates one step in the the journey of a message from the client on node Alpha to the client on node Beta.

The first step occurs when the client on Alpha sends a message over a networked port, and the message is received by the NMS instance on Alpha. Then, the NMS must annotate the message with a variety of data:

Label Foremost among these is the label of the port over which the message came. This will be needed by Beta's NMS instance, to ensure that MAC policy for the message is upheld. For example, Beta's NMS instance should ensure that when the message is sent on a port on Beta, the label of the port is the same as that of the Alpha port that the message was originally sent on.

User Profile Another kind of data added to the message is the user information about the sender: its user identity and label or range. These are used by the kernel for both MAC and DAC enforcement.

Net-Port-Id In addition to these data used for policy enforcement, Alpha's NMS instance must also include the net-port-id of the sender's port, so that Beta's NMS instance can correctly deliver the message to the proper recipient. This is security-critical data as well, since maintaining a secure state requires correct delivery.

Integrity Data Another very important function of the NMS is to protect the above data (and the message contents) from tampering by the untrusted NPS. One way to do this is simply to encrypt the entire annotated message. Alternatively, the NMS may compute a less computationally expensive message digest of the entire annotated message. Then only the message digest need be protected by encryption; the encrypted message digest would become the last component of the annotated message. Detailed treatment of encryption and data-integrity issues is given in [1].

The second step is when the NMS sends the annotated message to the NPS, along with some indication of which other NMS instance to send the message to. This indication will be protocol-dependent. For example, if TCP/IP were used, a TCP connection would be specified; setting up this connection would be part of initialization. The NPS computes a sequence of network packets which will convey the message to the requested destination.

The third step is when the NPS sends a sequence of packets to the NLS. The NLS must write each packet on the appropriate network device. Before doing so, however, the NLS must label the packet. Additionally, the packet must be protected from corruption in network transmission. At the very least, the label should be protected, although the integrity of the message as a whole may be of issue well.

The fourth step is when the NLS sends a packet to the network device's driver in the kernel. The device driver actually puts the data on the wire. Here the message begins to reverse its path through the system components. When the packet is available at node Beta, the driver has the data ready for the NLS.

The fifth step is when Beta's NLS instance gets a packet off of a network device. Then, the NLS must undo whatever integrity measures it applied on Alpha, and ensure that the packet arrived intact. The packet will be destined for some protocol-implementing task, such as an NPS. If the packet was intact, then it must examine the label and ensure that the port to the protocol-implementing task has the same label as the packet, which was the label of protocol-implementing task that created the packet. This ensures that the packet is moved between protocol-implementing tasks in accordance with MAC policy.

The sixth step is when the NLS sends a packet to the NPS, which accepts packets, and re-assembles them into the original message.

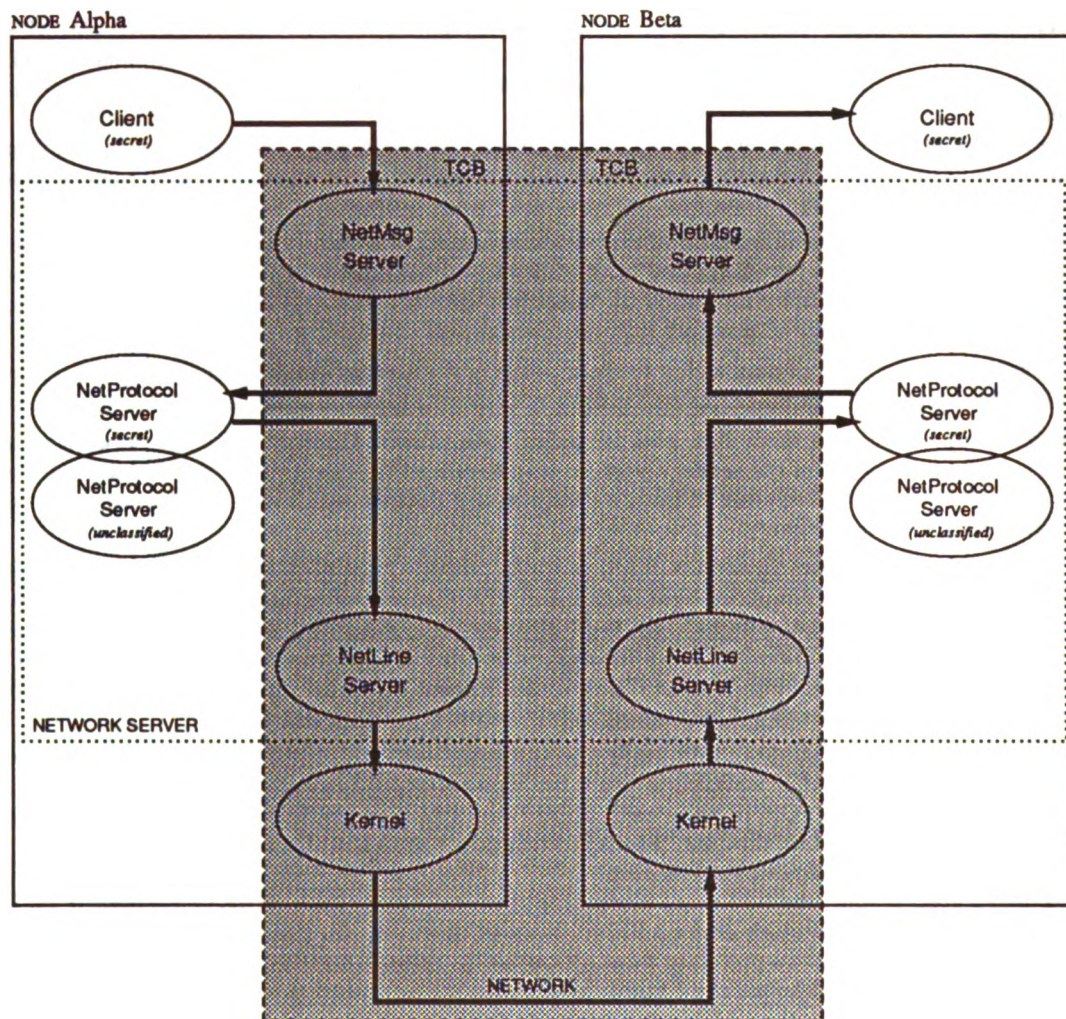


Figure 2: Message Transmission

The seventh step is when the NMS receives a reassembled message from the NPS. The NMS uses the various annotations put on the message by the originating NMS instance. First, the integrity check must be made. If this passes, then the NMS must verify that it is the intended receiver. Then, the net-port-id must be checked. The normal case is when the net-port-id maps to a local port. If the local label of the port is that of the message, then the message is sent over the local part.

The eighth and last step is sending the message to the receiving client. This is a special message send, however; the NMS uses its special kernel privilege (described in detail [1]) to inform the kernel of the user profile of the sender. The kernel uses this in its policy enforcement decisions. Assuming the message may be delivered, the kernel also uses this information to give the appearance to the receiver that the message was sent by a task with the identity of the original sender, rather than by the NMS.

A last note of this scenario concerns what happens when the normal case in step seven does not occur: when the net-port-id of the message does not correspond to a local port. In such cases, the NMS must determine what to do with the message. There are a variety of cases, but we describe one. The usual way that such a circumstance would arise is if the receive right to the networked port was originally held by a task on Beta, and then was passed to a task on another node. In that case, the Beta's NMS instance knows where to forward the message: to that other node. Beta's NMS instance should also send an administrative message to Alpha's NMS instance to inform it of the change of affairs.

These messages— forwarding messages, and administrative advisory messages— are purely between NMS instances themselves, rather than sent on behalf of IPC clients. As such, they are examples of kinds of messages sent in a protocol between NMS instances. There are other kinds as well, mostly pertaining to the NMS's attempts to keep net-port-id mappings reasonably up to date throughout the system. This protocol, for the Mach NMS, was described in detail in [9] and [10]. The protocol for the DTMach NMS will be based on this.

NetProtocol Server Alternatives

There are four different alternatives to the implementation of the NetProtocol Server, each of which is a different approach to the requirement to protect the data passing through the NPS. Each can be feasibly implemented, and would meet the requirements. However, we describe each because the decision involves several issues, and the results effect the architecture somewhat.

The four alternatives are based on two pairs of alternatives to the implementation of the NPS. One pair is two variations on the *multiple single-level* (MSL) NPS, in which data is protected from disclosure by being separated by label into different tasks. The other pair is two variations on the single NPS, an untrusted, system-low NPS from which data is protected by other means. Each of these has two variations, accounting for the four alternatives. These are illustrated in Figure 3.

The MSL NPS would consist of a set of single-level tasks, one for each label in the system; these are shown in Figure 3 in the upper two examples, as linked ellipses, noted as *MSL*. When using the transport services of the the NPS, the NetMessage Server would use the particular NPS task with the same label as the IPC message being sent. Likewise, when the NetLine server gets a packet that is part of an IPC message, it would send the packet to the NPS task with the same label as the packet. Although some complexity in the management of these MSL tasks is introduced by the approach, it is nevertheless a feasible alternative to the inclusion in the TCB of the NPS, which is probably insupportable because of the B3 minimality requirement.

To sum up, non-disclosure is achieved by the separation of labeled data, by using the existing kernel services: data is separated by label into distinct tasks, and the MAC enforcement of the kernel prevents this data from flowing in a way that violates mandatory policy.

In addition to protecting the data from improper disclosure, the integrity of the data must also be ensured. That is, the data must be protected from tampering, because such tampering could affect data which is used in MAC and DAC enforcement. There are two alternatives: full encryption³ (on the right side of Figure 3, with solid arrows), and message authentication (on the left side of Figure 3, with dashed arrows). Of the two alternatives, full encryption of the data ensures nondisclosure and integrity,

³It should be noted in passing that the architectural issues of including encryption in the system (including the encryption of classified data sent over unprotected networks) turned out to be entirely orthogonal to the encryption-related issues discussed here. The issues of network data protection are beyond the scope of this paper, but [1] gives details.

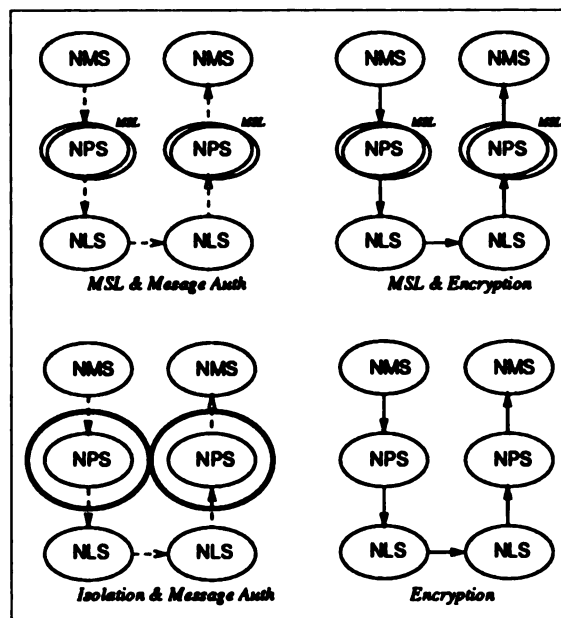


Figure 3: Four NPS Architectures

while message authentication of the data ensures integrity only. Full encryption is unnecessary because disclosure is already prevented by the use of single-level NPSs. Message authentication is sufficient to protect the integrity of the data as it passes through the untrusted NPS. Since message authentication can be less computation intensive, i.e., faster and more efficient, it is the preferred alternative for the MSL NPS.

For the single NPS approach, there are again two choices. One of them is to encrypt all traffic through the NPS, ensuring both non-disclosure and integrity. The other approach is to ensure integrity with message authentication, and to ensure non-disclosure by *isolation* of the NPS. That is, we ensure that the NPS can communicate only with the TCB—in particular, the NetMessage and NetLine servers—and not with any other other tasks. In other words, policy-violating disclosure is prevented by preventing the NPS from disclosing anything outside of the TCB. This results in a requirement (which we expect will be easy to fulfill) that the system be capable of enforcing such isolation.

Of the two alternatives, the isolation approach is preferable again because message authentication is more efficient than encryption. These alternatives are illustrated in the lower two examples in Figure 3. On the lower left, the thick ellipses around the NPS indicate isolation, and the dashed arrows indicate message authentication. On the lower right, the solid arrows indicate encrypted messages.

Having narrowed the four alternatives down to two, we can now say that we favor the isolated single NPS approach over the MSL approach, because of two observations:

- The MSL approach is somewhat more complex, because of the management of the multiple tasks.
- Both approaches offer similar levels of assurance.

Although the MSL approach may appear at first glance to offer higher assurance, it is nevertheless the case that both rely on the kernel's control over the NPS's IPC. In the MSL approach, the kernel is relied upon to enforce mandatory policy on the NPS's IPC; in the isolation approach, the kernel is relied upon to enforce isolation on the NPS's IPC, namely that it only communicate with the NMS and NLS. Since all kernel mechanisms may be regarded as having fundamentally the same amount of assurance, these two approaches have essentially similar assurance.

In each of the above alternatives, the data being protected cannot be disclosed or modified by the NPS. The worst that the NPS can do is deny service by not routing message, or mis-routing. Although there are no trust implications stemming from this denial-of-service possibility, there will of course be

measures taken in the interest of system robustness that will protect the integrity of the legitimate NPS in order to ensure system service.

Conclusion

We have described the security architecture and high-level design for the distributed IPC of DTMach. The important result shown by this description is that a trusted distributed operating system can be built without requiring invention or re-engineering of network protocols due to trust requirements. This result is particularly apt for DTMach: Mach includes the facility for network-transparent IPC, while TMach provides for trusted local IPC; DTMach therefore combines these two to provide trusted distributed IPC. That the network protocol implementation can be so separated is an indication both of the flexibility and modularity of the Mach architecture, and also of the fundamental way that security policy enforcement was incorporated into TMach.

One further positive result for DTMach IPC is that in combining these two essential features of Mach and TMach, neither was changed in any fundamental way. That is, the Mach approach to IPC over the network was augmented to deal with trust issues, but not changed. Likewise, the TMach kernel's enforcement of the security policy on ports was unchanged, save for addition of an interface for the Network Server to provide information from remote nodes. Thus, the port access control is still centralized in the kernel, with the Network Server acting in a supporting role; and the network management for IPC is centralized in the Network Server, without the kernel having to be directly aware of events on other nodes.

Finally, DTMach IPC provides a straightforward base for the implementation of trusted distributed servers which provide operating system features in a truly distributed, network-transparent way. Most of the trust issues of these trusted distributed servers derive from distributed database issues and/or particular TSCEC functional requirements (e.g., audit), rather than any concerns over security arising from network communication and IPC.

References

- [1] Trusted Information Systems, "Distributed Trusted Mach Concept Exploration Final Report," Rome Air Development Center, 1990. TIS Rep. 374.
- [2] National Computer Security Center, "Trusted Computer System Evaluation Criteria," DoD 5200.28.STD, December 1985.
- [3] M. Branstad, H. Tajalli, F. Mayer, "Security Issues of the Trusted Mach System," Rep. 138, Trusted Information Systems, January 1988.
- [4] M. Branstad, H. Tajalli, "Security Policy for the Trusted Mach Kernel," Rep. 179, Trusted Information Systems, September 1988.
- [5] M. Branstad, H. Tajalli, F. Mayer, D. Dalva, J. Graham, "Access Mediation in Trusted Mach," Rep. 203, Trusted Information Systems, March 1989.
- [6] Avadis Tevanian, Jr. and Ben Smith, "Mach: the Model for Future Unix," *Byte*, November 1989.
- [7] D. D. Schnakenberg, "Applying the Orange Book to an MLS LAN," Proceedings of the 10th National Computer Security Conference, September 1987.
- [8] Greg King, "Considerations for VSLANTM Integrators and DAAs," Proceedings of the 13th National Computer Security Conference, October 1990.
- [9] R. D. Sansom, *et al*, "Extending a Capability Based System into a Distributed Environment," *Communication of the ACM*, February 1986.
- [10] R. D. Sansom, "Building a Secure Distributed Computer System," Carnegie-Mellon University Rep. CMU-CS-88-141, 1988.

TRUSTING X: ISSUES IN BUILDING TRUSTED X WINDOW SYSTEMS
- OR -

WHAT'S NOT TRUSTED ABOUT X?

Jeremy Epstein
TRW Systems Division
1 Federal Systems Park Drive
Fairfax, Virginia 22033-4417
(epstein@trwacs.fp.trw.com)

Jeffrey Picciotto
The MITRE Corporation
Burlington Road
Bedford, MA 01730
(jpicc@mbunix.mitre.org)

Abstract

Keywords: Graphical User Interfaces, X Window System, multi level secure, industry standards.

The MIT X Window System¹ (X) has become a de-facto windowing system standard that is widely used throughout the computer industry. In many ways X is as important in the 1990s as standard operating systems were in the 1980s. Just as trusted versions of UNIX² operating systems (from C2 systems such as Gould's UTX/32S to B2 systems such as AT&T System V Release 4/ES) are critical to bringing trusted systems into widespread use, trusted versions of X are necessary in order to make the full power of those trusted systems available to users operating in today's workstation environments.

Adaptation of commercial systems to trusted systems generally involves tradeoffs between functionality and trust. X is no exception to this rule. Most commercial multi-user systems (such as UNIX and VMS³) implement mechanisms that enforce various security policies, such as access control and privilege policies, although those mechanisms are often relatively primitive (e.g., permission bits and super-user in UNIX). In contrast, X was explicitly designed to avoid enforcing any policies and, in fact, provides many mechanisms that tend to promote the sharing of data and resources among X applications. As a result, the tradeoffs between trust and functionality are far greater for X than typically encountered in operating systems.

This paper surveys the issues and outlines various solutions to problems encountered in designing and building trusted X systems. The paper focuses on issues that appear both at the B1 and B3 levels of trust specified in the Trusted Computer System Evaluation Criteria, and in The Security Requirements for System High and Compartmented Mode Workstations.⁴

1 Introduction

In the past few years, the X Window System has become the de-facto industry standard windowing system. As the use of X proliferates and vendor application support for X rises, the trusted computer systems user community will increasingly demand X for use on their trusted systems. There is, therefore, an immediate and significant interest in the security implications of running X. The most visible example of this phenomenon is the Defense Intelligence Agency's (DIA) Compartmented Mode Workstation (CMW[10]) program. Of the vendors currently under evaluation by DIA for a CMW rating, all have indicated their intent to use X as the basis for their trusted windowing system.

The X philosophy promotes cooperation among applications, including the sharing of data and resources. This is in fundamental conflict with the aim of trusted systems which requires some degree of isolation. The primary goal in building trusted X systems is to retain as much of the X functionality as feasible while providing the required degree of trust. The functionality goal is often stated as "well behaved clients should run unchanged."

¹X Window System is a trademark of the Massachusetts Institute of Technology.

²UNIX is a registered trademark of AT&T.

³VMS is a trademark of Digital Equipment Corporation.

⁴The TRW portion of this work is sponsored by the Defense Advanced Research Projects Agency under Contract No. MDA 972-89-C0029. The MITRE portion of this work was internally sponsored by MITRE's Information Security Center. Reproduction of this paper is permitted without charge except if copies are sold.

In this paper, we first describe the architecture and philosophy of X. Next, we describe the requirements for trusted X systems. We then survey the security issues, presenting various solutions and describing the functionality required both by different TCSEC levels [9] (specifically B1 and B3) and by the CMW requirements [10]. Finally, we describe some of the ongoing work in this area.

Note that this paper does not provide a cookbook solution to the problems of trusted X. Rather, it describes the issues that are critical to balancing the needs of X functionality and trust. Furthermore, we do not address assurance issues such as modeling or testing; the scope of this paper is limited to consideration of the impact of required security mechanisms on the functionality provided by X.

Throughout this paper, use of the term "X" refers specifically to the MIT X Window System, while "TXS" refers to any trusted X system.

2 X Architecture

The X architecture is based on the client/server model of distributed computing. As shown in Figure 1, the *X server* manages the screen(s), keyboard, and pointing device (typically a mouse).

X clients and the X server communicate via the X protocol [1]. Clients send *requests* to the server over a bi-directional communications channel using any reliable byte-stream protocol (for example, TCP/IP or DECnet), and receive *events* and *responses*. Errors are a particular kind of response. Protocol requests are typically asynchronous, since most of them have no reply. Protocol requests include administrative requests, requests to create and destroy resources (defined below), and drawing requests.

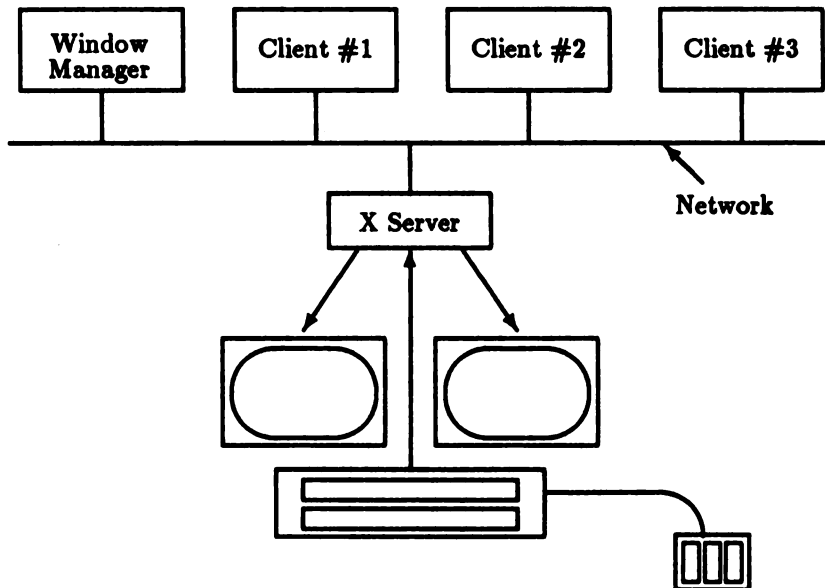


Figure 1: X Architecture

The X server manages *X resources* on behalf of the clients. Resources include windows, pixmaps, fonts, cursors, graphics contexts, atoms, and properties. X resources are data containers created by clients. The lifetime of a resource is generally (but not always) tied to the lifetime of the client that creates it. Resources are referred to by resource IDs that are associated by clients (and not the server) with the newly created resource. In addition to client-created resources, there are several global resources (such as the search path for fonts and the keyboard and pointer characteristics) that are created by the server when it is first started, and that clients may change but not destroy.

The X server manages resources in a manner analogous to how operating systems manage files. However, in a traditional file system, files are opened and subsequent operations use a file handle or descriptor. This allows access control to be checked only when a file is opened. In contrast, each X protocol request refers to required resources via their resource IDs. As will be seen later, this means that access control must be enforced on every protocol operation.

X clients can generate protocol requests directly. However, the Xlib library (described in [2]) provides a slightly higher level view of the protocol, including a subroutine interface that provides generation of the required byte stream for each protocol request, and some abstractions. Applications are more commonly written using even higher-level abstractions, such as a toolkit (e.g., Xt, described in [3]) and a widget set (e.g., Athena, or OSF/Motif⁵ [4] [5]). All of these libraries and widget sets are simply abstractions built on top of the protocol. Consequently, their use is invisible to the server.

X has no concept of privilege, and a minimal notion of protection. Protection is provided at connection time only. The X server maintains a host access list which identifies those computers from which connections will be accepted. In addition, an optional authentication mechanism allows the server to demand some form of authentication from the client (e.g., an MIT magic cookie or a Kerberos [6] authentication ticket). Once a client has connected to the server, it may perform any request, including a request to turn off authentication for clients that attempt to connect in the future. Clients can also directly impact other clients (e.g., by killing them), although such behavior is considered undesirable (see [7]).

Management of windows on the screen is performed by a *window manager*. There are many existing window managers, each of which provides a different look-and-feel. Because there is no notion of privilege in X, the window manager is simply another client. The conventions described in [7] are used to define an environment where “well-behaved” clients can interact cooperatively.

The MIT X Consortium distributes the source code for X free of charge. The release includes a *sample server*, widget libraries, window managers, and other clients which work on many systems.⁶ The Sun version of the X sample server is about 90,000 lines of C source code.

3 What is Trusted X?

A Trusted X System (henceforth referred to as TXS) typically involves an X Window System appropriately modified to provide functionality at multiple sensitivity labels. Specifically, a TXS should allow untrusted clients at different sensitivity levels to interact only in accordance with its security policies. TXS security policies usually include a mandatory access control policy (MAC), frequently a restrictive version of the Bell-LaPadula policy [14], a discretionary access control policy (DAC) and, in some cases, an information labeling policy [10].

The most easily implemented solution to building a TXS is to completely isolate clients from one another. This solution suffers from several rather severe problems:

- In order to fully isolate clients, other information channels must also be closed. Some of these channels (e.g., one client signaling to a second client by generating a series of expose events) are extremely difficult to close this way.
- Cut and paste no longer works. In X, unlike other windowing systems, cut and paste is a client-to-client operation, with the server simply acting as a passive intermediary. Client isolation prevents inter-client communications and therefore prevents successful cut-and-paste operations.
- Distributed applications no longer work. X’s model encourages development of distributed applications, where processes running on several computers in a network may work together on a problem. For example, a weather system might use a workstation to handle the menu processing, while a supercomputer performs the computation and makes the X requests to display the results. While these applications can be rearchitected, it is undesirable to prevent this programming paradigm.

Variations of the complete isolation policy have been proposed (e.g., the LINX project from Sun Microsystems) which provide limited interaction between clients. Other systems provide MAC and DAC policies to allow controlled sharing among clients at the same sensitivity level. Such compromise solutions tend to alleviate many of the difficulties associated with total client isolation, nevertheless, these solutions still curtail X’s flexibility and functionality beyond what is required by existing security requirements.

For this reason, this paper presents the issues and selected solutions that arise in a maximally flexible trusted X implementation. That is, in a trusted X system where the only restrictions imposed are those that are necessary and

⁵OSF/Motif is a trademark of the Open Software Foundation.

⁶Vendors can, and do, enhance the sample server (or reimplement it entirely) to suit their competitive needs. As previously noted, there are also various toolkits and window managers available.

sufficient to enforce the required policies. Solutions that impose additional constraints merely for expedience (e.g., total client isolation) are not considered.

Unfortunately, because X provides no rules and imposes no constraints on how the X protocol may be used, any change can potentially break existing X clients. Thus, the choice of what mechanism is used to enforce a particular policy must be made with extreme care. Our goal, then, is that well-behaved single level clients should run as they did before, without modification. Specifically, any changes to the X specification that would break commercial off-the-shelf software are considered undesirable.

One of the key difficulties in building a TXS is the lack of window system-specific TCSEC interpretations. While the CMW requirements [10] specify many of the characteristics of a multi-level secure windowing system, we are not aware of any National Computer Security Center-sanctioned criteria effort in the windowing area. Thus, there has been no official acceptance of the authors' interpretations of the TCSEC with respect to window systems.

4 Security Issues

In this section we describe some of the security issues associated with the X Window System, and present some solutions. We also explain which solutions are appropriate for B1, CMW, and B3 implementations. (We specifically omit B2 because we feel that the issues are adequately addressed by focusing on B1 and B3.)

4.1 Authentication

Authentication is the most obvious security problem with X. X provides a host access list. Any client on a remote host listed in the host access list can connect to the X server. The mechanism enforces no policy based on the identity of the user on whose behalf the remote client is operating, nor on the identity of the user logged into the local host. Once the client has connected to the X server, it can make any X protocol request it chooses (e.g., it may destroy arbitrary windows or lock the server). In hostile environments (e.g., university campuses) this X feature allows students to send requests to other servers to spy on or interfere with other users.

The X server does provide the "MIT magic cookie" authentication, that uses a secret shared between clients and the server. The magic cookie relies on the underlying operating system to store the secret, which is then passed (in clear text) from the client to the server. Access to the server is therefore governed by access to the operating system-provided storage container. The X server also includes hooks to allow additional authentication methods.

Fortunately, the general authentication problem has been the subject of a great deal of research and is a well-understood problem. In X, the most common solution is to use Kerberos [6]. Another method is to require the network to perform authentication, such as by having a name server which mediates access to the TXS server (as in the TRW TXS), or for systems that implement DNSIX [13], embedding the desired constraints in the Session Request Control Module (SRCM). In each of these solutions, authentication and identification is supported down to the granularity of a particular user on a particular remote host.

Authentication is an issue at all TCSEC levels and for CMWs. Use of Kerberos, or a name server is appropriate for B1, CMW, and B3. DNSIX (with appropriate constraints enforced by the SRCM) is additionally acceptable for CMWs.

4.2 Privileges

X lacks any notion of privileges. There is no mechanism to limit the X functionality available to clients on a per-client basis. As a result, all clients are treated equally and all are capable of performing any X function. Thus, for example, the window manager operates just as any other client, although it is explicitly manipulating other clients' resources. The Inter-Client Communication Conventions Manual (ICCCM) [7] specifies certain protocol requests that should only be used by window managers, however, since these are only conventions they are not enforced, and are sometimes ignored by clients.

Implementing a privilege mechanism in X raises several issues. These include:

- What privileges are necessary and reasonable to permit adherence to the least privilege principle within a window system. Depending on the granularity of privileges selected, it is feasible to define anywhere from a handful to several dozen TXS privileges. For example, changing the keyboard mapping (which affects the

meaning of the keys) is typically a privileged operation, but changing the font search path might not be. There is currently no agreement among vendors on what an appropriate set of privileges might be.

- How are these privileges communicated from the clients to the server. Several models exist (e.g., privileges are communicated once during connection startup and remain in force throughout the life of the client, or privileges are communicated by the underlying operating system with each X request). Currently there is no accepted way for a server to determine the privileges associated with a particular client.
- How are these privileges interpreted in a networked environment? Because there is no generally-accepted notion of domain-specific (e.g., TXS) privileges, each host may choose a different set of privileges. How a TXS might interpret these privileges, or map them into a common base set, is not well-understood.
- Finally, systems where clients enable and disable privileges (privilege bracketing) cause problems for a TXS because TXS's almost invariably buffer requests (to improve performance over networks). Indeed, the server and the standard libraries (including Xlib, Xt and widget sets) all implement buffering for requests, replies and events. How privileges are correctly maintained with buffered protocol elements is an issue that must be addressed.

At B1, privileges are not a major issue, as a single privilege is sufficient. For CMW and B3 (where adherence to least privilege is a greater concern), more sophisticated schemes are needed. CMW systems currently under development are implementing a wide range of solutions, from a single privilege passed only at client connection time, to fine-grained privileges passed with every protocol element. The divergence between these implementations, and the lack of consensus on a network privilege representation, are indicative of current uncertainty of the most appropriate model to adopt. We believe that until the problem is better understood, separate privileges should be defined for each class of X operation, and that each protocol element should be individually tagged with privileges (if the underlying transport layer supports this). Such an approach meets the least privilege requirements and does not impose limitations that may later become an impediment to the development of trusted applications.

4.3 Mandatory Access Control

X provides unlimited sharing of resources between clients. A window created by one client may be drawn in, or deleted by, any other client. This model simplifies the implementation of distributed applications. However, it is unacceptable in a trusted system.

X maintains two general classes of objects that differ only in their intended use. Local resources refer to those resources that are usually created, manipulated, and destroyed by a single client. Global resources refers to those resources that are intended to be shared among multiple (or all) clients. As described below, a flexible TXS generally treats the two types somewhat differently.

4.3.1 Local Resources

Enforcing mandatory access control in a TXS is relatively straightforward: each local X resource must be labeled with a sensitivity label, and mediation of a client's access to a resource can be performed in accordance with the Bell-LaPadula model based on a comparison of the client's and the resource's labels.

Although most TXSs treat local resources as described above, it is worth noting that in X the simple act of reading from, or writing to, a resource may cause events to be generated and sent to other clients (e.g., the creator of the resource). Thus an arbitrary read-down policy contains an information channel in that reading a window, for example, can cause the creator of that window to be notified under certain conditions. More seriously, write-up must be entirely prohibited because a client can detect whether a given resource exists by the results of the write-up.⁷ Since clients choose resource IDs, it is possible to use write-up as a broad signaling channel.⁸ For this reason, most TXSs typically support read and write equal, and a slightly constrained read down policy on their local resources.

⁷We believe, and industry consensus supports the view, that allowing write-up but always returning an error (or never returning an error) for the operation is of negligible utility in X.

⁸Consider a high client A that creates resources n1, n2, n3, ... Then low client B attempts to write to A's resources. By noting which requests give an error, B can detect which resources exist. Since A chooses the values n1, n2, n3, they can be used as a binary flag. Other, more sophisticated, schemes based on the same principle can be implemented that provide significantly higher bandwidth. If the server always returns an error for write-up, then this channel disappears, but the value of write-up is virtually eliminated.

4.3.2 Global Resources

Global resources are often treated either by polyinstantiation or by restricting access via privileges. The former technique is generally used only with resources that are not represented in some fashion to the user. If the resource is represented to the user, as is the case, for example, with pointer location, then polyinstantiation is too confusing, and the resource may be protected by requiring a client to possess a privilege in order to write to the resource.

Since most clients do not need to change the values of many global X resources (such as the keyboard mappings (e.g., QWERTY or Dvorak), keyboard characteristics (e.g., repeat rate), pointer characteristics (e.g., acceleration rate), and the host access list), requiring a privilege or making them fixed values does not significantly impair functionality. Other global resources (e.g., font path) can safely be polyinstantiated without unduly confusing the user. While some implementations might allow polyinstantiating the keyboard mapping, we believe the user confusion would be far too great.

Some global resources are frequently shared by multiple untrusted clients, yet do not fit well with polyinstantiation. Such resources include the root window (which covers the entire screen) and the default colormap (shared by most clients).

The root window is shared in several ways, such as setting its background, selecting its cursor, and attaching properties (arbitrary data values) to it, which can then be read or modified by other clients. Polyinstantiating the root window works to some extent, but several problems remain. For example, the window manager places properties on the root window to inform clients of icon sizes. If the root window is polyinstantiated, then the window manager must place the property at all levels, even though it cannot know ahead of time what the levels are. Polyinstantiating the background pattern and color is useless. Therefore, privilege seems to be a more workable solution for sharing the root window.

The default colormap is created when the TXS server starts, and initially contains black and white only.⁹ Clients then fill in colors as needed. However, clients can see and/or modify the entire colormap. Solutions include performing MAC on individual colormap entries or creating a fixed palette from which any client can select, thus treating the display as StaticColor or TrueColor. Note that the former solution introduces a covert channel pertaining to the number of unassigned colormap entries in a particular table.

At B1 and CMW, enforcing a standard Bell-LaPadula mandatory access control policy on local resources, with either polyinstantiation or privilege protection on global resources suffices. Read-down and write-up can be allowed by noting the existence of the covert channel. At B3, the same fundamental solutions apply, except the constraints must be somewhat tighter: typically a read and write equal policy is adopted and, when polyinstantiation is not a viable solution, global resources are forced to be static or are strictly protected for access by privileged clients only.

4.4 Discretionary Access Control

The issues surrounding discretionary access control (DAC) fall into two categories. First, if a TXS server is accessible to a single user at a time (e.g., on a standalone workstation) then DAC is unnecessary since all resources belong to the same user. However, some in the trusted X community believe that clients belonging to the same user should be able to protect their resources even from each other. This is based on a belief that X resources are more akin to data structures in a program (which the program can protect) than to files in a file system (which the program cannot protect).

Second, if clients operating on behalf of different users can simultaneously connect to the TXS server, what form of DAC must be implemented? Are permission-bit equivalents sufficient? Are access control lists (ACLs) necessary? If ACLs are necessary should they apply on a per-client or per-user basis? Are read, write, and execute permissions necessary, or should more window system-specific permissions be devised that logically address the functions that clients may apply to X objects? Finally, since TXS resources are ephemeral, and no existing clients expect DAC constraints, the default DAC value must be carefully crafted to support backward compatibility while providing some measure of security.

For B1 and CMW, permission bits (minimally read and write) indicating users' abilities to access particular resources are sufficient. At B3, a user-based ACL scheme is required. At any level, however, it is likely that per-client DAC (ACL or permission bits) will prove useful to future security-cognizant applications. Furthermore, TXS developers have generally agreed that partitioning particular attributable permissions into more than just read and

⁹This discussion applies to so-called PseudoColor displays, which are the most common type of color displays.

write, provides valuable functionality that will help security-cognizant applications perform more controlled sharing of resources.

4.5 Object Reuse

Object reuse is not a major issue in X. The designers were generally careful to specify the initial contents of X resources, and in those cases where initial contents are not explicitly specified, the X specification states that the contents are undefined (e.g., the creation of pixmaps and colormaps). Thus, in most cases, specifying initial values should not affect the operation of existing clients.

There is one case where the need to address object reuse affects the basic X functionality: the creation of windows having no specified background pattern. When such a window is mapped (i.e., made visible on the display), the X specification states that the window inherits the content of the screen enclosed within its boundary. Thus, for example, if a window, A, with no background, is mapped in such a fashion to overlay an existing window B, the contents of B would, in effect, be copied into A.

For B1, CMW, and B3, in all cases where the X specification states that initial values are undefined, the initial values must be set to some known value. In addition, the creation of windows where no background pattern is specified must be addressed. Typically this is done by limiting this functionality to trusted applications possessing appropriate privilege.

4.6 Secure Networking

The X protocol requires a reliable bi-directional byte stream as its transport layer. For a TXS, the transport layer must also be trusted, and must provide several special features. Specifically, the network must provide the message receiver (the TXS server) the ability to determine the security-relevant attributes of the message sender (the TXS client). These attributes must minimally include its sensitivity level. However, they may additionally include information pertaining to privileges, DAC, and for CMWs, information labels.

While the issue of trusted networks is outside the scope of trusted X, it is important that system engineers ensure that the network supports the functionality required by their implementation of trusted X. Thus far, no known protocols support the functionality required by a maximally flexible trusted X implementation. Work towards developing such protocols is ongoing in industry-sponsored groups. No firm results are expected in the immediate future.

4.7 Visible Labeling

In a TXS, the TCSEC is typically interpreted to require some sort of labeling of windows. Because windows can be arbitrarily nested, typically only top level windows (which enclose all child windows) are labeled. This approach is justified on two grounds: (1) labeling all X windows would lead to a visually incomprehensible screen, and (2) X defines a window as simply a data structure in the TXS, whereas the labeling definitions apply to windows as defined from a user perspective. From a user's perspective, top-level X windows are precisely those that should be labeled.

Pop-up windows are a particular problem in TXS. X clients can state that a window is a pop-up to avoid the overhead of window manager tracking (i.e., so the special borders and other "window dressing" window managers usually apply to windows is not applied to menus, dialogue boxes, etc). However, a client can claim a window is a pop-up and then uses it for any purpose it desires (e.g., a terminal emulation window). In this way, the client can very easily avoid visible window labeling. While solutions exist to label even pop-up windows, they are inelegant and have somewhat poor performance.

CMWs are subject to specific visible labeling requirements. These can be easily met in a variety of ways, usually by modifying a window manager to provide the necessary labeling in the same manner it already provides its existing functionality. These window managers can be further enhanced to properly address pop-up windows using functionality provided by unmodified X servers. For B1 and B3 systems, appropriate visible labeling policies are not immediately obvious. Alternatives are discussed in [11].

Spoofing of window labels appears to be unavoidable in a windowing system except by enforcing a tiling policy. Because tiling is generally unacceptable to users, further research is needed in this area.

4.8 Trusted Path

Trusted path in TXS might appear to be a non-issue: the TXS need only rely on the system-provided trusted path mechanism. There are two problems with relying on a non-windowing trusted path mechanism: it destroys the uniformity of the interface, and it may not provide needed functionality.

A trusted path implementation which bypasses a TXS will write directly on the user's screen, and not within a window. Any data which is on the screen is destroyed. Thus, once the trusted path operation is complete, the screen must be refreshed. While not a horrible problem, this is at least undesirable.

More importantly, a special trusted path client is needed for a TXS to provide X-specific functionality. For example, CMWs are required to provide a mechanism to determine the sensitivity and information labels of a window from the trusted path. This requires that the trusted path be implemented as some sort of an X client (or at a minimum, be highly integrated with X), so it can detect which window the user selected. Other required TXS trusted path functions, such as changing the input information label, require the same degree of integration with X for similar reasons.

A final major issue with a TXS trusted path is what to do with other clients while the trusted path is being used. Some implementations allow them to continue generating output, while others refuse any operations. The key issue is whether a client could seek to "hide" itself from the trusted path.

For B1, trusted path is not an issue. For CMW and B3, some notion of a TXS trusted path is required. For the reasons given above, most CMW systems either implement a special trusted path client, or integrate that functionality into a window manager.

4.9 Auditing

As a part of the TCB, a TXS server must audit certain actions. Auditing within a window system requires careful thought. The primary issue is that the NCSC auditing guidelines [12] are inappropriate for a TXS system. The CMW auditing requirements are very similar to, if somewhat more specific than, the TCSEC auditing criteria. As such they suffer from the same problems: both were written before the advent of window systems, and as a result, neither address the particular issues associated with window systems. The issues lie in two main areas: which actions must be audited, and how to identify and characterize those actions in a useful manner.

The first issue (which events must be audited) can best be explained using the following example. Both the TCSEC and the CMW requirements specify that making an object available is an auditable event. Yet, X resources are not explicitly opened or closed; they are simply referenced, so it is difficult to audit those actions. (Possible solutions are to audit every reference to the resource, audit each client's first reference to the resource, or simply audit the creation and destruction of the resource.) Other similar types of ambiguities exist in the requirements. Thus any TXS will need to make a window system-specific interpretation of the NCSC and CMW auditing requirements.

The second issue concerns how to identify the relevant subjects and objects being audited. For example, from the window system perspective, the active entities (i.e., subjects) are clients not processes. Therefore, one possible interpretation might be to audit client's actions. Although the CMW requirements explicitly state that processes must be audited, thought should be devoted to careful interpretation of the security requirements to window systems.

Because auditing is not an interoperability issue for TXS, no attempt has been made to reach an industry consensus on auditing.

4.10 Cut and Paste

As previously described, cut and paste is a client-to-client protocol in X. Furthermore, it is a two-way protocol. The pasting client tells the cutting client the desired format (e.g., text, graphics) and the cutting client responds by providing the data in the desired format. Although a TXS MAC policy will ensure that no information will flow directly from a client at a high sensitivity level to one at a low sensitivity level, additional mediation of cut and paste operations is required for two reasons. First, the CMW requirements call for a mechanism that permits a user to perform certain operations during a cut and paste (e.g., review the data, relabel it, etc). Second, B3 implementations must address the covert channels inherent in the ICCCM-specified protocol.

The covert channel arises when the cutting client is at a low sensitivity level and the pasting client is at a high sensitivity level. In this case, the reverse information flow (high to low) is roughly 50 bits per operation, depending on the options selected. Since these operations are not necessarily initiated by the user, the covert channel is of great concern at the B3 level.

Many solutions to these two problems have been proposed, including (1) requiring a trusted intermediary that supplies the CMW-required functionality then either limits the flow of data or asks for authorized user approval, (2) closing the covert information flow by limiting the formats and other data passed from the pasting client to the cutting client (dramatically reduces functionality), (3) designing a new cut and paste method (breaks existing software), and (4) building untrusted intermediaries which use operating system features to perform write-ups, and avoid needing write-downs (fairly complex). At B1 and CMW, the first solution is typically used. The last solution is best for B3, and is the method used in TRW's prototype.

4.11 Denial of Service

Denial of service problems are endemic in X. It is probably impossible to build a system which bears any resemblance to X that cannot be successfully subject to denial of service attacks. For example, clients can flood the server with graphics requests. Although the server attempts to service all clients in a round-robin fashion, service is not particularly "fair." Also, X clients are able to seize control of the pointer (in order to provide pop-up menu processing). Malicious clients could seize control of the pointer and never allow the user to regain control via the mouse. A tremendous array of other, similar, types of attacks exist.

One solution is to provide a trusted path function that allows the user to kill rogue clients, and in this way limit denial of service. In order to invoke the trusted path, some means of communicating with the TCB must be available regardless of client activities. The only generally accepted solution is to provide a secure attention key that is always delivered to the TCB and is not subject to the TXS normal processing.

Denial of service is not a concern that must be addressed at the B1 level or by CMWs. While appropriate at the B3 level, there appears to be no common resolution to this issue.

4.12 Input Processing

X allows clients to specify receipt of events (signals) upon various input conditions. For example, a client can request receipt of keyboard or pointer activity in its own, or other, windows. Additionally, clients can change the pointer (e.g., by warping it to another position on the screen). The interactions among the input processing requests are normally invisible to the user, and even to well-behaved clients. However, in a multi-level secure environment, the interactions become sources for both covert channels and user confusion.

The CMW requirements state that all user input must be labeled with an information label. A mechanism available through trusted path must be supplied to label input devices (i.e., keyboard and mouse). At any given time, therefore, an input device will have associated with it a sensitivity and information label. MAC checks must be performed to ensure that input events are not delivered to clients that do not have the appropriate sensitivity level, or are privileged. (In such cases, it would appear to the client as if no key were typed.) While existing systems tend to label the mouse and keyboard with the same label, since the mouse is typically unclassified (even when the keyboard is not), and is the source of many delivered events, the use of dual input labels (one for the mouse and one for the keyboard) is being considered. This type of solution also addresses most B3 concerns.

A similar solution which is appropriate for B3 is to have an input MAC label which is changed using the trusted path. Clients at other MAC labels cannot see or change the pointer or keyboard. To meet the requirements of the X protocol, they can simply appear "grabbed" (reserved for exclusive use of another client).

4.13 Overlapping Windows

Of all the covert channels in X, the hardest to solve is management of overlapping windows. Clients are responsible for redrawing themselves whenever they are uncovered. To optimize such redrawing, the X server sends the client notifications (expose events) which describe the size and position of the area uncovered. Because clients are uncovered as a result of the action of other clients, there is inherent flow between MAC labels.

Solutions to the problem are many, but all have significant problems. Typical solutions include:

- Doing nothing, and ignoring the large covert channel.
- Using backing storage so the server maintains a complete image, thus removing the responsibility from the client (requires potentially unlimited memory).

- Always having the client redraw the entire window, rather than just the required portion (slows down the covert channel, but at the price of greater computational effort by both the client and server).
- Using a tiling policy (instead of overlapping) to avoid the problem (doesn't work well with pop-ups; also reduces the usability of the system).

At B1 and CMW, this problem is frequently ignored. At B3, backing store appears to be the most effective solution.

4.14 Window Managers

Every project that investigates the design and implementation of a TXS begins with the laudable goal of not requiring a trusted window manager. Such an architecture minimizes the TCB size, and allows the user to substitute any look and feel desired. However, the window manager's job is to manage all windows on the screen, regardless of their sensitivity level. Since the management of windows requires both read and write access to the windows, it appears likely that window managers must be trusted.

As described earlier, X clients (including window managers) are typically written using libraries, toolkits, and widget sets. For example, the mwm window manager uses the Motif widget set, the Xt toolkit, and the Xlib library. Because the window manager is trusted, the library, toolkit, and widget set must also be trusted to operate correctly.

At B1 and CMW, the window manager, libraries, toolkit, and widget set are typically inside the TCB. At B3, the window manager should be rearchitected to remove as much as possible from the TCB. For example, that portion of the window manager that decorates windows could be outside the TCB. Also, the portion that starts other clients could be a separate process (a session manager) that might not need to be trusted. Rearchitecting a window manager is a costly and time consuming process. However, it appears to be the only solution at B3 which maintains functionality without vastly increasing the TCB size.

4.15 TCB Size and Structure

The MIT X server consists of roughly 90,000 lines of sparsely documented C code (depending on the hardware platform and options selected). Window managers, widget sets, toolkits, and libraries vary in size from 10,000 to 300,000 lines. If the entire system were inside the TCB, this could easily total 400,000 lines (including enhancements to support security, the trusted path client, etc.).

The MIT X server is a single task. While it has well-defined internal interfaces, it is an extremely complex body of code. Understanding it well enough to make a convincing argument of its trust characteristics is a major undertaking.

For B1 and CMW, the addition of the entire X system to the TCB is typically acceptable. For B3, the increased complexity and size of the TCB resulting from the addition of 400,000 lines of code (over and above the underlying operating system and network) is unacceptable. TRW's B3 prototype moves the entire window manager outside the TCB, as well as the vast majority of the X server code.

4.16 Other issues

The above descriptions do not list all of the issues in designing a TXS; other issues include enhancements to the XDMCP login protocol [8], use of classified fonts, and X extensions such as non-rectangular windows.

5 Current work

There are several projects currently ongoing to build TXSs, including MITRE's proof-of-concept prototype Trusted X design and implementation [15], CMW efforts by Sun, DEC, SecureWare, IBM, and Addamax (all aimed at commercial systems), and the TRW/TIS/CLI research prototype of a B3 TXS. Others reportedly looking at or working on TXSs include AT&T, IBM, Hewlett-Packard, Silicon Graphics, and Data General. Because X is a distributed system, it is desirable that different TXSs be interoperable, so a client targetted for one TXS will function properly on a different TXS. An informal group, the Trusted Systems Interoperability Group (TSIG), is working to define various trusted system interoperability specifications and serves as a working group where implementors discuss common concerns. The TSIG X subgroup has been working towards specifying the minimal functionality

a TXS must support (e.g., write equal), as well as trying to standardize on new interfaces to security functionality (e.g., getting and setting discretionary access control information). This is the only ongoing work on standardizing trusted X of which we are aware.

6 Conclusions

Many in the security community believe that X is inherently untrustable. While there are many problems, there are also solutions. By using some of the solutions presented here, with careful analysis and appropriate development techniques, it is possible to build TXS's at the B1, CMW, and B3 levels which still maintain a high degree of X compatibility and functionality.

7 Acknowledgements

The authors particularly appreciate the insights provided by the members of the Trusted Systems Interoperability Group (TSIG) X group. We would especially like to thank Jeff Glass for his patient explanation of many of the subtler X security issues.

References

- [1] *X Window System Protocol*, MIT X Consortium Standard, X Version 11, Release 4, Robert Scheifler, 1988.
- [2] *Xlib — C Language Interface*, MIT X Consortium Standard, X Version 11, Release 4, James Gettys, Robert Scheifler, and Ron Newman, 1989.
- [3] *X Toolkit Intrinsics — C Language Interface*, MIT X Consortium Standard, X Version 11, Release 4, Paul Asente and Ralph Swick, 1988.
- [4] *X Athena Widget Set — C Language Interface*, MIT X Consortium Standard, X Version 11, Release 4, Chris Peterson, 1989.
- [5] *OSF/Motif Programmer's Reference*, Open Software Foundation, Prentice-Hall, 1990.
- [6] Jennifer Steiner, Clifford Newman, and Jeffrey Schiller, "Kerberos: An Authentication Service for Open Network Systems" in *Proceedings of the Winter USENIX 1988 Conference*.
- [7] *Inter-Client Communication Conventions Manual*, Version 1.0, MIT X Consortium Standard, 1989.
- [8] *X Display Manager Control Protocol*, MIT X Consortium Standard, Version 1.0, 1989.
- [9] National Computer Security Center, Fort Meade, MD, *Trusted Computer Systems Evaluation Criteria*, DoD 5200.28-STD, December 1985.
- [10] *Security Requirements for System High and Compartmented Mode Workstations*, DIA Document Number DDS-2600-5502-87, John P. L. Woodward (MITRE), November 1987.
- [11] Jeremy Epstein, *A Prototype for Trusted X Labeling Policies*, in *Proceedings of the Sixth Annual Security Applications Conference*, Tucson AZ, December 1990.
- [12] National Computer Security Center, Fort Meade, MD, *A Guide to Understanding Audit in Trusted Systems*, NCSC-TG-001, June 1988.
- [13] *DNSIX Detailed Design Specification, Version 2*, DIA Document Number DDS-2600-5985-90, L. J. L. LaPadula, J. E. LeMoine, D. F. Vukelich, J. P. L. Woodward, April 1990.
- [14] *Secure Computer Systems: Unified Exposition and Multics Interpretation*, MTR 2997, The MITRE Corporation, D. E. Bell, L. J. La Padula, July 1985.
- [15] *Trusted X Window System*, MTP 288, The MITRE Corporation, J. Picciotto, February 1990.

USING EXISTING MANAGEMENT PROCESSES TO EFFECTIVELY MEET THE SECURITY PLAN REQUIREMENT OF THE COMPUTER SECURITY ACT: THE IRS EXPERIENCE.

**Richard A. Stone & Joseph Scherer
Internal Revenue Service ISM:S:R
1111 Constitution Avenue NW ARFB 2402
Washington, DC 20224**

INTRODUCTION

This paper presents information about how IRS has implemented the sensitive systems inventory and security plan requirements of the Computer Security Act. It covers those activities related to the identification of sensitive systems and the writing of security plans. Security activities outside of writing security plans, which are part of the normal development of all systems at IRS, are not addressed in this paper. The IRS approach seeks to implement active management involvement in security planning as it relates to the specific requirements of the Act, i.e. a sensitive systems inventory and security plan preparation. We believe that the successful completion of a security plans for each sensitive system in our agency is the result of a carefully coordinated attempt to take advantage of existing organizational structures and processes.

BACKGROUND

The Internal Revenue Service (IRS) is a large, organizationally complex federal agency. IRS is the custodian of very sensitive information -- people's tax accounts -- and has traditionally had a strong awareness of and concern for security. Major databases are centrally controlled and until recently isolated from other systems. The agency's automation environment is currently undergoing major change due to:

- planning and implementation of a modernized tax processing and information system,
- aging existing systems and the transition to a modernized system,
- a large and growing field automation effort, with decentralized control,
- increasing connectivity in both the existing and the conceptual system, and
- a climate which encourages innovation in developing automated applications.

These changes create a challenge for security. Security planning is basic to controlling these changes. IRS has developed procedures and tools to assure that security issues are identified and proper safeguards developed.

The IRS Security Program

Security program responsibility is shared by several organizational components. A central program office is supplemented by field and functional security components. Security staffs exist at several levels:

- the Information Systems Risk Management Branch in the Systems Management Division (National Office), responsible for the security program, support of the field operations, agency-wide continuity of operations planning and privacy issues.
- field staff in each of approximately 80 offices nationwide, part of the local information systems organization.
- functional security staffs within operational organizations.
- physical, personnel and disclosure security operations.

The program emphasizes user responsibility for security implementation. The Information

Systems Risk Management Branch supports users in this responsibility through training, consulting, distribution of information and technical knowledge.

In addition, a management level Security Council has been established to coordinate security activity throughout the agency.

Summary

Given the background of strong security emphasis in IRS, the challenge was to meet the spirit of the Act, i.e. active management involvement in security plan development, and to avoid any potential for viewing the Computer Security Act requirements as a paper exercise.

THE IRS SECURITY PLANNING PROCESS

The security planning process consists of several phases, culminating in a complete sensitive systems inventory and a security plan for each sensitive system.

Phase 1:

Plan generation. This phase was intended to identify the majority of our sensitive systems and coordinate preparation of a security plan for each. It was in this phase that we determined our approach and first identified the need to look to our organizational structure and processes for strategic help. Questions we asked at this time included:

- what internal IRS authority is the appropriate level to initiate this activity?
- who will decide what is a system?
- who will make the sensitivity decision?
- who will review these decisions?
- what will the security plans look like?

Enabling the process. To establish and define the process, information would have to be communicated to all organizations, describing the requirement, fixing responsibility for implementation, and establishing procedures and time-frames. We decided to use an internal memorandum to communicate this information. We asked the Chief Information Officer (CIO) to sign this memo. The CIO is the senior information resources executive in IRS, and has security responsibilities to Treasury. He also oversees both the information systems operational and design functions. Organizationally he is at the deputy commissioner level. He is also chairman of the Information Systems Policy Board (ISPB), the senior decision making body for all major new systems. The memorandum was addressed to the next lower level of organization, the Assistant Commissioners. One value of communication at this level is that the correspondence is controlled and responses come through this same assistant commissioner level for signature. Thus we began an awareness process at a high level through this memorandum/response process. Of course, subsequent comments, calls for clarification or additional information now can follow this same route, allowing reinforcement of management's part in the security planning process.

Sensitive system identification. Each year sponsors of budget initiatives must provide input to a report called the Information Systems Plan. This input provides a description of the initiative, its intent, what information systems are part of it, as well as a multi-year development plan, budget information and project status report. It is a major annual submission, forwarded to Treasury to become part of annual submissions to OMB. The detail provided by this plan and the fact that it identified responsible parties for named systems, led us to use it for purposes of identifying

sensitive systems. Sponsors of ADP initiatives were asked to identify the sensitive systems contained in each of their budget initiatives, using the OMB "definitions" for systems and applications. Sponsors were assured that they had authority to make sensitivity determinations. The security function was available to discuss the implications of a decision, or to help draw logical lines around systems, but in the end would not dictate what was and what was not sensitive. Information Systems Risk Management Branch analysts prepared briefings about what constitutes sensitivity, including the need for confidentiality, integrity and/or availability. Many managers, we discovered, had considered only the confidentiality issue and were surprised to learn that a system can be sensitive for other reasons. The system names, along with a contact name and phone, were forwarded to the Information Systems Risk Management Branch for review.

By responding with a sensitive system name, sponsors assumed a *de facto* responsibility to create a security plan for each identified system.

Creating a standard security plan format. OMB Bulletin 90-08 communicated an outline of information to be included in a security plan. This guidance was the basis for the form we created. However, Treasury had created a security planning form, with input from its bureaus, before OMB guidance was distributed. We had distributed the Treasury format in several documents prior to OMB 90-08 guidance. Considering possible reviews of plans by either Treasury or OMB at some future time, and the usefulness of a standard format for our own plan development and review procedures, we created a new form using both OMB and Treasury-requested information. We incorporated the OMB guidelines for reporting control measures into two worksheets, one for applications and one for support systems. We have made this form available in hard copy, DOS text or database screen version. The Information Systems Risk Management Branch provides training and consultant services to those responsible for writing of security plans.

Plan review, approval and follow up. Identifying the plan approval process required some strategic planning. Approval is an act of validation and even when done at the branch level stands for agency validation. Since OMB A-130 and the Computer Security Act are very clear in making the information system or application user/ management responsible for security, we felt that an approval mechanism limited to the Information Systems Risk Management Branch would be inconsistent with the intent of those documents. Thus we made the Information Systems Risk Management Branch review of plans a technical evaluation of the plan's completeness and had it focus on missing information and unresolved security issues. With this we reinforced the assurance we had given earlier to plan preparers, that user organizations could make valid security decisions.

A newly formed executive-level Security Council provided us with a way to again involve management in the planning process, and make agency approval of plans be at a high organizational level. The Security Council consists of directors from divisions having some major security responsibility, and include a field representative. Its charter is to foster a climate of security within the agency, among other activities. We approached the council, emphasizing not plan approval, but concurrence. We showed them how plan review by the council would give them a very quick picture of security within the agency and help them prioritize their concerns. This concurrence also assures high level management support for security planning activities and provides independent support to the functionally approved Security Plan.

Our current process then is to have draft security plans reviewed by the Information Systems Risk Management Branch for completeness and to identify the need for any additional training or technical assistance. After clarifications and additions are complete, the plan is given a further technical review by Risk Management Branch analysts. In addition, selected plans are reviewed by an independent third party, to validate the internal review process. This review leads to a recommendation to the Security Council to concur with the plan, concur with caveats or to reject the plan.

The Information Systems Risk Management Branch has created a database to contain plan information. It is intended that this database will provide information needed for any Treasury or OMB call, without going back to the users to ask for another piece of paper. However, the most valuable part of the database will be the tracking of "planned for" safeguards, pending and planned reviews, and risk analyses and other timed events. We anticipate using reminder notices and offerings of help from the Branch, to keep the users active in plan updating and implementation.

Timetables. The following schedule of phase one activities is currently being completed.

- November 5, 1990 - Memorandum from Chief Information Officer
- November 15, 1990 - Contact Point designated
- November 30, 1990 - Sensitive System names due
- November 27, 1990 - Help Session for developing Security Plans
- February, 1991 - Security Council review of sensitive systems inventory
- March 12, 1991 - completed security plans due to System Management Division
- April/July, 1991 - plan review
- September, 1991 - concurrence by Security Council

IRS expects to have a security plan completed for the majority of its sensitive systems by September 30, 1991.

Phase 2:

Field systems. Although Phase 1 will account for the majority of IRS systems, it will not account for them all. Field components will be asked to review the final Phase 1 sensitive systems inventory. They will be asked to identify any of their systems which do not appear on the inventory. They will then follow the same procedures for security plan preparation. Field visits by Information Systems Risk Management Branch will communicate the security planning process and address the field role. Field security analysts are already involved in the planning for Phase 2.

Phase 3:

Plan implementation. Implementation of security plans is most significant part of the planning process. As mentioned above, plan information will be entered into a database maintained by the Information Systems Risk Management Branch. This will be used to:

- facilitate reporting to Treasury and internally to IRS (e.g. Security Council);
- allow monitoring of implementation;
- reveal timetables for future reviews and risk analysis:
 - advance notice to functions,
 - opportunity to prioritize needed actions;
- make update easy.

Plans will be retained by users. However, there will be a need to involve the Security Council and the Chief Information Officer in some reporting mechanism, to continue their involvement in the planning process. The details of this involvement are now being organized.

SUMMARY

The IRS implementation of the security planning and sensitive systems inventory requirements of the Computer Security Act use organizational structure and processes to bring about a larger acceptance of and responsibility for security planning at management levels. We feel that security planning activity has reinforced the role of everyone in securing sensitive information.

VIRUSES IN AN OS/2 ENVIRONMENT: REMEMBRANCES OF THINGS PAST AND A HARBINGER OF THINGS TO COME

by Kevin Haney
National Institutes of Health
Division of Computer Research and Technology
Building 12A, Room 3039
Bethesda, Maryland 20892
Internet Address: khv%nihcr31.bitnet@cu.nih.gov

ABSTRACT

To date, there have been no confirmed incidents of a computer virus that specifically targets OS/2 systems. However, the many DOS viruses loose in the land do present a real and present danger for OS/2 users since most OS/2 systems are capable of running DOS programs, including DOS programs that have been infected by a virus. This paper describes the danger to OS/2 systems posed by DOS viruses and suggests countermeasures that may be employed against viruses in the future. The information presented is based on a series of experiments conducted with various DOS viruses in a controlled OS/2 environment. Some prognostications are also offered as to the form OS/2 viruses may take when they are eventually created. A plea is made for the notion that security and antiviral features should be built into OS/2 and other advanced microcomputer operating systems as an integral component.

With the current hype surrounding Windows 3.0, the subsequent defection of many OS/2 application developers to the Windows camp, and the delay of new versions of OS/2, the small but faithful band of OS/2 users have had few things to be thankful for recently. But, while we do not have the huge installed base of DOS or the flood of new applications that are becoming available for Windows, we do have at least one thing over the hordes of DOS and Windows users—to date, there have been no incidents in the general computing community of a computer virus that specifically targets OS/2 systems. That will no doubt change as the installed base of OS/2 grows and, in the belief that to be forewarned is to be forearmed, the present paper will offer some prognostications as to the form that OS/2 viruses may take when they are eventually created. In the meantime, however, the many DOS viruses loose in the land do present a real and present danger for OS/2 users. The primary purpose of this paper is to describe that danger and suggest countermeasures that may be employed by both the developers and users of OS/2. The information pre-

sented is based on a series of experiments conducted with various DOS viruses in a controlled OS/2 environment. The primary conclusion will be that, along with DOS compatibility, OS/2 has inherited the virus problems and security vulnerabilities inherent in DOS as well.

BASIC CHARACTERISTICS OF OS/2 VERSUS DOS

The Disk Operating System (DOS), which was introduced with the first IBM PC in 1981, is a singletasking, real-mode operating system based on the Intel 8088/86 microprocessor instruction set. It is designed to run one application at a time in the 1 megabyte real-mode address space of an Intel 8088 or compatible processor. Its user interface is character-based and command-driven, although graphical shells such as Windows can be substituted for the command line interface. Since DOS is at heart a singletasking operating system, DOS programs operate on the assumptions that they are the only

program in memory and that they exercise complete control over the system hardware. In today's world of terminate-and-stay-resident programs and DOS program switchers, these assumptions may in fact be incorrect and, as a result, well-known compatibility problems may occur.

In 1987, IBM introduced Operating System/2 (OS/2), an advanced, multitasking, multithreaded, graphical operating system for machines based on the Intel 80286 and above processors. The fact that OS/2 is multitasking means that you may run two or more OS/2 programs simultaneously, and the fact that OS/2 is multithreaded means that each program may concurrently run two or more separate processes or threads of program execution. OS/2 has full preemptive multitasking where applications can intelligently request CPU cycles which are then assigned on a priority basis by a scheduler process which is a part of the operating system kernel. This is a more advanced and efficient mode than the more usual time-slicing as seen in Windows 3.0 and the Macintosh operating system. Under Presentation Manager (the graphical interface of OS/2), up to sixteen OS/2 programs can run concurrently.

OS/2 is a protected-mode operating system. Memory protection mechanisms are built into the Intel 286, 386, and 486 processors and are utilized by OS/2 so that concurrently executing programs or tasks cannot bring down the whole system as a result of a crash. DOS, on the other hand, offers no such protection. Any DOS program can modify any other program in memory and can also modify the operating system itself, for example, by changing the interrupt vector table to intercept keystrokes or disk accesses. There is nothing to stop any executing DOS program from accessing and changing the value of any physical memory location within the address range of the processor. OS/2, on the other hand, uses a system of local and global descriptor tables (i.e., listings of what parts of physical memory each program is allowed to access) so that programs can be prevented from either reading or writing to any memory address outside of their allocated memory space. If an application attempts to do this, either purposely or because of a programming error, a protection violation will be produced and the offending application may be cleanly terminated without affecting other applications or the operating system. In effect, the use of descriptor tables creates a logical address space so that the application is insulated from having to deal with physical memory addresses. Also, since OS/2 is based on the 80286 processor, it implements the

four-level hardware protection scheme of that processor to isolate programs from each other and from the operating system. These features provide for a much more stable operating environment than DOS could ever provide.

A feature of OS/2 that will become important when we discuss program-infecting viruses is the High Performance File System (HPFS). Before HPFS, operating systems used a single file system which was fixed and unchangeable. Support for installable file systems was introduced with OS/2 version 1.2. A file system is that part of the operating system that translates "logical" file requests from an application program, such as requests to open, create, read, or write to a file or directory, into sector-oriented requests that the disk controller can understand. Anyone can write a device driver to support a file system for a non-standard storage device such as a CD-ROM drive and have it installed as part of the OS/2 system. IBM supplied the High Performance File System in an attempt to address the problems and limitations of DOS's File Allocation Table (FAT) file system. HPFS provides faster access to large disk partitions of up to 2 gigabytes, support for up to 16 partitions on a drive, file names up to 255 characters long with case preservation, extended attribute support, and built-in directory and disk caching.

HPFS maintains compatibility with the FAT file system at the Application Programming Interface (API) level. This means that all DOS or OS/2 programs that use the standard API disk and file calls will have access to HPFS partitions. This includes DOS programs running in the DOS compatibility box (see below). The FAT file system is still embedded in the OS/2 kernel and can be used concurrently with HPFS. All disk partitions may be configured as either FAT or HPFS, or the primary partition may be configured as a bootable FAT partition with one or more extended HPFS partitions (or vice versa). OS/2 includes a dual-boot facility which enables a system to boot up either DOS or OS/2. If an OS/2-DOS dual-boot system is booted under DOS, programs cannot access an HPFS partition or any FAT partition that comes after an HPFS partition. Only fixed disks may be formatted for use with the HPFS—diskettes are not supported.

Another difference between OS/2 and DOS that is important when discussing program-infecting viruses concerns the structure of executable files within each operating system. In order to manage simultaneously executing programs within a limited

amount of physical memory, OS/2 must be able to move programs around in memory to take advantage of the memory blocks that are available. Thus, since every OS/2 program must be relocatable within memory, there is no OS/2 analog to the DOS .COM file, i.e., a program file that is an image of the program as it exists in a certain location in memory. All OS/2 programs are .EXE programs, with file headers that contain the information necessary to relocate the program to any part of memory. Any OS/2 .COM files which may be present on an OS/2 system really have the .EXE format, even though their extension is .COM.

OS/2 retains compatibility with DOS programs by providing a "DOS compatibility box," which is an emulated DOS environment in which a single DOS program can run (OS/2 version 2.0 adds the capability to run multiple emulated DOS sessions simultaneously). The compatibility environments of OS/2 versions 1.2 and 1.3 are really a subset of DOS 4.0. Only one DOS program can run at a time, and it will run only when it is in the foreground—when switched to the background, it ceases to execute. However, when a DOS program is being run in the foreground, other OS/2 programs can be executing simultaneously in the background. Most DOS programs will run in the DOS box and *this includes DOS programs that have been infected by a virus*. Those programs that might not run properly in the DOS box include programs that are timing-dependent, such as communications pro-

grams, those that require special device drivers, and those programs, such as low-level disk utilities, that attempt to directly control the system hardware by bypassing the normal system device drivers.

DOS VIRUSES ON AN OS/2 SYSTEM

The classic definition of a computer virus is by Cohen and states that "a computer virus is a program that can 'infect' other programs by modifying them to include a possibly evolved copy of itself."¹ While not a precise definition, it nevertheless does provide a good working notion of what a virus is. An essential part of this definition is that a virus must be a piece of executable code, i.e., a program. A plain data file cannot contain a virus, although if that data file also contains embedded executable instructions, such as a spreadsheet or word processing macro, then those instructions may indeed harbor a virus.

We can therefore divide the total class of viruses into different types depending on the kind of executable code they can infect. I propose the following taxonomy. What I will call Type I viruses infect the boot sector of hard disks and diskettes. Such viruses can infect only boot sectors, but there are two subtypes in this category that can also infect hard disk partition tables and program files. Type II viruses only infect executable program files. Type III viruses infect program overlay (.OVL) files

Table 1 - Virus Types

Virus Type	Subtype
Type I - Boot Sector Infectors	1. Only infects boot sectors
	2. Also infects partition tables
	3. Also infects executable files
Type II - Program Infectors	1. Infects .EXE programs
	2. Infects .COM programs
	3. Infects .COM & .EXE programs
Type III - External Routine Infectors	1. DLL Infectors (OS/2 and DOS)
	2. OVL Infectors (DOS)
Type IV - Device Driver Infectors	1. Infects DOS device drivers
	2. Infects OS/2 device drivers
Type V - Macro Infectors	1. Infects spreadsheet macros
	2. Infects word processing macros

and dynamic link libraries (.DLL's), or any other type of code that is not executable by itself, but is called from other programs. Type IV viruses include those viruses that infect device drivers (.SYS files), since device drivers are a different kind of executable code than anything in the other types. Type V viruses infect macro instructions or other executable code found in data files. Such viruses are not really operating system-specific but rather application-specific. Since examples of Type III, IV and V viruses are currently very rare, we will concentrate our discussion on viruses of Type I and Type II.

Type I Viruses - Boot Sector Infectors

The first thing that should be understood about most boot sector viruses is that the primary way for a machine to become infected with such a virus (e.g., the Brain virus) is for it to be booted up from an infected floppy diskette. Accessing files on an infected diskette after the system has booted up from another clean hard disk or diskette cannot spread a normal boot sector virus infection.² The boot sector is a good virus infiltration vector because one is present on every disk or diskette formatted with the DOS or OS/2 FORMAT command. The DOS and OS/2 boot sectors do differ slightly, but their essential mode of functioning is the same. Another infiltration point for Type I viruses is the partition table of hard disks. A partition table, or Master Boot Record (MBR), is present on every microcomputer hard disk no matter what operating system the disk has been formatted for.

In order to understand how Type I viruses infect boot sectors and partition tables, it is necessary to understand the process that occurs when a PC is booted up. After a powered-up PC has run its initial diagnostic tests, it will read track 0, sector 1, side 0 of the floppy disk in the A drive if one is present. If the A drive is empty, it will read that same location on the hard disk, which contains the MBR. The MBR is a single sector which indicates how the hard disk is divided into partitions, which partition is the bootable one, and a name designation that indicates what operating system the partition is formatted for. After this information is read, code in the MBR is executed that reads the boot sector of the bootable partition, which then goes on to load the operating system into memory. This short piece of executable code in the MBR can be infected by a virus. Examples of Type I viruses that target the MBR are the Stoned-B, Anthrax, EDV, and Joshi viruses.

Since bootable partitions on hard disks do not normally have access to other bootable partitions, viruses that infect the MBR must also have some other medium of transmission, usually either the boot sector of diskettes or program files, i.e., no viable virus can infect just the MBR. It is also important to note that since the MBR code is executed before the operating system is loaded, MBR viruses are operating system-independent in that they do not rely on services provided by the operating system in order to load and execute. However, since all of the MBR viruses to date are written to propagate in a DOS environment, they all assume that DOS will be the operating system that will be loaded. We will see what happens when that assumption turns out to be incorrect.

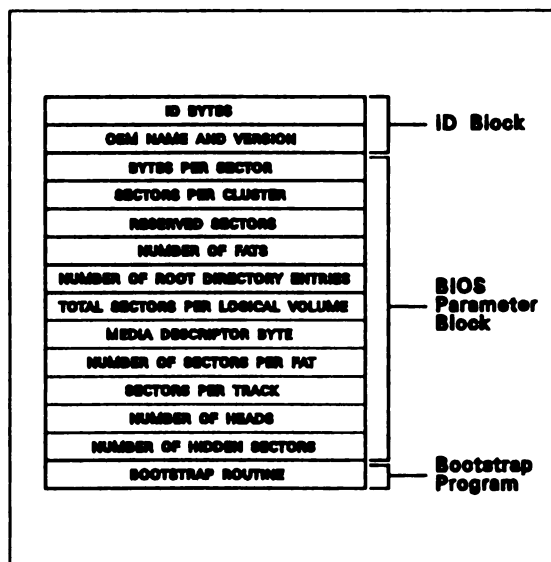


Diagram 1 - Layout of a DOS Boot Sector

To return to the role of the boot sector in the boot process, after the MBR has been read and the bootable partition identified, control is passed from the code in the MBR to the code in the boot sector of the bootable partition. The first part of the boot sector is the ID block, which contains some identification bytes and the OEM name and version number. The next part is the BIOS parameter block (BPB) which contains information needed by the device drivers on the physical format of the hard disk or diskette. After the information in the BPB is read, a short executable program, the "bootstrap" program, is run to load the primary operating system files into memory. These files are IBM-

BIO.COM and IBMDOS.COM for PC-DOS, IO.SYS and MSDOS.SYS for MS-DOS, and OS2LDR and OS2KRNL for OS/2.

This bootstrap routine is the point at which a boot sector virus infects a system. Such a virus will typically hide its viral code in hidden sectors that have been marked as bad in the file allocation table, which makes them inaccessible by normal means. The virus will then insert a jump instruction at the front of the bootstrap routine that points to this hidden code. When executed, the bootstrap routine will jump to the viral code and execute it, then return to the original bootstrap program and

In an experiment conducted on an IBM AT using IBM OS/2 Standard Edition 1.2, the Stoned-B virus, a variant of the original Stoned virus that is able to infect hard disks as well as diskettes, was able to successfully infect the MBR of the hard disk when booted up from an infected diskette. This was confirmed by inspecting the MBR with a sector editor program. However, when the newly-infected machine was booted up, OS/2 was still able to load and function normally. A memory scan was performed in the DOS compatibility box with the Norton AntiVirus program. It identified the Stoned virus as being present in memory at the top of the conventional DOS memory space (hex address

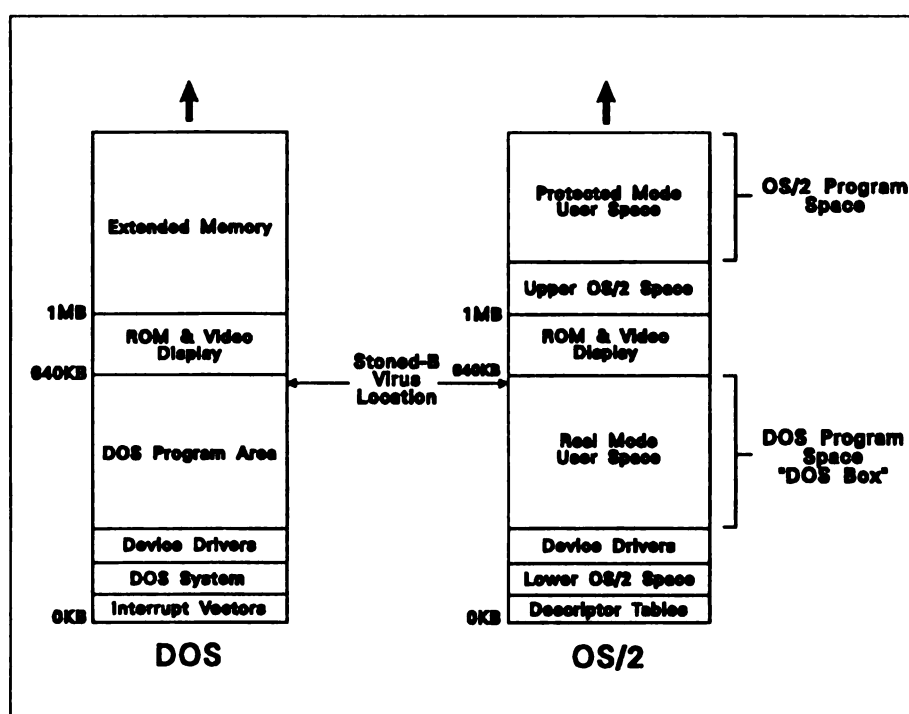


Diagram 2 - Generalized DOS and OS/2 Memory Maps

proceed to load the operating system files into memory. In the meantime, the virus program may have infected other disks or become resident in memory, allowing it to infect diskettes as they are accessed by the system. If the bad sectors into which the viral code was copied happen to have been part of a file, that file will be corrupted and at least part of its data will be lost. Viruses that infect the MBR also use this general redirection strategy.

9F81:0005—see diagram 2). In a parallel experiment, exactly the same results were achieved when the machine's hard disk was formatted as an HPFS partition. This is to be expected since the Stoned-B virus, when it has infected the partition table on a hard disk, is activated before any operating system is loaded and thus is not dependent on any particular file system which the operating system may employ. Therefore, not even using a non-DOS file system like HPFS is enough to prevent infection by insidious viruses such as Stoned-B.

The situation is not as bleak as it may sound, however, because in neither of the above experiments was the virus active nor could it infect any diskettes or spread any further. To understand why the virus was not active, we need to understand how the virus normally installs itself into memory and activates, and also understand the concept of an interrupt. An interrupt is a facility of the micro-processor that enables it to suspend whatever it is doing, respond to some system event or program request, and then continue the task that was suspended. Interrupts issued by a physical device like a keyboard or disk drive are called hardware interrupts. An interrupt issued by a program requesting some system service is called a software interrupt. When the Stoned-B virus is initially activated, either through the MBR on a hard disk or the boot sector on a diskette, it reserves about 2 kilobytes of memory for itself and changes the interrupt vector table address for interrupt 13H, which is used to control disk services, to point to the viral code in memory. After DOS is loaded, the virus will intercept any requests for a disk read or write via interrupt 13H, check the hard disk or diskette, copy itself to the accessed disk if it is uninfected, then perform the requested read or write operation.³

The addresses for the interrupt handling routines in the interrupt vector table are initially set by the ROM BIOS during the initial system boot process. However, the operating system may change any of the interrupt vectors once it is loaded. The fact that lets the Stoned-B virus operate is that DOS does not reset interrupt 13H when the DOS kernel is loaded. OS/2, on the other hand, does reset interrupt 13H when it is loaded because it uses its own interrupt handling routines instead of the ROM BIOS routines. This resetting of the interrupt vector table causes the virus to lose its "activation hook." The area of memory to which the Stoned-B virus copied itself is part of the memory space used by OS/2 for the DOS compatibility box. This memory area is not overwritten when OS/2 is loaded, thus the virus scanner found the hex string corresponding to the Stoned virus in memory. The virus was "dead," so to speak, because its hook into interrupt 13H had been overwritten. It can be expected that any virus that operates similarly to the Stoned-B virus will suffer the same fate.

In another experiment with the Stoned-B virus, an infected hard disk on an OS/2-DOS dual boot system was able to successfully infect the OS/2 installation diskette, which is a bootable diskette, when the machine was booted under DOS and the

diskette was accessed. When this infected diskette was booted on a clean OS/2 machine, it was able to infect the MBR on the hard disk. The message "Your PC is now Stoned!" appeared and the OS/2 installation program attempted to load, but the system hung up after the copyright message was displayed, requiring a cold boot. It is good security practice to never under normal circumstances boot a hard disk system from a floppy diskette. However, when OS/2 is installed, you must boot up from the installation diskette in order to run the installation program. This requirement introduces a potential virus infiltration point for Type I viruses. Unfortunately, it may not be possible to eliminate the boot requirement since OS/2 must be installed under a common, fixed operating environment.

Type II Viruses - Program Infectors

More numerous than boot sector viruses, are viruses that infect executable program files, i.e., .EXE and .COM files. These are what I call Type II viruses. There are several subtypes in this category. Some viruses of this type specifically target the operating system files. For example, the Lehigh virus only infects COMMAND.COM. These viruses are known as system infectors. However, most program-infecting viruses will infect any executable program, although some are restricted to just .COM files or just .EXE files. Many viruses remain resident in memory so that they have access to programs and disks that are accessed by the system during normal operations. These viruses are known as TSR (terminate-and-stay-resident) viruses. Many, if not most, Type I viruses are also TSR viruses.

Following Stubbs and Hoffman⁴, we may draw a further distinction between overwriting and nonoverwriting program-infecting viruses. Overwriting viruses are the simplest to create—they just overwrite the first part of the program with their own viral code. When an attempt is made to execute the infected program, the viral code is executed instead and the virus may attempt to spread or do its damage at this time. If the part of the original file that was overwritten contains essential instructions, the program will either not run, behave abnormally, or crash the entire system. However, since a problem is then apparent, overwriting viruses are usually discovered early and their spread is thus greatly reduced. Some examples of viruses in this category are the AIDS and Kamikazi viruses.

Nonoverwriting viruses (also known as parasitic viruses) are a far more serious threat. They retain all of the functionality of the original program and add their code to it. They do this by either increasing the file size or by hiding in unused space within the original file, such as stack or data space. Most nonoverwriting viruses that attack .EXE files will append their viral code onto the end of the program file and insert a jump instruction at the beginning of the program which points to the viral code. After the viral code is executed, the virus will then jump back to the original program and allow it to run. No abnormal symptoms are usually produced by these types of viruses until they do whatever destructive thing their author programmed them to do. As most viruses of this type have a

Many Type II viruses remain memory-resident after their initial invocation. It is important to note that these TSR viruses can function in OS/2's DOS compatibility box as it supports the normal TSR calls that the program would make under DOS, allowing the virus to be active when other DOS programs are running. If the DOS session is suspended, the virus will likewise be suspended, but while the DOS session is active, a TSR virus can function normally and spread to other programs or disks.

A series of experiments was conducted with various Type II viruses on an IBM P70 running IBM OS/2 Standard Edition 1.2. In these experiments, the Devil's Dance, Yankee Doodle, Cascade,

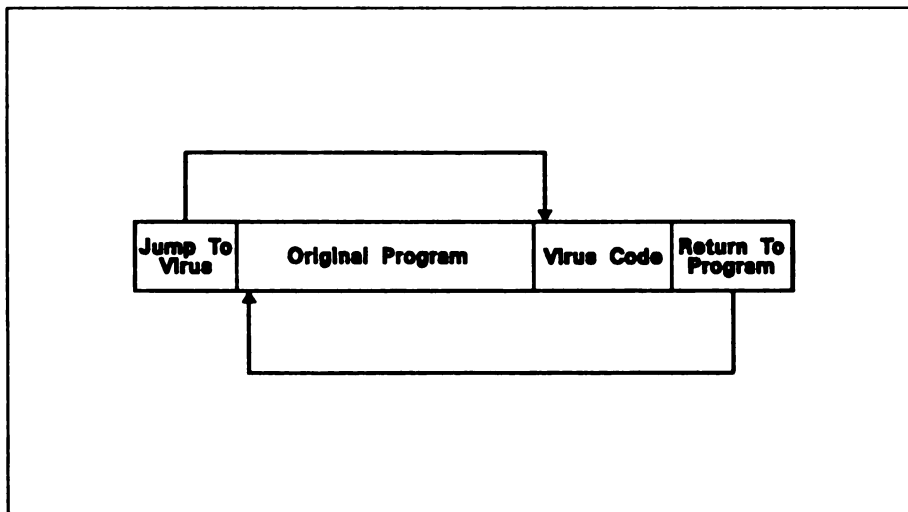


Diagram 3 - Structure of a Nonoverwriting Virus

built-in delay until the destructive portion of their code is activated, their detection usually takes a longer time with a result that the virus has a chance to spread widely. Some of the most common examples of nonoverwriting viruses are the Jerusalem, Devil's Dance, and Cascade viruses.

The analogy between computer viruses and human viruses has perhaps been over emphasized, but it is appropriate here. If a virus (biologic or electronic) proves immediately destructive to its host (a person or a computer), the virus will not have a chance to spread and most likely will quickly die out. It is the viruses that allow their host to go about their normal activities that are able to spread widely, even though the host, and therefore the virus, may be killed in the end.

and Sunday viruses were all able to install themselves in memory when DOS programs infected with these viruses were run. The viruses were then able to infect a DOS memory mapping utility (MAPMEM.COM) when it was run, as well as various other DOS programs. This was found to occur on both FAT partitions and HPFS partitions. The infections were confirmed through the use of the IBM Virus Scanning Program, version 1.3.

It is very important to note that when a Type II virus is run in the DOS compatibility box, it can infect files on an HPFS partition as well as a FAT partition, since OS/2 routes the disk and files requests through the appropriate file system driver. However, if a virus were to attempt to infect a file on an HPFS partition by directly reading or modi-

fyng the FAT or root directory, unpredictable results will occur since there is in fact no FAT or root directory on an HPFS partition. The sectors normally occupied by the FATs and root directory serve other functions on an HPFS partition. If an area which is attacked by a virus happens to be used by a file, that file will of course become corrupted.

When a Type II virus looks for an executable file to infect, it does not know or care if that file is a DOS program or an OS/2 program. If a virus infects .EXE files and identifies its potential targets not by the file extension but by looking for the normal .EXE signature of 4D 5A (MZ) as the first two bytes, then that virus will infect both DOS and OS/2 .EXE files as well as OS/2 .COM files, which have the .EXE format. In another experiment, both the Devil's Dance virus and a strain of the 1260 or V2P2 virus were able to infect the OS/2 system editor (E.EXE) when it was placed in the same subdirectory as an infected DOS program and the infected program was run in the DOS compatibility box. In both cases, when the system editor was subsequently invoked from the OS/2 command line, the editor would not run and a SYS2070 error message was produced. This error message indicates that the system could not demand load the application's segment, which is a general indication of a corrupted program file. The Sunday virus was also able to infect the system editor. When the editor was invoked, a SYS0193 error message was produced. This means that the specified program is either a DOS mode program or is not compatible with OS/2. The Sunday virus was thus able to overwrite the information in the header of E.EXE that OS/2 uses to identify the program as an OS/2 program. We can therefore conclude that OS/2 programs are as susceptible to damage by a DOS virus infection as are DOS programs.

OS/2 VIRUSES OF THE FUTURE

It is inevitable that in the not-too-distant future, someone will write the first OS/2 virus. Since OS/2 is an advanced operating system with many more features and capabilities than DOS, it provides a virus creator with more resources, as well as a greater programming challenge because it is many times harder to write an OS/2 program than it is to write a DOS program. This is one reason why we have yet to see an OS/2 virus. Mirroring the development of OS/2 applications themselves, we will most likely first see basic DOS viruses ported to

OS/2, followed by viruses targeted specifically at OS/2 systems.

One aspect of OS/2 that provides additional opportunities for virus infiltration is OS/2's use of dynamic link libraries (.DLL files). DLL's perform the same function in OS/2 that overlay (.OVL) files do in DOS—they provide a library of programming routines that can be stored externally to the program file itself, and that can be linked at run time instead of when the program is originally compiled. This provides a means for different programs to use a common set of routines, thereby reducing the size of the program file and saving disk space. It also provides for more efficient memory management since a routine does not have to be loaded into memory until it is actually used.

.DLL files are executable code. Thus, it is possible for a Type III virus to insert itself into such a file and be activated whenever the DLL is called by an application program. Since it is probably not very common to carry around .DLL files on ordinary diskettes, a DLL-infecting virus would most likely have a difficult time propagating. When the use of dynamic link libraries becomes more common for OS/2 programs, the danger of a DLL virus will increase. DLL's are also beginning to be used for DOS programs. Windows 3.0, for example, makes extensive use of them.

Two means of protection from DLL-infecting viruses suggest themselves. Programs can (and should) include code that can check a .DLL file to make sure it has not been modified before it is loaded. This is a simple extension of the self-checking ability that is increasingly being built into DOS programs, and it makes just as much sense in an OS/2 environment. The second method of protection is that virus scanning programs should be able to scan .DLL files for the presence of viruses, without having to scan every file on a disk. It is now common for scanning programs, besides scanning for viruses in .COM and .EXE files, to also scan .OVL, .SYS, and even Windows .PIF files. With the numbers of dual DOS-OS/2 systems increasing and the increased use of DLL's in DOS programs, manufacturers of virus scanning programs should include at least the option of scanning .DLL files.

There is another feature of OS/2 that lends itself to exploitation by virus authors. The basic feature of a computer system that determines the form of the programming code that can be written

for it is called the Application Programming Interface (API). The API for a particular operating system specifies the form and conventions of the coded instructions that application programs use to communicate with the operating system and hardware. Besides the normal protected-mode OS/2 API, OS/2 includes another API called the Family Application Programming Interface, or FAPI. The FAPI is a subset of the regular OS/2 API that roughly corresponds to the basic system functions provided by DOS. These services are essentially the non-multitasking, system API functions such as low-level video, keyboard, file I/O, and device management services. Programs written to the FAPI will run under both DOS and OS/2. This means that the same .EXE file can run in both environments. Although hardly any application developers have taken advantage of this basic level of DOS-OS/2 compatibility, the possibility exists that a virus could be written using the FAPI that would run under both DOS and OS/2. Needless to say, such a virus would be equally destructive in both environments.

Since FAPI programs are .EXE files, virus scanning programs will check them for a possible virus infection and no extra security measures are called for besides those that are normally followed for executable programs. The potential existence of a FAPI virus does mean that we cannot divide programs into the two mutually exclusive categories of DOS programs and OS/2 programs and treat them differently with regard to the possibility of them being infected by a virus. Rather, all executable program files have to be considered to be potential virus infiltration vectors and treated with the appropriate caution.

WHAT CAN BE DONE?

Generally speaking, OS/2, even though it is one of the most advanced operating systems yet developed for microcomputers, is no more secure than DOS. Security in the microcomputer world has hitherto been confined mainly to virus scanning programs and the protection and encryption of classified data and has been provided by add-on products that are not a part of the operating system itself. This will change. The notion that security should be an integral part of a microcomputer operating system is rapidly gaining acceptance. Built-in security and antiviral features will come to be expected as a matter of course for any advanced operating system in the next decade. Organizations

who entrust mission-critical applications to an advanced operating system have a right to expect built-in security and data protection features. As increased corporate downsizing results in applications that were formerly being run on a mainframe now being run on microcomputer systems, some of the security features of mainframe operating systems will have to be provided for microcomputer operating systems as well. Thus, microcomputer operating systems will start to take on more and more of the characteristics of mainframe operating systems, a process that the multitasking nature of OS/2 seems to exemplify today.

There are two basic reasons that would lead one to the conclusion that security features which are built into the operating system would be more desirable than having to rely on add-on security products. First, such built-in security measures, since they would be developed by the developers of the operating system itself, would be more tightly integrated with the operating system than any add-on product could ever be. This would hopefully result in a more efficient and better performing security subsystem. The second, and probably the more important reason, is that if the security subsystem was an integral part of the operating system, everyone who had a copy of the operating system would also possess a copy of the security subsystem and its use would therefore be much more widespread. No matter how cheap, effective, and easy to use a security product is, if you have to buy, install, and operate it separately, fewer people will go to the trouble to do so. And after all, the only way to effectively reduce the proliferation of viruses is for a large percentage of the computing community to use effective antiviral and security products.

As we have said, when OS/2 workstations and networks running mission-critical applications become more common, it will be necessary for the operating system (and hardware) to have built in safeguards against viral infections, data loss, data corruption, and unauthorized tampering. What antiviral features could a true security subsystem for future versions of OS/2 contain? Here are some suggestions.

1. **System Self-check** – When OS/2 is initially loaded, it should perform a self-check of all of its essential files to ensure that they have not been modified, employing a secure cryptological algorithm. Many DOS application programs do this now and it is even more important for operating

system files to be checked since these files provide a very common infiltration point for viruses.

2. **DLL Self-check** – A self-check should be performed on any dynamic link library that is called to ensure that it has not been modified. The operating system should check its own DLL's and application programs should check any DLL's which they call.

3. **Disk Check** – When OS/2 is loaded, an integrity check should be performed on the hard disk partition table and boot sector to detect any viral code since, as we have seen, the successful loading of OS/2 is not enough to ensure that the hard disk is not infected by a virus. ROM BIOS support might be required for the successful implementation of this capability.

4. **Monitoring Program** – Since OS/2 is a multitasking operating system, it could contain a monitoring process that could execute in the background and would monitor programs and intercept any attempts to do something destructive, such as write to an .EXE file or change the hard disk boot sector. Such a monitoring process could also use a checksum-type algorithm to compare the current state of the program to a previously recorded state to determine if it had been modified in any way, before the program is allowed to execute. This corresponds to Hruska's idea of an "integrity shell," difficult to implement securely in a DOS environment but perfectly suited to a protected-mode, multitasking operating system such as OS/2.⁵ This should be a user-selectable option so that it could be disabled in cases where it is not needed, or where system performance is critical.

5. **System Utilities** – Built-in OS/2 system utilities should include utilities that could make a backup of the system areas on a disk (i.e., partition table, boot sector, FAT's and root directory), which could then be restored in case of a viral attack or disk corruption. A utility for file undeletion should also be included to aid in recovery after a viral attack. There are programs, such as the Norton Utilities, that perform these functions in the DOS environment. Yet, four years after the introduction of OS/2, there are still no commercially available utilities that can perform these functions in OS/2.

What can OS/2 users do to protect themselves against viral infections? Exactly the same things that DOS users should do, e.g., make backups, scan

all new programs, write-protect diskettes, don't boot up from a diskette, don't run a program of unknown origin, etc. In addition, if you do not need the ability to run DOS programs, you can configure OS/2 to operate in protected mode only. However, with the current dearth of OS/2 application programs, it will likely be a long time before most users can afford to give up the DOS compatibility box. If the DOS box will run all the DOS programs you need to run and you do not need a DOS dual-boot capability, formatting your entire hard disk for the high performance file system provides some measure of protection against viruses that wipe out FAT's and root directories, as well as providing much better performance. However, if you do have disk problems, there are as yet no disk utilities that can work with HPFS partitions.

One would have hoped that with the advent of a new, advanced operating system for personal computers, the risks associated with computing could be reduced. That has yet to happen, however. Let us hope that the developers of OS/2 will take to heart the notion that security should be an integral part of a microcomputer operating system and include a true security subsystem in future versions of OS/2. In today's perilous world of viruses, worms, and Trojan horses, we need all the help we can get.

Cited references and notes

1. Fred Cohen, "Computer Viruses—Theory and Experiments," *Computers & Security*, Volume 6 (1987), Number 1, pp. 22-35.
2. However, there are now some viruses, such as the Invader or Plastique Boot virus, that attack both boot sectors and executable files. These are sometimes referred to as "flip-flop viruses."
3. For a more detailed description of how the Stoned virus functions, see *Computer Virus Handbook* by Dr. H. J. Highland, pp. 63-66, Elsevier Science Publishers, Ltd., 1990.
4. Brad Stubbs and Lance Hoffman, "Mapping the Virus Battlefield: An Overview of Personal Computer Vulnerabilities to Virus Attack," reprinted in Hoffman, ed., *Rogue Programs: Viruses, Worms, and Trojan Horses*, pp. 143-158.
5. Jan Hruska, *Computer Viruses and Anti-Virus Warfare*, p. 76, Ellis Horwood, Ltd., 1990.

WHY DOES TRUSTED COMPUTING COST SO MUCH?

Susan Heath Phillip Swanson Daniel Gambel
Grumman Data Systems
2411 Dulles Corner Park
Herndon, Va. 22071

Abstract

This paper discusses the relationship between the high cost of trusted computing and the way security requirements are stated in Request for Proposals (RFPs). This is done by introducing four types of trusted computer systems: Evaluated, Accredited, Tailored and Customized. These types of trusted systems along with their associated costs are discussed in detail and it is shown how systems transition from one type to the next. Finally, examples are given of how misstated or conflicting security requirements in RFPs lead to the development of each of these types of systems, thus driving up the cost of trusted system acquisition.

Introduction

During the 1970's and early 1980's, procurements of computer systems requiring security tended to be one-of-a-kind efforts. Acquisition was done via the standard Request For Proposal (RFP) process, and security requirements were developed from scratch for each effort. This process was both ineffective and inefficient.

It was ineffective because it soon became apparent that security requirements could not be met by adding on features to an existing Commercial-Off-The-Shelf (COTS) system. In order to provide the security that was needed, the system had to be designed from the beginning with security in mind and that meant a complete software design and development effort.

The process was inefficient because each major program became, in effect, another security research and development effort. There was little or no carry over from one program to another, and, since the products were not COTS, the entire effort became expensive.

This led to the development of the Trusted Computer System Evaluation Criteria (TCSEC)[1], followed by RFPs which now require that integrators meet the security requirements with TCSEC evaluated COTS products. While this is an excellent concept and could prove to be very cost effective, it is not usually achieved. This is because the requirements for a trusted system in a specific operational environment, as written in current RFPs, are usually over specific, generally in conflict with the TCSEC in some manner, and are seldom fully met by any generic COTS evaluated product.

While there are cases where legitimate operational requirements contradict TCSEC security requirements, most of the problem stems only from the wording used in RFPs. First, RFP's tend to contain

very specific security implementation requirements which are misstatements or contradictions to security design principles, due to lack of understanding of those security principles. Second, RFP's have begun to mandate compliance with multiple security policies, security functionalities, security guidelines and security methodologies which often contain conflicting security principles. This conflict arises due to the dynamic state of security technology and concepts between policy authors acquiring experience and the actual policy formulation. These two occurrences make it impossible for integrators to use COTS products to meet the requirements of RFPs.

A further negative outcome of this occurrence is that the specific methodologies contained in RFPs are, at best, based on the perceived current state-of-the-art and, often, are based on technology which is already considered outdated in the highly dynamic world of trusted system development. This ties the hands of the integrator from using the newest, lowest cost and best feasible solution.

In this paper we define four different types of trusted computer systems: evaluated, accreditable, tailored and customized and explain how system implementations require conceptual transition from one type to the next. Based on these definitions, we illustrate how the problems with RFPs, as stated above, lead to excess costs in providing trusted systems for secure environments. Examples are given from existing RFPs to reinforce the points being made.

Definitions

The following terms are used throughout the paper, in accordance with the National Computer Security Center (NCSC) definitions.

Certification Process. "The comprehensive examination of the technical and nontechnical security features of an automated information system and other safeguards, made in support of the accreditation process, that establishes the extent to which a particular design and implementation meet a specified set of security requirements." [2] The certifications support the accreditation process by establishing the extent to which particular designs and implementations meet security standards. Certifications are typically made for each aspect of the system including: Administrative, Procedural, Physical, Personnel, Communications, Emanations, and Computer Based (i.e. hardware, software, firmware).

Accreditation Process. "A formal declaration by the DAA that the AIS is approved to operate in a particular security mode using a prescribed set of safeguards. Accreditation is the official management authorization for operation of an AIS and is based on the certification process as well as other management considerations." [2] The accreditation process is intended to evaluate the adequacy of the security solution against the mission

need. This process includes an evaluation of the cost-effectiveness of implementing additional safeguards deemed necessary by the DAA.

Evaluation Process. The evaluation of the technical protection capabilities of COTS computer security products performed by the NCSC to establish conformance to a specific level (C2, B1, B2, etc.) of the TCSEC.[3]

Based on these terms, the following four systems are defined.

Evaluated System. An automated information system including hardware, software, and/or firmware that has been evaluated against, and found to be technically compliant, at a particular level of trust, with the TCSEC by the NCSC. Such systems are usually general purpose in nature and normally designed to provide the vendor with the widest possible market and are independent of specific environment.

Accredited System. An accredited system is an automated information system installed in a secure environment that is certified as meeting the computer security policy for a given mode of operation, in that specific environment, based upon the security requirements established by the DAA. The evaluation of policy adherence is based on various certifications of the system, supported by testing and additional DAA evaluation of the mission-need versus residual risk.

Tailored System. The term tailored system is used to mean a TCSEC evaluated system that is changed by adding trusted processes to the COTS Trusted Computing Base (TCB). Since Evaluated systems are designed for general use, they may not meet specialized security and unique operational requirements. They can, however, be modified, with some additional risk, by adding trusted processes to the TCB to meet the user specific requirements. This is the simplest and therefore, the most cost effective means to meet increased functional security requirements. If evaluation is required for a tailored system, these enhancements must meet the same documentation and engineering standards as required for the evaluation class of the original system. Normally, the additional evaluation need focus only on the trusted process and its interface to the TCSEC evaluated TCB.

Customized System. The term customized system is used to mean an evaluated system that has been significantly changed by modifying the implementation of the security model of the COTS TCB. In this case, an evaluated system is used as the starting point to develop a significantly revised system for which the operational requirements may contradict the TCSEC standards to which the original evaluated system was built. This customization requires modification to the implementation of the TCB as well as adding trusted processes.

Two additional terms are used throughout this paper, customer and

integrator. The term customer is used to signify the organization responsible for the daily operation and maintenance of the automated information system. This is the organization which will present the certification evidence to the DAA for accreditation to operate. The term integrator is used to signify the organization responsible for delivery and installation. This organization may be a government entity, a vendor, a manufacturer or a systems integrator.

Using these definitions we will examine how transitions are made from one system type to another and provide examples of why the transitions are necessary.

Transitions

Evaluated to Accredited. In order for an automated information system to process sensitive information within the government, it must be accredited. Therefore, while an integrator can start with a TCSEC evaluated system (or more likely a combination of TCSEC evaluated systems), the conceptual transition to an accredited system is required to specifically meet the mission of any customer processing sensitive information.

From the trusted system integrator's perspective, the transition from evaluated to accredited is essentially a documentation issue. The NCSC evaluation of a product states that it meets a specified level of trust when configured, installed, implemented and operated in accordance with the manufacturer's documentation set. This documentation may include but is not limited to the following: Security Policy, Security Model, Covert Channel Analysis, Security Features User's Guide, Trusted Facilities Manual, and Security Test documentation. These documents are generic in nature and provide only a starting point for the system specific documentation needed to meet the accreditation requirement. They do not address the operational environment of the system and they do not address the "glue" that holds a network system together. This includes the hardware and/or software used to connect the individual components together. In addition, these documents do not take into account the interface between the TCB and any nonsecure applications which may be added.

Therefore, supplementary documentation must be written which translates the generic COTS information from each component into the site specific system level implementation. This documentation set may include site specific versions of the above as well as the following additional documentation: System Security Plan, Configuration Management Plan, Risk Analysis, and Security Concept of Operations. This documentation must account for the integrators recommendations for secure operation and the customers operational constraints and requirements.

Naturally, there is cost involved with this documentation which can be incurred in one of two methods, explicit or implicit. Explicit refers to documentation that is generated by the integrator and is

considered deliverable under the original contract (and therefore included in the original cost). Implicit refers to documentation that is self-generated by the customer and is therefore, not part of the integrator delivery. The appearance is that by using the integrator to provide the system and developing the accreditation documentation with in house resources, a cheaper overall price tag is obtained. This appearance is very deceiving.

Implicit procurement of security documentation is similar to buying a VCR without an owner's manual. You may be able to figure out how to operate it, but you probably will not be taking advantage of all it's capabilities and you may even damage it by using it incorrectly. Additionally, it would take months of investigation to be able to write your own owner's manual. As a user of the system you do not know the precise details of operation and therefore are at a disadvantage over the manufacturer who actually assembled the components of the system.

This is particularly true in a secure data processing environment, where a mistake could be costly to national security. In most cases, the customer will realize their inability to produce the documentation after an initial attempt and then be forced to procure the documentation from an outside source, anyway. At this point, the cost will be higher than if procured with the product while in competition and will appear as an overrun which will be attributed to security. In reality, this additional cost could and should have been avoided up front by including the documentation in the original contract.

Evaluated to Tailored. In some cases, an evaluated system cannot meet all of the specified operational requirements. This occurs either because the evaluated system is too generic and doesn't support the required application or because there is an operational requirement for features or capability which is not foreseen by the security policy of the TCSEC generically evaluated system.

For example, if a TCSEC evaluated system uses a strict enforcement of the Bell-LaPadula model, information theoretically cannot be passed in any form from a higher to lower classification level. Information can however be passed from the lower level up to any higher level. This means that while the data can be passed up, the higher level user cannot request it, or acknowledge receipt, (electronically, at least) because nothing (including the request) can be sent from the higher level to the lower level. The solution to this problem is to develop a trusted process to perform the write down of the request. This trusted process then becomes part of the TCB and is trusted to perform write down only if the write is a specific type of information request.

A tailored system requires accreditation just as an accredited system did, but the accreditation process will be more extensive. This is because the evaluation given under the TCSEC applies only to the specific implementation on the specific hardware that was used by the NCSC during the evaluation. Therefore, the assurances

provided by the evaluation carry less weight due to the uncertainties surrounding the modification and the accreditor will want a more extensive review of the system.

A tailored system has an analogous documentation problem to an accredited system. However, more extensive documentation modifications will be needed to incorporate the functionality of the additional trusted software implementation. Also design documentation for the software modification and affected components may be required, as well as for the "glue" hardware and software.

In addition to the documentation costs, there are the costs of developing the new software which include design, configuration control, integration and test. The addition of trusted processes may require that the entire system be retested to ensure the new software does not hamper the original security mechanisms.

All of this adds up to a significant increase in overall cost for a tailored system. The cost escalation is based on two components. The first is the cost to tailor the evaluated system including configuration control and the second is the cost of documentation.

Evaluated to Customized. As with the tailored system, a customized system is one in which the operational requirements could not be met by a COTS evaluated system. In this case, however, in addition to adding trusted processes, the operational requirements necessitate modification of the evaluated system security kernel. Examples of such modifications are: a change to the label structure; the addition or modification of the implementation of an integrity model; or the addition of multilevel device drivers to handle hardware configurations other than those envisioned by the vendor (such as a network).

The accreditation process for a customized system will most likely be very extensive. While with a tailored system, there was some assurance because the changes were only additions to the TCB, a customized system involves a change to the foundation on which the original evaluation was based. The resulting system is just too different to place much reliance on the original evaluation.

The costs of developing new software escalate at an alarming rate due to the essential resources involved and of course the entire system has to undergo extensive testing. It does not take much imagination to see how this type of system becomes extremely expensive.

Examples

The previous discussion has shown how costs escalate in designing, managing and developing trusted systems. The documentation effort will be more extensive including more design and development specifications, rigorous configuration management, and a nearly complete rewrite of the manufacturer's documentation set. Given this information, why would an integrator propose anything other

than an evaluated system? In some cases there are legitimate operational requirements which make a tailored system necessary. In most cases, however, integrators are forced to propose customized systems due to poorly written, over explicit RFP requirements as stated earlier. The following examples illustrate this problem.

Example One

Problem: The RFP adds to an Orange Book requirement such as explicitly stating that a C2 system is required, but also stating that categories of information must be protected or labels are required.

Discussion: These are conflicting requirements. A C2 system does not provide for protection of categories of information or labels. These things are not provided until the B1 level of evaluation. In order to have a compliant proposal, an integrator must propose a C2 evaluated system and customize it to provide protection of categories of information or labels. In general, proposing a B1 system would be considered non-compliant and "gold-plated". The C2 customized system will, however, be much more expensive than the B1 evaluated system in terms of software development, documentation development, accreditation effort and time, all of which amount to more dollars.

Resolution: The Orange Book is not a chinese menu type document and is not meant to be invoked with explicit implementation policy requirements. You cannot take requirements from different levels of evaluation or conflicting policies and stick them together. They must be taken in general and in order. The RFP should state the requirements in terms of operational environment and let the integrator determine which level meets them. If a specific level is required then it should be stated in terms of "at least C2".

Example Two

Problem: The RFP requires that the system use only evaluated products or that the successful bidder submit the system to the NCSC for evaluation.

Discussion: This is a requirement which could possibly preclude the use of the most technologically advanced solution. Because the evaluation process takes years, there may be a more cost effective, security enhanced solution nearing completion of evaluation or the evaluated version may have been replaced by a better (not yet evaluated) release. This requirement will also be extremely expensive in terms of time and money for the integrator and therefore for the government. In most cases, if a COTS evaluated product is not used, it is because one is not available. This is usually because there is a specialized function needed to meet the requirement. Vendors who submit products to the NCSC for evaluation, plan to amortize the evaluation costs over time by selling the product in large quantities. Integrators are not in

that business and would not be able to amortize the costs. The full cost of having that product evaluated will be passed on to the Government.

Resolution: The RFP should require the specific evaluated product only where it makes sense. Products in the evaluation queue or updated versions of evaluated products should be acceptable substitutes, where necessary. The evaluation requirement should be left out all together.

Example Three

In addition to NCSC evaluated product rating requirements, the RFP specifies the security requirements, including implementation techniques, in explicit detail and in such a way that they conflict with DoD security policy or the Orange Book such as:

Provide system generation features such that operating system and TCB elements can be inserted, deleted or replaced on-line without requiring a complete regeneration of the system or the TCB.

Discussion: These features are inconsistent with a trusted operating system. One of the Orange Book criteria is System Integrity which is required at all levels of evaluation. Even at the C1 level, the requirement for System Integrity is "The TCB shall maintain a domain for its own execution that protects it from external interference or tampering (e.g. by modification of its code or data structures)"[1]. Clearly the RFP requirement above is in violation of this. A COTS TCB would have to be broken at great cost to customer to provide this functionality. It makes no sense to require a trusted system and then require that its trustability be rendered useless.

Resolution: It is best not to specify security implementation details because problems like the above often occur. The Orange Book is very specific about how trusted systems must function, yet it does not specify a solution. This allows the integrator to review all existing technology and come up with the best solution for the particular environment. Again, security requirements should be stated in terms of the Orange Book.

Example Four

Problem: The RFP requires evaluation of the proposed system by NCSC, requires this within a certain amount of time after contract award, and requires a monetary penalty for not meeting this requirement.

Discussion: There are several problems with this. The first is similar to example two above. The NCSC evaluates vendor products, not unique applications of trusted products. Second, committing to having a product evaluated within a certain time frame is playing Russian Roulette. To a large extent the evaluation

schedule is government controlled. This represents great risk for the integrator because it requires them to be responsible for a process over which they have only partial control. The only way to be certain of meeting this requirement is to propose components which are already evaluated. Finally, the government controls which products are accepted for evaluation rendering the vendor helpless to control penalties.

Resolution: There are several possible resolutions. The first is to eliminate requirement that the system be evaluated within a certain time frame. Second it could be stated that the integrator will not be held accountable for government controlled action or third the requirement could be changed such that an Memorandum of Understanding (MOU) is required to be initiated within a certain time frame.

Example Five

Problem: The RFP has requirements which conflict with each other such as requiring a B2 system operating in the System High mode.

Discussion: A B2 system is considered synonymous with the multilevel mode of operation although it can be configured to run in the System High mode. The additional cost and assurance of a B2 system may be wasted on a System High implementation.

Resolution: Again, specify the operational requirements and let the integrator decide which level of system meets these requirements.

Example Six

Problem: The RFP contradicts itself by having different security requirements in different parts such as specifying System High mode one place and Multilevel mode another place or requires these modes in a phased approach, such as System High in phase one and Multilevel in phase two.

Discussion: The only way to meet this is to propose a system which can support the more stringent requirement, multilevel (i.e. B2 level) because a system cannot migrate from one evaluation class to another. If multilevel operations are not required, this represents a significant cost escalation over what is needed.

Resolution: Ensure the RFP does not contain conflicting security requirements.

Conclusion

Trusted computer systems are to some extent more expensive than nontrusted computer systems. There is no way to get around this. However, the cost of trusted system implementation does not have to be uncontrolled or exorbitant. The Orange Book provides a method of standardizing trusted computer system design but its

principles must be followed exactly or their advantages are reduced. System integrators are in the business of understanding this process and knowing the intricacies of trusted systems. They must be given the leeway to provide an architecture that is the most cost effective, state of the art solution. RFPs which contain conflicting security principles, very specific design details, and conformance with multiple security guidelines undermine the integrators ability to do this and invariably drive up the cost.

References

1. Department of Defense Trusted Computer System Evaluation Criteria, DoD 5200.28-STD, December 1985.
2. Glossary of Computer Terms, National Computer Security Center Trusted Guideline 004 (NCSC-TG-004), Version-1, October 21, 1988, National Computer Security Center, 9800 Savage Road, Fort George G. Meade, MD 200755-6000.
3. Information Systems Security Products and Services Catalogue, National Security Agency, April 1990

Executive Summary

Panel: Acquiring Computer Security Services and Integrating Computer Security and ADP Procurement

**Dennis Gilbert, NIST, co-Moderator
Barbara Guttman, NIST, co-Moderator**

**Panel Participants
Nickilyn Lynch, NIST
Nander Brown, RTC
Gary Oran, FEMA**

**Merv Stuckey, Census
Victor Marshall, Booz-Allen & Hamilton/NASA
E. Taylor Landrum, Grumman Data Systems**

The Computer Security Act of 1987 and other federal regulations place responsibility on federal organizations to protect automated information and the means of processing it. Accordingly, agencies must perform many computer security functions throughout the system life cycle. They must also incorporate computer security as early as possible in the system development and system acquisition processes. However, agencies lack a general understanding of how to describe these activities. Federal organizations could more effectively carry out computer security responsibilities if they had access to such descriptions. To provide support, NIST sponsored two interagency working groups of federal and industry specialists to develop two documents with the descriptions. The groups represented the fields of computer security, procurement, and information resources management.

The session presents the results of the two working groups efforts: the first effort addresses descriptions of computer security services resources; the second effort addresses computer security and ADP procurement.

The document from the first group presents sample statements of work (SOWs) for several computer security activities. Organization staff and government contractors can use these as a basis for understanding each described activity. The sample SOWs should promote more consistent, high-quality computer security services. Agencies could use the descriptions to either contract for the services or get them from within the organization.

The document from the second group addresses computer security in automated data processing procurements. It describes how to integrate computer security in all four phases of the procurement cycle: planning, solicitation, source selection, and contract administration and closeout. Its use helps in acquiring information processing resources with the most cost-effective security.

Both documents are intended to be used with agency or GSA guidance on procurement and computer security.

In this session, working group members place the documents in context, describe the contents, discuss the more significant issues, and give advice on how to use them. The session presents a unique opportunity to explore an area that is often a source of confusion.

Executive Summary

Compartmented Mode Workstation (CMW) Program Overview

Mr. Steven T. Schanzer, Moderator
Defense Intelligence Agency

Panelists:

Dr. Stuart Milner Defense Intelligence Agency
Mr. Scott Wiegel ADDAMAX
Mr. Paul Cummings DEC
Mr. Gary Luckenbaugh IBM
Mr. David Arnovitz Secureware
Mr. Gary Winiger Sun Microsystems

What is a CMW? *(see figure 1)*

- **A Trusted Operating System with a Trusted Window Management System (i.e., X Window System)**
- **A Workstation capable of supporting Compartmented Mode Operations (Not all persons with access to the system are "read-on" for all compartments processed)**
- **A CMW will provide separation of Sensitive Compartmented Information (SCI) compartments, subcompartments, Special Access Programs, etc.**
- **A CMW will provide the Trusted Computing Base upon which to access information on hosts that operate at different security levels**

Who needs a CMW?

- **Users who require a workstation with a Trusted Computing Base**
- **Users who require the ability to reliably separate different levels of information on a single workstation**
- **Users who require simultaneous access to hosts operating at different security levels**
- **Users who require sanitization, decompartmentation, and/or downgrading capabilities**

**"One Analyst, One Secure Workstation
with Multiple Connections
at Different Security Categories"**

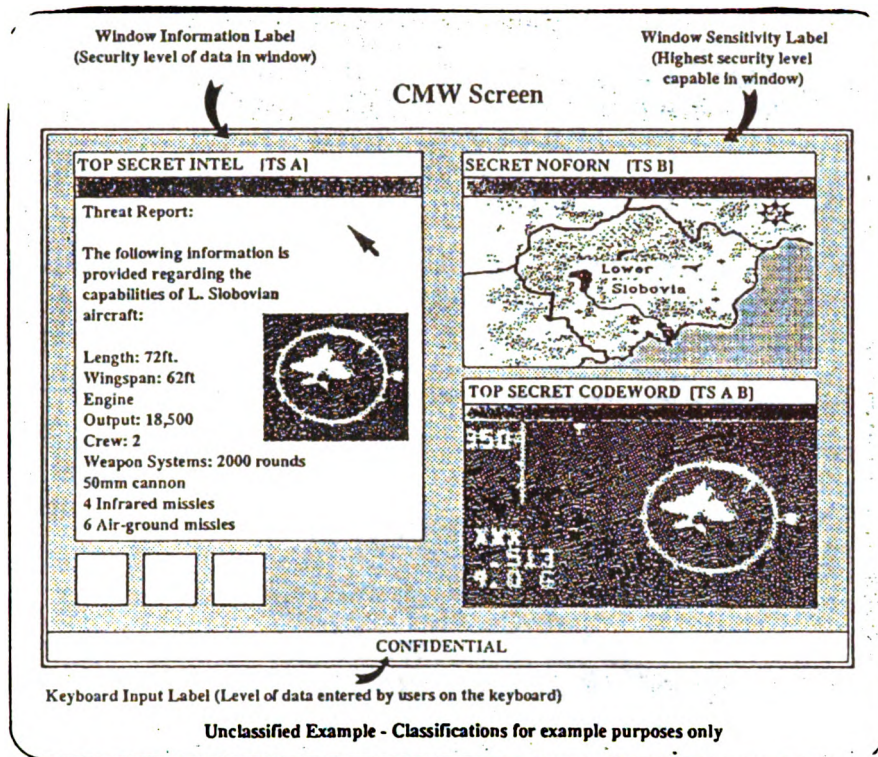


Figure 1
CMW Program

Technical:

- Meets or exceeds all requirements for compartmented mode operations specified in DDS-2600-5502-87
- Meets or exceeds all requirements for "Labeled Protection" (B1) criteria under DoD5200.28STD (TCSEC)
- A Trusted Window Management System, providing user interfaces (windows) at multiple security levels

Programmatic:

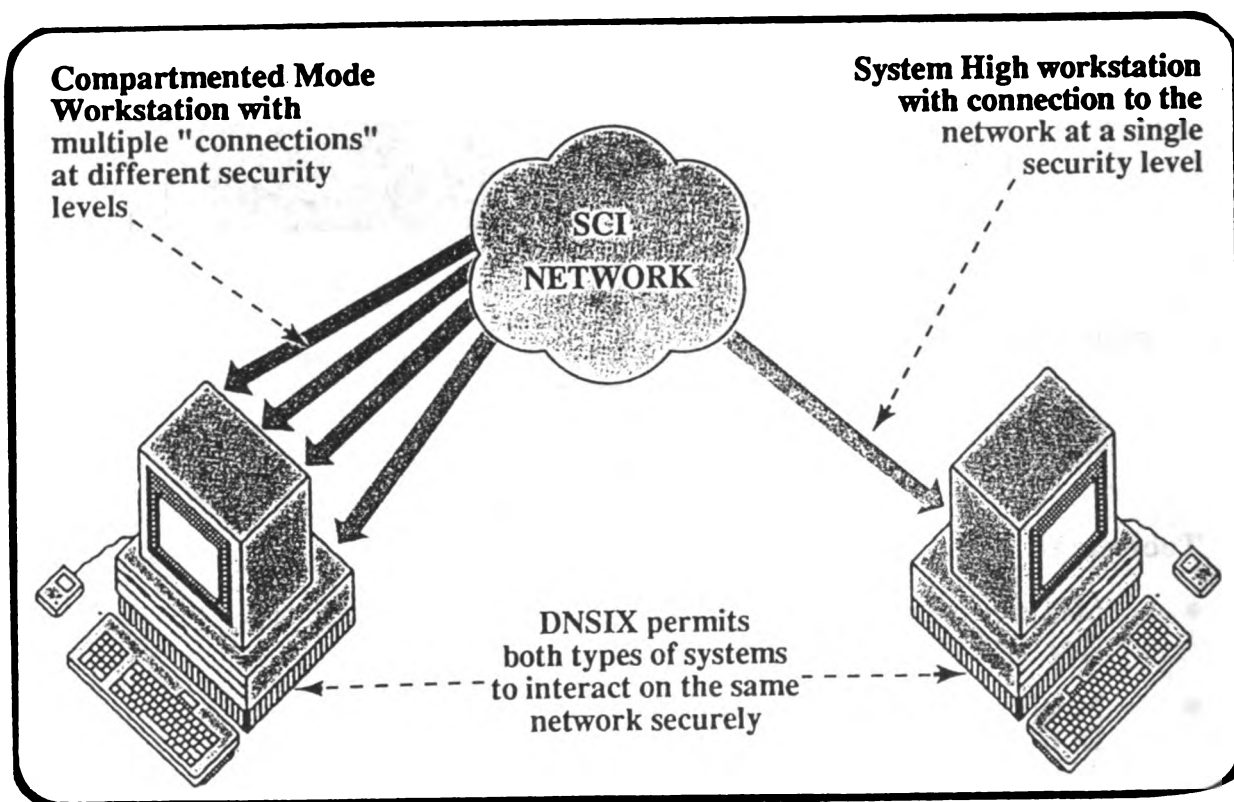
- Currently, five DIA contracts to develop commercial CMWs:

ADDAMAX	(Zenith 386, System V, OPENLOOK)
DEC	(VAXStations, ULTRIX, MOTIF)
IBM	(RS 6000, AIX, MOTIF)
Secureware	(MacIntosh IIx, A/UX, MOTIF)
Sun Microsystems	(SPARC, SUNOS, OPENLOOK)
- Joint DIA - NSA Evaluation against both the DIA CMW Requirements and the TCSEC

CMW Program Extensions

Department of Defense Intelligence Information System (DoDIIS)
Network Security for Information Exchange (DNSIX):

- Meets or exceeds all requirements for compartmented mode network operations on SCI networks
- CMWs provide security separation on the workstation, DNSIX provides security separation over the network



Trusted Applications:

- A CMW will provide a programming interface for augmenting the Trusted Computing Base
- Trusted Applications will provide additional security functionality to users

Additional Information:

Defense Intelligence Agency
ATTN: DSO-3A
Washington DC 20340-3434

Executive Summary

THE COMPUTER EMERGENCY RESPONSE TEAM SYSTEM (CERT SYSTEM)

E. Eugene Schultz

Lawrence Livermore National Laboratory
P.O. Box 808, L-303
Livermore, CA 94550
gschultz at CHEETAH.LLNL.GOV

Richard Pethia

Software Engineering Institute
Carnegie-Mellon University
Pittsburgh PA 15213-3890
rdp@cert.sei.cmu.edu

Abstract

This paper describes CERT System, an international affiliation of computer security response teams. This affiliation's purpose is to provide a forum for ideas about incident response and computer security, share information, solve common problems, and develop strategies for responding to threats, incidents, etc. The achievements and advantages of participation in CERT System are presented along with suggested growth areas for this affiliation. The views presented in this paper are the views of one member, and do not necessarily represent the views of others affiliated with CERT System.

The Formation of CERT System

Following the Internet worm in 1988, a number of organizations created, or expanded their existing security groups to create, computer security incident response teams. Each team focused on a particular user community and worked with its community to respond to incidents when they occurred. Some teams also became proactive and worked with their communities to raise the awareness of security issues, provide guidance on improving the security of operational systems, and identify and eliminate vulnerabilities to lower risk.

Even in the early weeks of operation, it became apparent that cooperation, collaboration and coordination across the various teams would be necessary to effectively deal with the global problem: intruders taking advantage of the international meta-network of connected computer networks, conferencing systems, and communications systems. While individual teams focused on their own communities and provided support that was sensitive to the culture, needs, policies and regulations of those communities, they were faced with intruders who ignored the boundaries and used multiple attack vehicles to exploit vulnerabilities that were common across the networks.

The model that emerged presumed the creation of multiple emergency response teams with each team focused on a particular user community. The various teams would collaborate and pool resources when necessary to respond to incidents, share vulnerability information, and develop tools and techniques that would benefit all

groups. This distributed model was tacitly accepted by several groups and various teams began cooperating with others on an as-needed basis. For example, during the WANK-OILZ worm attack in 1989, the Department of Energy's Computer Incident Advisory Capability (CIAC) cooperated extensively with NASA's Space Physics Analysis Network (SPAN) and the Defense Research Projects Agency's sponsored Computer Emergency Response Team Coordination Center (CERT/CC) to deal with the problem. As the CIAC and SPAN teams worked to develop immunization and eradication scripts to combat this worm, the CERT/CC team worked to prepare advisory and status information and to alert members of the network communities that were potentially under attack.

At a post-mortem meeting on the WANK-OILZ incident several weeks after the cessation of the worm attacks, representatives from the CIAC, SPAN, SPAN-France, and CERT/CC teams determined to take additional steps to strengthen the cooperative effort to share information among these response teams, and, if needed, to mutually aid one another during incidents. Interest in this cooperative arrangement spread rapidly among other response teams.

In November, 1990 an operational framework for an affiliation of 11 incident response teams (ten from the U.S.A., one from France) was approved by every representative of each response team. This affiliation, presently called CERT System, was formed for a number of purposes. One was to provide a forum for participating response teams where ideas, methods of responding to incidents, etc. could be exchanged and evaluated by peers with similar job responsibilities and experiences. Another purpose was to share information about current attacks, vulnerabilities, etc. Still another purpose was to solve common problems, such as obtaining cooperation from vendors in closing vulnerabilities in vendor products. Finally, this organization was formed to plan future strategies for dealing with computer security threats, coordinating with U.S. Government investigative agencies, etc.

An operational framework specifying goals, types of participation, organization of CERT System, meetings to be held, requirements, and operational activities, procedures and policies was approved last year. Structured as a cooperative activity, there is no lead organization. Members of the CERT System are accepted through a nomination and acceptance procedure and, once accepted, are able to vote for candidates for a steering committee and secretariat. The steering committee is responsible for general operating policy and procedures, and is supported by the Secretariat that assumes additional coordinating activities. Other activities are carried out by working groups that are created by the steering committee as needed to work on priority projects or deal with specific problems.

Issues to Be Addressed

This paper addresses a number of issues concerning CERT System and its activities. What has this organization of response teams accomplished so far? Where, if anywhere, has this organization fallen short of its goals, and what must it do to

accomplish all of the purposes enumerated in the CERT System operational framework?

Accomplishments

Forming an affiliation of response teams has been, in and of itself, a major accomplishment. The 11 response teams in this affiliation work for a wide variety of agencies and/or institutions, have a diversity of purposes and operating environments, and have differing expectations with respect to CERT System involvement. The effort of individuals from the National Institute of Standards and Technology in preparing the CERT System Operational Framework (1990) has resulted in an excellent structure and effective procedures for participation in CERT System.

During its short existence, CERT System has already established a useful role in the incident handling community. First, this organization has been an impetus for establishing communication among the many incident response teams. What has resulted is a forum for discussing a wide variety of issues, including working with vendors, dealing with vulnerabilities, determining what specific sites/organizations constitute a particular response team's constituency, recognizing signatures of current network intrusions, etc. This forum has also helped new teams learn about forming and operating an incident response effort on the basis of other teams' lessons learned communicated through this forum. In at least one instance, there was cooperation between at least four response teams in a series of sensitive intrusions involving several Government agencies and other academic and commercial sites. CERT System helped pave the way for cooperation by providing a way for members of the different response teams to become acquainted and establish communication before the crisis situation arose. Also, because there were agreed upon procedures within CERT System for sharing sensitive information, there appeared to be very little resistance in sharing information between response teams.

CERT System also has become a vehicle for sharing vulnerability information and information about network intrusions and probes. There is a mechanism for distributing information through CERT System before a response team releases this information to its own constituent community. This gives response teams an early alert about issues (e.g., network intrusions and vulnerabilities) that may possibly require action. This aspect of CERT System operations seems especially advantageous to response teams with smaller constituencies; these teams often receive less information from technical personnel within their constituencies than do teams with larger constituencies.

Growth Areas

CERT System is a fledgling organization with numerous areas in which it must grow to provide leadership and direction to computer security incident response efforts. Interaction across member teams has been effective in many cases, but more work must be done to develop fast, secure channels of communication. In addition,

efforts must be made to develop a better understanding of the roles and jurisdictions of various law enforcement agencies to allow working relationships that are effective at dealing with even international problems. In addition, a critical next step involves increased interaction with vendor communities. Many vendors have enhanced their ability to correct reported system vulnerabilities and provide their customers with corrected software, but additional work must be done to develop software correction and distribution mechanisms that are even more timely and cost effective. CERT System must initiate and continue these dialogues as a first step for facilitating the interaction across these communities who must cooperate in responding to computer security incidents.

A second growth area concerns sharing information within CERT System. Response teams freely exchange bulletins which warn of some threats or announce the availability of software that eliminates vulnerabilities, but more work must be done to build mechanisms for more timely exchange of information about vulnerabilities, threats, and network attacks. Exchange of vulnerability information is a very troubling area. As individual teams identify problems and drive forward for solutions, their narrowing focus sometimes excludes the communication that could assist other groups. This leads to duplication of effort and frustration as teams discover they are chasing problems that are already being worked. To resolve these problems, the CERT System must set up a mechanism to assign responsibility for resolution of particular problems and to communicate the assignment to all members. To facilitate this exchange, the CERT System should adopt a secure mail facility that authenticates the sender of the message and assures the integrity and protection of the sensitive data. It is also important to improve the interaction with the classified community and to insure that all vulnerabilities found in the unclassified community are reported to the classified world.

CERT System members have become painfully aware of the difficulty of building and maintaining trust across organizations when dealing with security issues; especially with actual incidents. While all CERT System members recognize the importance and utility of sharing incident data, they each face a reluctance on the part of their communities to release data as incidents are in progress. Withholding information lowers the likelihood that an investigation will be compromised or that an organization will be embarrassed, but raises the probability that additional sites will be affected. The tension between the need to disseminate information and the desire to withhold it will only be resolved over time as individual groups take the risk of releasing the information and CERT System members demonstrate their ability to handle it discretely. Each CERT System member must be especially sensitive to the fragility of trust and must be vigilant to insure none of its actions diminish it.

Another challenge CERT System faces concerns the current level of participation within this organization; participation by existing members as well as the addition of new members. Most of the affiliation's steering committee members attend steering committee meetings, but, with some notable exceptions, very little activity occurs between meetings. Individual members, focused on meeting the needs of their constituents, have difficulty devoting the time and resource necessary to

further the cooperative effort. In addition, the organization does not yet have enough representation from the commercial sector to allow it to develop effective solutions to certain problems. Even more important, there is only very limited representation from outside the United States. Many networks and communications systems are expanding rapidly outside the United States with dramatic increases in levels of connectivity. International representation is vital to allow the organization to deal with threats and attacks that are international in scope. Also, international representatives would have the ability to bring the response team perspective to the policy makers who are sure to emerge as the networks grow in importance and size. Although initially comprised of representatives from response teams that have proven to be effective in their arenas, CERT System needs to more actively promote greater membership and participation, and should examine mechanisms it might use to insure resources are available to work the cooperative efforts.

Finally, CERT System must strive to accurately represent the nature and constituency of this organization to others, and must actively work to remove misconceptions surrounding this organization. For example, contrary to what the media has sometimes depicted, CERT- System is not an organization of Government agencies. Although some response teams within this organization represent Government agencies, others, such as CIAC and CERT/CC, do not. Another widely spread misconception results from the name "CERT System." This name too often leaves the impression that the CERT/CC team from Carnegie Mellon University somehow directs the efforts of the other response teams. Still another misconception to correct is that this organization exists to regulate the activity of response teams. Through timely press releases and a careful choice of a name for this affiliation, CERT System can remove these misunderstandings, and become a more effective agent for disseminating accurate and useful information to the computer security arena as well as others.

Summary

In summary, CERT System is an affiliation of incident response teams formed for purposes such as promoting cooperation and information sharing within teams, facilitating problem solving, and providing a forum for discussing issues. Although new, this affiliation has already realized success in a number of areas, but especially by raising the level of communication between the various response teams. CERT System must also address a number of problems associated with its existence to provide bona fide leadership to the incident handling community.

Note

Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract W-7405-Eng-48.

Reference

CERT System, CERT System Operational Framework, 1990.

Executive Summary

PANEL: Computer Security Management and Planning

Christopher Bythewood, NCSC, Moderator

Jon Arneson, NIST

Richard Carr, NASA

Dennis Gilbert, NIST

Irene Gilbert, NIST

Barbara Guttman, NIST

Gerald Lang, DVA

Ed Springer, OMB

The Computer Security Act of 1987 (the Act) places major emphasis on computer security management and planning. This session focuses on this subject from several perspectives.

The Act directs federal agencies to establish minimum acceptable security practices for federal computer systems that contain sensitive unclassified information. Initially under the Act, federal agencies identified such systems and submitted security plans to a joint NIST/National Security Agency (NSA) review team for advice and comment. Based on this experience, OMB, NIST, and NSA evolved a strategy for guiding federal agencies in identifying and protecting sensitive information systems. This strategy emphasizes implementing computer security plans. Current OMB instructions on the Act provide for agency assistance visits by OMB, NIST, and NSA staff to provide direct comments, advice, and technical assistance about how the agency is implementing the Act. Several agency assistance visits have taken place. This session reports on the agency assistance visits and the learnings gained from them, from both central agency and visited agency perspectives.

Federal agencies, and other organizations, increasingly accept that computer security must be addressed in the earliest stages of system development and system acquisition. In fact, these concerns must be attended to throughout the system life cycle. Two recent NIST-sponsored interagency working group efforts looked at these areas: one covers integrating computer security and ADP procurements; the other covers how to obtain computer security services, either by contracting out or from within the agency's resources. The working groups consisted of federal and industry specialists in the fields of computer security, procurement, and information resources management. The session will present the results of the two working groups.

Computer security awareness and training is another area of management responsibility identified in the Act. The session also covers the management of and planning for this vital area.

Commercial organizations can also learn and benefit from federal management and planning experience in implementing the Act. While federal managers must satisfy specific regulatory conditions, significant elements of their data processing and security requirements and perspectives are similar, or directly analogous to their commercial counterparts.

Executive Summary

Cracking the Cracker Problem

Dorothy E. Denning, Moderator
Georgetown University

Panelists:

John Perry Barlow, Electronic Frontier Foundation

Matt Bishop, Dartmouth College

Donald P. Delaney, New York State Police

Mitchell Kapor, Electronic Frontier Foundation

Donn B. Parker, SRI International

This panel will address the problem of hackers who break into computer systems. The questions to be addressed include: How serious is the problem? What will the problem look like in the future? What can be done about it? To what extent can technology solve the problem through better security, including systems that are designed to be secure, security checkers, intrusion detection systems, and strong authentication? To what extent do the DoD criteria for trusted systems lead to systems that cannot be cracked? To what extent can law enforcement help solve the problem? Are strong penalties an effective deterrent? What should be done about teaching ethics and how effective is it likely to be?

Executive Summary

The Role of Technology in the Cracker Problem

Matt Bishop

Department of Mathematics and Computer Science
Dartmouth College
Hanover, NH 03755

The "cracker problem" is the problem of young computer crackers breaking into computer systems. To solve this problem, computers must be better protected than they are now, and crackers must be discouraged from cracking systems. There are two aspects to this, the technical and the human.

Technologically, excellent mechanisms (such as strong authentication techniques, security checkers, and intrusion detection systems) and strong criteria (such as the Trusted Computer Security Evaluation Criteria) exist to improve the security of computer systems; while they are by no means perfect, when installed and used correctly, they will either foil or detect most attacks -- and all those that fall into the class of "cracker" attacks.

The catch is that they must be installed, maintained, and used correctly. This aspect of the cracker problem is often overlooked. If the security mechanisms are too cumbersome for users, if they are difficult or time-consuming to install and too complex to maintain, they will either not be used or will be used incorrectly, leading to non-secure sites. The more dangerous situation is when the tools are installed, maintained, or used incorrectly, as their existence will give management and users a false sense of security.

When an attacker attacks a secure system, the simple lines of attack will fail. In this case, an attacker could either abandon the attack, deciding other sites would be more fruitful for the effort, or could take the challenge of trying to crack such a secure system. The first possibility suggests that the technology has not solved the cracker problem, but merely shifted the sphere of attack; the second argues that the problem may not have abated at all.

Thus, human issues must be factored into the development and deployment of computer security mechanisms. For this reason, the technology should not be seen as a solution to the cracker problem. It should be seen as an aid to implementing a human solution.

Acknowledgement: Thanks to Maria Gallagher for very helpful discussions.

Executive Summary

PANEL: ELECTRONIC DISSEMINATION OF COMPUTER SECURITY INFORMATION

Marianne Swanson, Moderator, NIST

Abstract

Computer security vulnerabilities and remedies are routinely provided to the public through computer security bulletin boards and electronic forums. Computer security managers should be aware of the oasis of computer security related information that is available through their standard ASCII terminal or their personal computer with communications capability.

Introduction

The purpose of this panel is to inform the audience of several sources of computer security related information that are available to the public. The types of information that are on the systems as well as how to subscribe or obtain the information is discussed by the panel members.

Panel Members

Marianne Swanson

Systems Operator

NIST Computer Security Bulletin Board

The National Institute of Standards and Technology's Computer Security Division maintains an electronic bulletin board system (BBS) focusing on information systems security issues. The security bulletin board is intended to encourage sharing of information that will help users and managers better protect their data and systems.

Cindy Hash

System Administrator

DOCKMASTER

The National Computer Security Center has developed an unclassified system, DOCKMASTER, which provides a focal point for interacting and exchanging computer security related ideas amongst its users. DOCKMASTER provides online access to the Information Systems Security Products and Services Catalogue and offers "forums" where users can attend online meetings. The "MAIL" feature allows users to send or receive message to and from users of government networks.

Peter G. Neumann

Moderator

***Forum on Risks to the Public in Computers and Related Systems
(RISKS FORUM)***

RISKS FORUM is an electronic publication located on the Internet that is generally about risks that pertain to use of high technology. Much space is dedicated to risks with computers, such as with the use of computer technology in aviation, medicine, the military, credit agencies, and so forth. The discussions are usually entertaining and often include interesting (and sometimes frightening) anecdotes about problems encountered with the use of computer technology in society.

Kenneth van Wyk

Moderator

VIRUS-L

VIRUS-L is a moderated mailing list with approximately 1600 direct subscribers world wide. The mailing list is dedicated to information about computer viruses, including Macintosh, PC, Amiga, and Apple, as well as others. VIRUS-L is an e-mail forum for Internet users that generally includes useful information such as references to repositories of anti-viral software, publications, and other items.

WHAT CAN DOCKMASTER OFFER YOU?

Cindy Hash
9800 Savage Road
Ft. Meade, MD 20755-6000
(301) 859-4509

The National Computer Security Center established DOCKMASTER in 1985 to disseminate computer security information to a variety of interest groups. These groups include Government organizations, industry, academe, as well as individuals who have an interest in computer security. In the last 6 years, the user population has grown from 400 users to over 2500 users. Several factors have been attributed to the rapid growth: EMAIL, forums, an interest in how security is practiced, and cost free access to other computer security professionals.

DOCKMASTER was designed to be and is a Computer Security Showcase. We strive to be a leader in implementing security features. The operating system is MULTICS, an evaluated B2-product. One subsystem is the Watchword Generator, which provides additional identification and authentication. The DOCKMASTER administration group provides many services to users. Security is a serious responsibility to both the administration and operations group. Audit trails are reviewed daily. Users are called if any anomalies are found. We also encourage users to call us on an 800 number to report security problems or to ask questions. Our actions have caused us to be written in Cliff Stoll's book, "The Cuckoo's Egg." Some people have called DOCKMASTER the most secure system on the Internet.

Several mechanisms are used for access to DOCKMASTER; users in the National Computer Security Center are connected directly; users in the Baltimore, Maryland area can call through the local C&P Telephone Company; users on the MILNET can also use TYMNET (paid by the National Computer Security Center) and the Internet to connect to DOCKMASTER. Users can also use TAC (Terminal Access Cards) where TAC Access is available.

Due to the "free" connectivity, the operations staff also reviews users every 6 months for continued access to DOCKMASTER. We have removed over 3500 users in the last 6 years as well. Users are removed if their accounts are inactive; they are removed if they change jobs without revalidating their continued need for access to DOCKMASTER.

DOCKMASTER disseminates information through electronic bulletin boards called **FORUMs**. A **FORUM** can be described as a public mail box facility. There are over 60 public **FORUMs** which cover a vast number of computer security related topics. Examples of the public **FORUMs** are:

Bulletin Board	General discussion
Comms	Electronic Communications Questions
EPL	The Evaluated Products List
Questions	Help with the Multics Operating System
Conferences	Information Security Courses and Conferences announcements
INFOSEC	A compendium of bulletin boards providing information on the Industrial TEMPEST Program, the Endorsed Cryptographic Products, Endorsed DES Products, Protected Services, and General INFOSEC Information
RISKS	ACM sponsored out of Stanford Research Institute
VIRUS-L	CERT sponsored from Carnegie Mellon Institute
Security Discussions	General discussion of security related issues.

Some of the **FORUMs**, like **RISKS** and **VIRUS-L** are read-only and are sent to us by the moderator. Some are generated by the National Computer Security Center, like **EPL**. Others are fully interactive with all the users of **DOCKMASTER**. Subgroups of users also have the ability to limit access to **FORUMs** allowing private "bulletin boards".

DOCKMASTER also provides an electronic mail (Email) facility which allows the exchange of information among professionals in the Computer Security field. Email can be exchanged with not only other **DOCKMASTER** users but other users on the **MILNET** and many Internet sites.

Users of **DOCKMASTER** can be designated as restricted users or unrestricted users. Restricted users can choose between a limited menu subsystem (**INFOSEC**), or a limited subsystem (**Catwalk**). **INFOSEC** users remain in a tightly controlled menu driven environment. **Catwalk** users have complete access to the Email facilities, to public forums, and certain features like the editors. These users may not execute software which has not been approved by the **DOCKMASTER** Staff. Non-restricted users are users who are not on **INFOSEC** or **Catwalk**. These users have the above privileges as well as the ability to program and execute non-system programs. Project Administrators provide system related assistance to the non-restricted users. Restricted users account for approximately 55% of the user community on **DOCKMASTER**.

Our goal is to provide service 24 hours a day. Our facility is manned from 7:00AM to 5:00PM Monday through Friday and from 8:00AM to 4:00PM on the weekends. Staff can be reached during normal duty hours at (301) 859-4360 or (301) 850-4446, or for those outside the Maryland area, (800) 336-DOCK. Requests for a user account on **DOCKMASTER** can be handled at these numbers.

Executive Summary

TOWARDS MUTUAL RECOGNITION OF SECURITY EVALUATIONS

**Andrea Arnold
Cornelia Persy
Gottfried Sedlak**

**c/o VDMA
Attn: Hans-Joachim Bierschenk
Lyoner Strasse 18
W-6000 Frankfurt 71
Germany**

Abstract

We work towards mutual recognition of security evaluations that are performed under different criteria. The approach is:

- to modularize different criteria to a level of granularity that allows their comparison
- to compare the modularized criteria
- to merge them into a superset.

We demonstrate the feasibility of our concept with examples taken from the Trusted Computer System Evaluation Criteria TCSEC and the Information Technology Security Evaluation Criteria ITSEC.

BACKGROUND

Security evaluation criteria for information technology systems have been developed in the United States and Europe since the early 80s. The Trusted Computer System Evaluation Criteria TCSEC, known as the Orange Book, was published by the US Department of Defense in 1983 (updated in 1985). Other countries followed the example set. In 1990, France, Germany, the Netherlands, and the United Kingdom in a concerted effort published the Information Technology Security Evaluation Criteria ITSEC, known as the Harmonised Criteria. We concentrate on these two criteria catalogs.

Although we assume that the reader is familiar with both sets of criteria, we list the major differences:

- TCSEC cover confidentiality primarily - ITSEC include integrity and availability
- TCSEC combine classes of functionality and assurance - ITSEC define functionality and assurance independently
- TCSEC focus on operating systems primarily - ITSEC address products and systems.

Currently, evaluations are still done separately in different countries with no mutual recognition of the resulting certificates.

OBJECTIVE

We developed a concept to support mutual recognition of security evaluations that are performed under different criteria schemes. The concept is independent from criteria catalogs. It supports evaluation consistency and improves objectivity for evaluations done by different evaluation authorities. This work was done by the VDMA/ZVEI¹ Working Group based on a suggestion by the EUROBIT² Industrial Policy Group

1.VDMA (Verband Deutscher Maschinen- und Anlagenbau) and ZVEI (Zentralverband Elektrotechnik- und Elektronikindustrie) are German business associations.

2.EUROBIT (European Association of Manufacturers of Business Machines and Information Technology Industry)

The concept describes a bi-directional mapping between the TCSEC and the ITSEC criteria catalogs. The criteria catalogs were taken as is, redefining them was not an objective. The concept is applicable to other criteria catalogs as well and allows for extension to more than two criteria catalogs.

CONCEPT DESCRIPTION

Building a superset of the criteria catalogs seemed to be the best way to support mutual recognition. Each catalog's profile can be mapped easily to the superset. We modularized the functionality and assurance aspects of each catalog, compared, and finally merged them into a superset (see Figure 1).

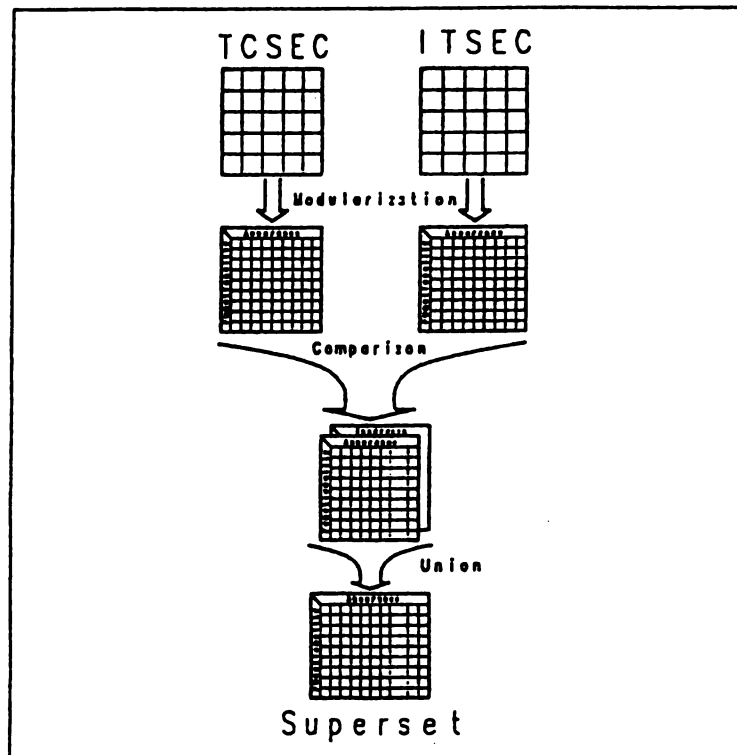


Figure 1. Concept

Our superset example was developed in four steps:

1. MODULARIZATION OF TCSEC:

The TCSEC, in part, were modularized to a level of granularity that allows comparison with the ITSEC.

Functionality:

We modularized the security function group Audit TCSEC.

Assurance:

Although the TCSEC separate the assurance and documentation aspects TCSEC, Summary Chart, p. 109 we treated both as assurance, as is done in the ITSEC, and modularized them completely.

2. MODULARIZATION OF ITSEC:

The ITSEC, in part, were modularized to a level of granularity that can be compared with the TCSEC.

Functionality:

We modularized the security function groups Accountability and Audit ITSEC. Both together correspond to Audit TCSEC.

Assurance:

The ITSEC distinguish two assurance aspects, the correctness and the effectiveness. We completely modularized the correctness ITSEC, Assurance - Correctness, pp. 23. But we did not consider the effectiveness, which still needs to be addressed.

3. COMPARISON:

The modularized criteria were compared to find the differences in meaning.

We compared functionality and assurance aspects of TCSEC and ITSEC. The comparison was easier for functionality than for assurance. For assurance it was sometimes difficult to find the corresponding aspects.

Functionality:

We used Audit TCSEC / Accountability and Audit ITSEC of steps 1 and 2.

Assurance:

Due to resource constraints only a subset of the modularizations of step 1 and step 2 was considered for the assurance comparison:

- Covert Channel Analysis TCSEC / Vulnerability Analysis ITSEC
- Configuration Management TCSEC / Configuration Control ITSEC
- Security Features User's Guide TCSEC / User Documentation ITSEC
- Trusted Facility Manual Guide TCSEC / Administration Documentation ITSEC

We discovered that for some aspects, e.g. documentation, the level of granularity in the modularization was either too high or too low. This needs further investigation.

4. SUPERSET:

The modularized criteria were merged to build the superset.

In the superset we reduced the aspects further in order to keep the resulting matrix representation easy to understand.

Functionality:

We used a subset of Audit TCSEC / Accountability and Audit ITSEC of step 3.

Assurance:

We used the following subset of step 3.

- Covert Channel Analysis TCSEC / Vulnerability Analysis ITSEC
- Security Features User's Guide TCSEC / User Documentation ITSEC

The superset consists of three parts. The first part associates the functionality aspects of TCSEC and ITSEC (see Figure 2). The second part associates the assurance - correctness aspects (see Figure 3). We chose the most appropriate wording from either TCSEC or ITSEC. In few cases minor modifications were made. In the third part we visualized the result in form of a superset matrix, where the rows represent functionality and the columns represent assurance (see Figure 4).

As an example, the superset matrix in Figure 4 is filled with scores for TCSEC class B2 in the left part and ITSEC class F4 with evaluation level E4 in the right part of the matrix cells.

Appending the score to the chapter number of the assurance aspect results in the subchapter number. E.g. the upper left cell contains the score 1 for '2.1 Covert Channel Identification' and this results in subchapter '2.1.1 Covert Storage Channel Identification'. There the assurance aspect for this score is defined. See Figures 3 and 4.

Detailed results are available in TMRSE.

OUTLOOK

Our results show that a mapping between the TCSEC and ITSEC cannot be done with a simple correspondence table ITSEC, p. 114. A mutual recognition requires therefore detailed and precise work on this subject. The superset example shown above demonstrates the feasibility of the method. It was not our intention to define the entire superset matrix with all criteria aspects in detail. We wanted to show the method's feasibility, that it can be done and how it can be done.

A complete detailed work on this subject may generate valuable feedback for the responsible evaluation authorities. Some problems that were discovered by our working group are mentioned here:

- The separation between functionality and assurance aspects sometimes seems to be inconsistent within ITSEC (e.g. covert channel should be functionality not assurance).
- The ITSEC chapter effectiveness cannot easily be mapped on the notion of Trusted Computing Base (TCB) in TCSEC.
- There is a risk of adopting wrong interpretations. Additional inputs (interpretation documents, evaluator manuals, etc.) may be needed to create the superset matrix.

In the next step a detailed and complete superset matrix must be defined and agreed upon. Furthermore, the results should be adapted to new releases of criteria catalogs.

CONCLUSION

The mapping of criteria catalogs via modularization, comparison, and merging into a superset is feasible. We recommend to complete this work. We propose to establish an international group working full-time on this subject. Support and recognition by the official authorities is required to get the results accepted and agreed upon. Finally, we suggest that tools should be developed to reduce paper work and increase efficiency.

MEMBERS OF THE WORKING GROUP

Andrea Arnold (Digital Equipment, chair)
Hans-Joachim Bierschenk (VDMA)
Ulrich van Essen (GISA, advisor for ITSEC)
Siegfried Gerber (PCS)
Cornelia Persy (Siemens AG)

Wolfgang Schaefer (DATEV)
Siegfried Schall (AEG)
Hans-Dieter Schaupp (ZVEI)
Dr. Gottfried Sedlak (IBM)
Manfred Sielemann (Mannesmann
Kienzle)

REFERENCES

- | | |
|--------------|--|
| TCSEC | Trusted Computer System Evaluation Criteria (TCSEC), Department of Defense DoD 5200.28-STD. December 1985. |
| ITSEC | Information Technology Security Evaluation Criteria (ITSEC) Harmonised Criteria of France - Germany - the Netherlands - the United Kingdom Herausgeber: Der Bundesminister des Innern, Bonn. Mai 1990. |
| TMRSE | Towards Mutual Recognition of Security Evaluations (TMRSE) VDMA/ZVEI Working Group Editor: c/o VDMA, FG BIT, Lyoner Strasse 18, W-6000 Frankfurt 71, Germany. October 1991. |

1. Functionality

1.1 Accountability, object access

The system shall contain an accountability component which ...

1.1.1 Date

ITSEC F2, F6. Annex A, 1. F1 - F5 (TCSEC Classes).

Page 97.

TCSEC C2 - A1. 2.2.2.2 Audit. Page 16.

1.1.2 Time

ITSEC F2, F6. Annex A, 1. F1 - F5 (TCSEC Classes).

Page 97.

2. Assurance - Correctness

2.1 Covert Channel Identification

2.1.1 Covert Storage Channel Identification

The system developer shall conduct a thorough search for covert storage channels. ...

TCSEC B2. 3.2.3.1.3 Covert Channel Analysis. Page 30

ITSEC none

2.1.2 Covert Channel Identification

The system developer shall conduct a thorough search for covert channels.

TCSEC B3. 3.3.3.1.3 Covert Channel Analysis. Page 39

ITSEC E4 - E5. 3.5.1.1.4.b Detailed Design. Page 57.

Figure 2.
Superset: Functionality

Figure 3.
Superset: Assurance - Correctness

Figure 4.
Superset matrix with scores for
TCSEC B2 and ITSEC F4/E4

Funtionality	Assurance				
	2.1 Covert Channel Identification {1 ... 3}		2.2 Covert Channel Bandwidth {1 ... 3}		
	TCSEC B2	ITSEC F4/E4	TCSEC B2	ITSEC F4/E4	
1.1 Accountability, object access					
1.1.1 Date	1	2	1	-	
1.1.2 Time	1	2	1	-	
1.1.3 User Identity	1	2	1	-	
1.1.x Object Name	-	2	-	-	
...					
Accountability					
...					
System as a whole					

Executive Summary

Fielding COTS Multilevel Security Solutions: The Next Step

James P. Litchko, Trusted Information Systems, Inc., Moderator
Lorraine Dunn-Martin, Unisys Defense Systems, Inc.
Mindy E. Rudell, The MITRE Corporation
George R. Mundy, Trusted Information Systems, Inc.

As a result of the 1989 Joint Multilevel Security (MLS) Initiative, MLS requirements for DoD C41 systems were formally identified by JCS . At the same time, the initiative determined that there were no commercial-off-the-shelf (COTS) solutions available to support the MLS requirements identified. In the past two years, many new NSA approved COMSEC and trusted COTS products have become available. System integrators have been actively working with these INFOSEC products and developing MLS system solutions to support the identified requirements. During these efforts, many questions were asked and continue to be asked:

- What approved INFOSEC products support MLS?**
- How do we integrate these products to develop a MLS system?**
- What problems are involved when using COTS to develop MLS system?**
- What other COTS products are necessary to improve the availability of MLS?**

This panel of experienced MLS professionals will offer their personal insights on all of these questions based on their recent experiences involved with integrating INFOSEC products. Based on Mindy Rudell's involvement with MLS testbeds and development of the MLS Target Architecture and Implementation Strategy for the Joint MLS Technology Insertion Program, she will identify the availability and applicability of COTS INFOSEC products to support DoD MLS requirements. Lorraine Dunn-Martin and George Mundy will provide a brief review of several methods used to integrate these products using actual development examples using operating systems rated at the B level of trust. Using these presentations as a foundation, the panel will spend the majority of the time discussing the issues related to developing MLS systems from COTS products and concepts on how to migrate to the effective MLS system development.

Through interactive discussions with the audience and the panel, integrators and program/system managers will be provided the opportunity to gain the panels recommendations and perspectives on issues and concerns as they relate to their own MLS system development.

Issues and topics developed during this session are expanded upon in the **Trusted Applications in the Real World** which occurs 0900-1030 on 4 October 1991 in the Palladian Room.

INFERENCE AND AGGREGATION IN MULTILEVEL DATABASES: RESEARCH DIRECTIONS

Teresa F. Lunt, Panel Chair
Computer Science Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

Panelists:

Thomas D. Garvey, SRI
Bhavani Thuraisingham, MITRE
Cathy Meadows, NRL
Cristi Garvey, TRW
Gary Smith, National Defense University

The inference problem is when some set of data with a low access class can be used to infer data with a high access class. Some researchers have approached the problem from a data design viewpoint, attempting to find techniques for defining data structures and assigning classifications to these structures in such a way that inference problems are minimized. Other researchers have proposed to develop techniques and mechanisms for detecting inference problems during query processing. These mechanisms would evaluate each query in the context of previous information returned to that user and make a decision to accept the query or withhold the results, based on a set of classification rules. Each of these approaches has advantages and disadvantages. Each can be evaluated in terms of its complexity, performance costs, degree of assurance achievable, and degree of assurance attainable.

Following are remarks by each of the panelists.

Detecting and Evaluating Inference Channels

Thomas D. Garvey
Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

Introduction

The inference problem is when some set of data classified at a low level (or *low data*) can be used to infer data classified at a high level (*high data*). That is, there is a direct inference path (possibly including external data) from the low data to the high data.

Inferential security remains one of the most critical and challenging problems to the database community. We have begun work toward developing a formalism for characterizing inferential problems of different types based on formal logical reasoning and theories for approximate reasoning. We believe the essence of inferential security problems are well captured by these formalisms.

Logical Formalisms for the Inference Problem

We characterize inferential security problems as belonging to one of three distinct types, based on the degree to which high data may be inferred from low data. The most restrictive type of channel occurs when a formal deductive proof of the high data can be derived from the low data—when this is the case, we say that a *logical inference channel* (or a *logical channel*) exists. A slightly weakened requirement for a channel is when a deductive proof may not be possible, but a proof could be completed by assumption of certain axioms. In this case, an abductive proof is possible, and we will term the channel an *abductive inference channel* (or an *abductive channel*). The third situation is when it is possible to determine likelihoods that assumed axioms might be knowable by a user with legitimate access to low data that would enable the inference of high data with some measure of belief greater than an acceptable limit. In this case, we will (loosely) call the channel a *probabilistic inference channel* or just a *probabilistic channel*.

Logical channels can be described by standard propositional logic (PL) or first-order predicate logic (FOL). If PL is applicable, determining whether a logical channel exists is a decidable proposition, but may be quite expensive. In the more general case of FOL, the question is not decidable. This means that there is no way of knowing whether a logical channel exists until one is found. Since, in general, logical channels must not involve assumptions of facts, they must be based entirely on data found within the database.

Abductive reasoning is a distinctly different form of reasoning than deduction, that is not limited to demonstrating that a formula is a consequence of a theory. In abductive reasoning, the objective is to find assumptions A such that $T \cup A \vdash Q$ even though Q may not be provable from T alone. Abduction has traditionally been applied to diagnostic tasks that reason from events to causes. If Q is observed and $P \supset Q$ is known, then P can be offered as a possible explanation of Q .

Abductive channels represent a much more serious issue, since most inferential channels exist due to knowledge that a normal user might be expected to contribute to the problem but that is not an explicit part of the data or knowledge base. An abductive proof, however, can include

assumptions and can consider the degree to which a user is likely to know some fact necessary to the completion of a proof. Since abduction involves assumptions about the user's belief structure, it involves modal logics, particularly epistemic logics.

In using an abductive theorem prover (ATP) for inference channel detection, high facts would become theorems to be proved. The ATP would back-chain through inference rules to low data (which would become the proof axioms) or to assumptions. No assumption would be permitted that was already present as a high fact. Acceptable assumptions proposed for a proof would need to be evaluated by the database security manager to determine the degree to which they may be known to low users.

A variety of schemes have been devised to determine the cost of an abductive proof. These typically include a cost for each additional proof step and a cost associated with an assumption. SRI has developed an abductive theorem prover (ATP) as an extension to Prolog that allows one to set these weights as appropriate for the problem of interest. Setting assumption costs high relative to proof steps leads the ATP to prefer deeper proofs with fewer assumptions. Setting the assumption costs to infinity leads to standard theorem proving. Setting them low causes the ATP to prefer assumptions.

From an informal point of view, an abductive theorem prover used for detecting inference channels should have a cost for proof steps chosen to cause it to search moderately deeply for logical channels (i.e., channels that do not require assumptions), but not too deeply, as the deeper the proof required, the more work a user will have to put into the deduction, and therefore, the less likely (or the lower the bandwidth of) the channel.

One means of setting these costs is to consider the likelihood that a particular user might know the assumed facts. Assumption costs could be related to these likelihoods, and the overall cost of the proof would then be a function of these probabilities. A variety of computational schemes, based on classical probabilities, belief functions, or fuzzy logic could be considered for the task of determining the cost of an abductive proof incorporating beliefs. Using a formal theory for approximate reasoning would allow the computed cost to reflect the likelihood that high data could be inferred by a low user with ordinary or particular knowledge.

Our investigations of this formalism will, we hope, lead to the development of database design tools so that a proposed database design can be analyzed for inference channels and restructured so that the problems are eliminated or minimized.

Approximate Reasoning for Evaluating Inference Channels

Inferential security problems arise when it is possible for a user to use low data to infer the truth of high data with some degree of probability. For example, flight destination airports may be sensitive data, while aircraft range, payloads, and departure fields may be stored at a low security level. By combining information about range, payloads, and departure fields, a user may be able to greatly narrow the set of possible destination airports, and in so doing increase the *likelihood* that an aircraft's destination is among the reduced set. Further information (say, data about the aircraft-handling capabilities of the airfields in the reduced set), may serve to reduce the space of possibilities even more.

Such probabilistic channels are related to abductive channels because the assumptions and logical rules used in an abductive proof may have degrees of belief associated with them which represent the likelihood that they may be known to a user. These degrees of belief can then be propagated through the abductive proof tree to determine the degree to which the user is likely to be able to infer the high data in question. In effect, the ATP can be used to uncover the existence

of a channel and approximate reasoning methods used to evaluate the relative seriousness of the channel.

We are investigating the use of evidential reasoning in evaluating the seriousness of an inference channel. Evidential reasoning departs from classical probability theory in that it permits beliefs to be attached to disjunctions of statements, rather than requiring they be assigned to singletons in the universe of discourse (the set of mutually exclusive and exhaustive statements that form the "vocabulary" for the problem statement).

For example, we may know that a particular aircraft, due to its range and location, may be able to fly to a set of airports. When considering which airport it is really going to fly to, we can identify it only as a member of this set. Therefore, we may assign our belief about the plane's destination to the *set* of possibilities. When beliefs of components are later needed, they are underconstrained as a result of the disjunction, and an interval representation is needed to capture the true constraints. This interval enables the explicit modeling of both what is known (although with uncertainty) and what is unknown.

For inference control, an abductive proof structure combined with information about the likelihood that a user might know facts assumed in the proof can be used to calculate the likelihood that the user could infer high data. Furthermore, sensitivity analyses can be carried out over the information structure in order to determine which information has had the greatest impact on the inference. This information might then be an initial candidate for upgrading in order to eliminate the channel.

Evidential reasoning techniques have been automated in SRI's Gister system.

Summary

The application of abductive reasoning offers a computational mechanism for detecting inference channels in databases. We feel that as a logical formalism, abduction is the most appropriate model for most inference channels involving strictly logical inferences. We identified probabilistic channels as another important class of inference channels, those associated with the likelihood of inferring high data from low data that a user might be likely to know with some probability. We offer evidential reasoning as a candidate technology that could be linked with abduction to provide an effective computational framework for reasoning about such probabilities.

Inference Prevention in Databases:

Data Design vs. Query Processing

Catherine Meadows
Code 5543
Naval Research Laboratory
Washington, DC 20375

Recently, researchers have proposed two methods for the prevention of inferences in database. One of these is to detect potential inference problems beforehand and to then design the database so that unwanted inferences can be prevented. This may require the use of specialized semantic modeling techniques. The other is to keep a record of past accesses, and whenever a new access is requested, to compare the query against the past access history to determine whether or not any unwanted inferences can be drawn. We will refer to these two approaches as the data design approach and the query processing approach.

Clearly, the data design approach has its attractions. Instead of having to check for inferences during each query, one checks only once, at the time the database is being built. However, before rejecting the query processing approach out of hand, we should ask the following questions:

1. How easy is it to protect against all future inferences? Will we be able to predict the future history of the database? What if we discover new inferences? Will we have to redesign the database?
2. How does the complexity of examining an entire database for inferences compare against the complexity of examining an access history or set of access histories?
3. How well do our semantic modeling techniques capture the kinds of inferences possible? Can we develop a measure of the effectiveness of these techniques? Are there inferences that can't be prevented by semantic modeling techniques?
4. What do we do when the sensitivity of data decreases? How hard is it to build inference prevention mechanisms that take this into account into data design versus building them into the query processor?

Finally, we should investigate the possibility of augmenting the data design approach with the query processing approach. It may be that certain kinds of information are best protected by one approach, and certain kinds by another. For example, information whose sensitivity is relatively static might be best protected by the data design approach. On the other hand, information whose sensitivity might change, either because it may later be augmented by new information later on from which sensitive inferences might be drawn, or because its sensitivity decreases over time, might be better protected by the query processing approach.

Challenges in Addressing Inference and Aggregation

Gary Smith
Information Resources Management College
National Defense University
Washington, DC

This paper identifies some of the issues that must be considered (and questions to be asked) when evaluating different approaches for addressing inference and aggregation in multilevel secure database systems.

In one sense, inference and aggregation are the same problem — they both refer to the ability to obtain data/information that is classified high from data/information classified low. In fact, in most instances of aggregation, high data is normally *inferred* (rather than explicitly revealed) when the low data is combined. Thus inference and aggregation have several challenges in common. First, the primary consideration for understanding, and therefore solving, these problems is the requirement to explicitly identify the data/information that must be protected. Unfortunately, this requirement is not always easy. Moreover, the answers are dependent on the data/information/knowledge that forms a part of the application domain (i.e., the piece of the real world that the automated system supports). Approaches to providing automated support for inference and aggregation must be able to handle all the generic types of problems. Unfortunately, a comprehensive taxonomy of generic inference and aggregation problems is yet to be formulated. (What types of generic inference and aggregation problems can an approach handle?) The second challenge relates to the invalidity of a *closed world assumption* (i.e., an assumption that the database contains all data/information/knowledge needed to infer high data). The closed world assumption is not practical because humans possess great cognitive powers for deducing new facts (i.e., inference). Often, facts that are external to the database are combined with data from the database to allow a user to infer new data/information. (How, and to what extent, does an approach to solving the inference and aggregation problem incorporate data/information/knowledge that is not in the database?) The third challenge relates to the identification of possible inference paths. Relying solely on the designers and domain experts to exhaustively identify inference paths may not result in all possible paths being identified. Providing automated *reasoning capabilities* for identifying possible inferences over complex application domains is essential. (How robust are the reasoning capabilities being provided?)

On the other hand, aggregation presents additional challenges. Tom Hinke made an important characterization of two types of aggregation: *inference aggregation* (combination of two *different* types of data objects is classified higher than the classification of either object) and *cardinal aggregation* (when multiple instances of *the same* data object are classified higher than each instance). The distinction between these two types of aggregation is important for two reasons. First, inference aggregation can be effectively handled through good database design; it is the *real* aggregation problem that is most difficult and requires research for further understanding. The second reason involves the *soundness* of an aggregation security policy. At the 3rd RADC Database Security Workshop, Roger Schell asserted that (cardinal) aggregation security policies are inherently unsound; therefore, we should not expect to find acceptable mechanisms to implement those policies. Often artificial constraints are suggested (e.g., a user can retrieve only ten records). (What facilities are provided to deal with cardinal aggregation?)

Approaches to Handling the Inference Problem

Bhavani Thuraisingham
The MITRE Corporation
Burlington Road
Bedford, MA 01730

Introduction

It is possible for users of a database management system to draw inferences from the information that they obtain from the database. The inference process can be harmful if the inferred knowledge is something that the user is not authorized to acquire. That is, a user acquiring information which he is not authorized to know has come to be known as the inference problem in database security. We are particularly interested in the inference problem which occurs in a multilevel operating environment. In such an environment, the users are cleared at different security levels and they access a multilevel database where the data is classified at different sensitivity levels. A multilevel secure database management system (MLS/DBMS) manages a multilevel database where its users cannot access data to which they are not authorized. However, providing a solution to the inference problem, where users issue multiple requests and consequently infer unauthorized knowledge, is beyond the capability of currently available MLS/DBMSs.

We believe that a triple approach to research is needed to combat the inference problem; one is to build inference controllers which act during transaction processing, the other is to build inference controllers for database design, and the third is to build inference controllers to act as advisors to the Systems Security Officer (SSO). This is because the inference problem is a complex one and therefore an integrated approach is necessary to handle it.

Summary of Effort

Our preliminary investigation of the inference problem included the following. (i) Identifying various inference strategies that users could utilize to draw unauthorized inferences. These strategies included inference by deduction, inference by induction, inference by heuristic reasoning, inference by semantic association, inference by analogical reasoning, and statistical inference. (ii) Designing techniques for handling certain inference strategies during query processing. (iii) Analyzing the complexity of the inference problem.

Later, we focussed on developing techniques for handling inferences during query processing, update processing, and database design. We utilized security constraints to assign security levels to data and information. The inference controller, which functions during query, update, and database design operations, processes these security constraints in such a way that security violations with respect to certain types of inferences do not occur. We also carried out an investigation on the use of conceptual structures to represent and reason about multilevel applications as well as the issues involved in designing a knowledge-based inference controller. We discuss some of our approaches briefly in this paper.

Security Constraint Processing

Security constraints play an important role in our approach to handling the inference problem. They are rules that assign security levels to the data. In our approach security constraints are specified as horn clauses. Therefore techniques developed for verifying and validating logic programs could be utilized for checking the consistency of the constraints. We have defined various types of security constraints. They include (i) simple constraints that classify a database, relation or an attribute, (ii) content-based constraints that classify any part of the database depending on the value of some data, (iii) event-based constraints that classify any part of the database depending on the occurrence of some real-world event, (iv) association-based constraints that classify associations between attributes and relations, (v) release-based constraints that classify any part of the database depending on the information that has been previously released, (vi) aggregate constraints that classify collections of data, (vii) logical constraints that specify implications, (viii) level-based constraints that classify any part of the database depending on the security level of some data, and (ix) fuzzy constraints that assign fuzzy values to their classifications.

Our approach is to process certain security constraints during query processing, certain constraints during database updates and certain constraints during database design. The first step was to decide whether a particular constraint should be processed during the query, update or database design operation. After some consideration, we felt that it was important for the query processor to have the ability to handle all of the security constraints. This is because most users usually build their reservoir of knowledge from responses that they receive by querying the database. It is from this reservoir of knowledge that they infer unauthorized information. Moreover, no matter how securely the database has been designed, or the data in the database is accurately labeled, users could eventually violate security by inference because they are continuously updating their reservoir of knowledge as the world evolves. It is not feasible to have to re-design the database or re-classify the data continuously.

The next step was to decide which of the security constraints should be handled during database updates. After some consideration, we felt that except for some types of constraints such as the release and aggregate constraints, the others could be processed during the update operation. However, techniques for handling constraints during database updates could be quite complex as the security levels of the data already in the database could be affected by the data being updated. Therefore, initially our algorithms handle only the simple and content-based constraints during database updates. The constraints that seemed appropriate to be handled during the database design operation were those that classified an attribute or collections of attributes taken together. These include the simple and association-based constraints. For example, association-based constraints classify the relationships between attributes. Such relationships are specified by the schema and therefore such constraints could be handled when the schema is specified. Since a logical constraint is a rule which specifies the implication of an attribute from a set of attributes, it can also be handled during database design.

We have developed a query processor prototype and an update processor prototype. We have also developed techniques for handling certain constraints during database design. The update processor and the database design tool could be used off-line while the query processor must augment the MLS/DBMS and is used on-line. Our ultimate goal is to combine the solutions that we have developed to process security constraints during query, update, and database design operations, and subsequently develop an integrated tool for processing security constraints. The update processor and the database design tool should ensure that the database as well as the schema are consistent with the constraints. However, if the real-world is dynamic, and the database and/or the schema are at any time inconsistent, then there must be a mechanism to trigger the query processor to

process all of the relevant constraints.

Conceptual Structures

The integrated tool discussed above assumes that an initial set of security constraints and schema are available. However, generating these schemas and constraints from the specification of the multilevel application is by no means a straightforward task. A tool to aid the application specialist and/or the SSO for constraint and schema generation from the application specification would be desirable. One can envisage this tool to be a front-end to the integrated tool discussed above. Our approach to developing such a tool is to first develop a conceptual data/knowledge model to represent the multilevel application and then develop techniques for reasoning about the application in order to detect potential security violations and inconsistencies. We have investigated the use of conceptual structures to represent and reason about the multilevel application. The particular conceptual structures that we have investigated are semantic networks and conceptual graphs. We have developed multilevel semantic nets and multilevel conceptual graphs and showed how multilevel applications could be represented by these structures. We also showed how an SSO could reason and consequently detect security violations via inference.

Knowledge-based Inference Control

The prototypes that we have developed handle only logical inferences that users could utilize to deduce unauthorized information. As discussed earlier, in reality users could utilize several inference strategies. Therefore for an inference controller to be effective, it should be able to use various types of reasoning techniques in order to handle the users' inference strategies. We have carried out a preliminary high level design of a knowledge-based inference controller called XINCON (eXper INference CONtroller). XINCON uses frames and rules to represent knowledge. The major components include an inference engine which handles logical as well as fuzzy inferences, a truth maintenance system which ensures that the beliefs are consistent, a knowledge manager, and a conflict resolution module which determines the actions to be taken in a conflicting situation. XINCON could augment an MLS/DBMS and/or it could act as an advisor to the SSO.

Acknowledgements

We gratefully acknowledge the Department of the Navy (SPAWAR) for sponsoring our work on the Inference Problem under contract F19628-89-C-0001. We thank Marie Collins and William Ford for their contributions to the work described in this paper.

Executive Summary

MILITARY AND TELECOM SECURITY: SPECIALIZED METHODS

Richard Lefkon, New York University, Moderator

PANELISTS

**Debra Banning, Sparta
Myron Cramer, Booz Allen & Hamilton
Ed Fulford, Northern Telecom**

Each speaker makes a formal presentation with questions and answers, and a general symposium concludes the session.

The four presentations explore potential defense security threats posed by unfriendly computer programs such as viruses and Trojan Horses.

Ed Fulford discusses some of the current limitations to security public networks and proposes awareness programs and other solutions.

Myron Cramer discusses computer viruses, their insinuation and execution.

Debra Banning and Gail Ellingwood discuss the need to protect embedded computer system critical functions. They propose pervasive anti-virus measures.

Dick Lefkon discusses the implications of a Millennium Trojan Horse. He proposes that software be examined and tested for calendar dependencies.

Speaker presentations are followed immediately by a moderated discussion between the panel and attendees.

Executive Summary

MALICIOUS CODE PREVENTION FOR EMBEDDED COMPUTER WEAPONS SYSTEMS

Debra L. Banning
Gail M. Ellingwood
SPARTA, Inc.
3440 Carson Street
Torrance, CA 90503

ABSTRACT

With the recent virus infection for personal computers being shipped to the Persian Gulf during Operation Desert Storm, the vulnerability of our military defenses to malicious code attacks has been highlighted. Modern weapon systems make extensive use of embedded computer systems for such critical functions as weapon aiming, weapon sensor processing and guidance, safe and arming, and real-time control. Concern has been raised over the potential for sabotage of weapons by the insertion of malicious code, either directly into the weapon's application code or indirectly via the application software development environment. This paper summarizes the results of a recent study¹ that examined Embedded Weapon System (EWS) vulnerability to malicious code.

INTRODUCTION

The study of EWS vulnerability was performed in three phases: The development of a taxonomy of malicious code; a weapon system vulnerability analysis; and identification of a suitable defense methodology for protecting against malicious code attacks. This paper will briefly focus on the results of the vulnerability analysis and the definition of a Malicious Code Resistant Security Architecture (MCRSA) for defending against malicious code attacks.

MALICIOUS CODE THREATS TO WEAPON SYSTEMS

To understand how malicious code could affect EWSs it is important to understand the functions of a typical weapon system. An EWS is a computer or group of computers that is a component of a larger system used to perform a specific military mission. The EWS is most likely to be a part of a distributed computer system architecture, where other remotely located computers interact in some fashion with the computers residing on-board the weapon. One embedded computer may also cooperatively act with several other embedded computers as in a military aircraft. Figure 1 depicts general weapon system functions.

Malicious code may affect weapon system functions in both obvious and more subtle attacks. Obvious attacks may result in destruction of the weapon or failure at a critical time. When the malicious code triggers in this manner, it would be easy to determine that the weapon system software has been infected. However, if a more subtle attack is used (e.g., performing a modification in aiming functions to slightly miss the target) the malfunction may be initially attributed to some other cause. In many cases a detailed understanding of the functions of a weapon system is necessary for the writing of a malicious program that would affect its functions. However, some

1. The study was performed by SPARTA, Inc., with support from UC Davis, for Picatinny Arsenal.

defined a Malicious Code Resistant Security Architecture (MCRSA). The MCRSA consists of three primary components:

- Malicious code prevention mechanisms incorporated within the software development system.
- A malicious code detection system, called the Malicious Code TestBed (MCTB), used to test the weapon system software and the utilities used within the development system to create the software.
- Weapon system defenses in the ECWS itself.

The MCRSA is supplemented by a set of administrative controls incorporated within each of the above three components. This includes strict configuration management and methods to provide a reasonable assurance that malicious code is not carelessly and needlessly introduced into the ECWS life cycle. The MCRSA is shown in Figure 3.

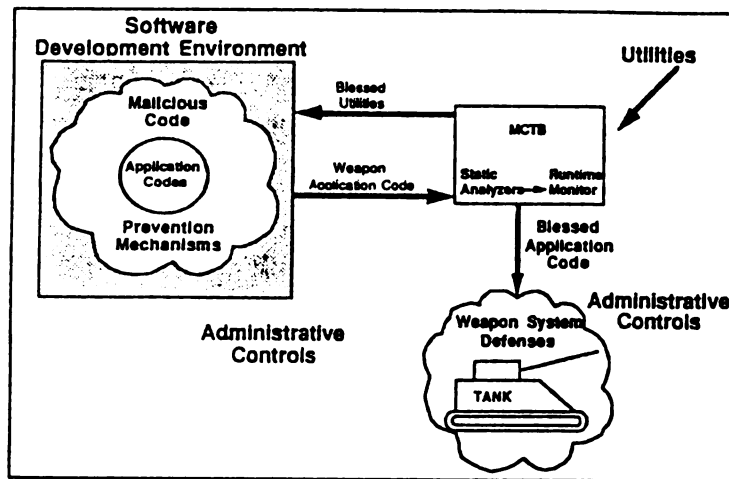


Figure 3. Malicious Code Resistant Security Architecture (MCRSA)

Most software development environments provide administrative controls and technical mechanisms that assist in preventing malicious code infection. However, these have proven to be unsuccessful in completely preventing infection. Therefore, the definition of a MCTB which would be used to test software prior to incorporation into a weapon system is a very important aspect of the MCRSA.

Due to in-field programmability, maintenance updates and the use of communication links by weapon systems, it is not sufficient to provide defenses only while the software is being developed. Previous to this study weapon systems did not provide a means for detecting malicious activity once the weapon system was deployed. This led to the definition of a Weapon System Security Monitor (WSSM) that can be added to a weapon system bus as an additional co-processor to detect unusual activity that could indicate malicious code infection during the system's operation.

functions (e.g., ballistic computations) use common library routines (e.g., square, square root) that may be affected by malicious code that has been developed with very little knowledge of the specific weapon system.

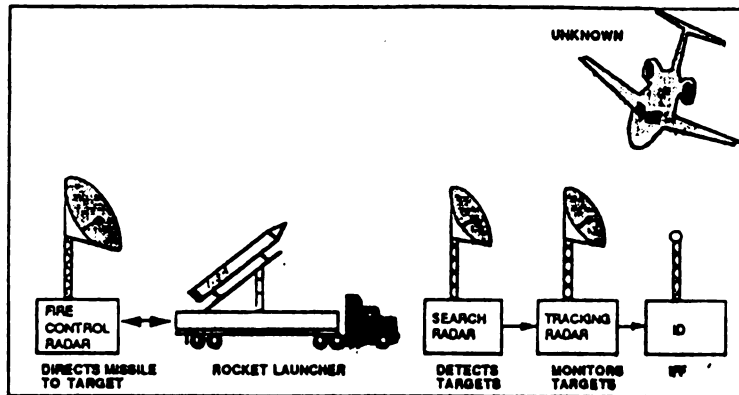


Figure 1. General Weapon System Functions

Prior to this study it was assumed that the primary means of malicious code infection was during the weapon system software development state. Furthermore, Trojan Horse programs or Trap Doors were considered more of a threat to an operational system than viruses. Viruses were not considered a primary threat since, once the software was burned-in and included in the weapon system, they would be unable to propagate. This may not be the case. Several weapon systems have the capability for in-field programmability and maintenance updates which would allow viruses to further propagate. In addition, current research indicates it may be possible to infect weapon systems with viruses via radio links which many complex weapon systems use for communication between components. Figure 2 shows that malicious code can affect software throughout its life cycle.

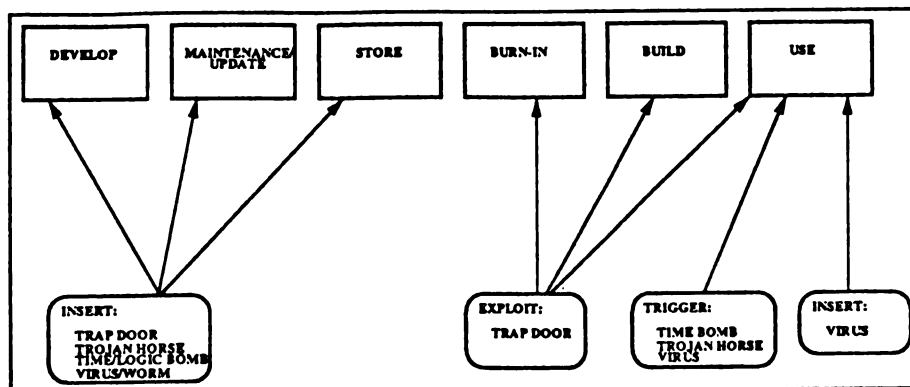


Figure 2. Software life Cycle Vulnerability

MALICIOUS CODE RESISTANT SECURITY ARCHITECTURE (MCRSA)

To minimize the threat caused by malicious code, security controls must be provided for all phases of the weapon system software life cycle. In order to do this, we have

MALICIOUS CODE TEST BED (MCTB)

The Malicious Code Test Bed (MCTB) is a stand-alone system in which the development software can be loaded for malicious code detection. Weapon system software is loaded onto the MCTB and tested using a variety of tools directly prior to downloading for the ECWS build process. The MCTB can also be used to provide assurance that utilities (e.g., compilers, debuggers) used to develop the software do not contain malicious code. The tools used on the MCTB should be capable of detecting a variety of malicious code, particularly Trojan Horse and virus programs. The tools recommended for incorporation into the MCTB consist of research tools that are in line with the state-of-the-art of malicious code detection and can be adapted to the general development environment.

Given the nature of malicious code writers and their proclivity to adapt a virus rapidly once a defense is provided, it is important for an effective malicious code detection system to have the potential to detect malicious code of the future. Therefore, the MCTB should include a learning capability such that the system's knowledge base would be modified as new types of malicious code are detected.

WEAPON SYSTEM SECURITY MONITOR (WSSM)

Weapon systems that provide maintenance update capabilities while deployed or are in-field programmable, are susceptible to infection from malicious code that may not have been previously detected. More importantly, recent investigation has shown that it is feasible that adversaries may attempt to infect weapon systems via the communications links. Given these possibilities, it is important to provide defenses against malicious code in the weapon system itself.

Generally, a weapon system is a distributed system consisting of hosts, shared storage, a shared I/O controller, a simple distributed operating system on each host and static allocation tasks to processors. It is not feasible to propose handcrafting operating systems for existing weapon systems. Therefore, security must be retrofitted to the existing equipment. Commercially available security mechanisms are not successful in detecting many types of malicious code on other than PC operating systems. However, malicious code can be detected by adding a monitoring capability within the weapon system that monitors the actions of the processes and detects suspicious activity.

POSITION IN BRIEF

Malicious code attacks are continuing to evolve. New avenues for infection, methods for disguising code and methods for evading detection are being discovered. To protect our weapon systems from these attacks it is important to provide mechanisms that are not geared towards one type of malicious code but instead have the ability to adapt to the malicious code evolution.

These mechanisms must be incorporated at all stages of the software life cycle to provide the necessary protection for the weapon system. It is our intention that the MCRSA defined under this study provides a method for accomplishing this goal. The use of the MCTB in the development environment and the WSSM within the weapon system itself should provide an effective defense against present and future malicious code attacks.

COMPUTER VIRUSES AS ELECTRONIC WARFARE

Myron L. Cramer
Booz, Allen & Hamilton Inc.
4330 East West Highway
Bethesda, MD 20804
(301) 951-2228

ABSTRACT

This position paper introduces the concept for a new type of electronic warfare based upon the capabilities of computer viruses. These capabilities include the ability of viruses to infect a military computer's software and to propagate through enemy tactical data networks.

DISCUSSION

The purposes of electronic warfare are to deny an adversary the effective use of his electronic systems. This is accomplished through the use of electronic jamming of radio links. Deception jamming techniques can often be more effective than simple noise jamming, since they deny an adversary the opportunity to respond to the action. As electronic systems have become increasingly computerized, the functions of these systems are becoming increasingly implemented in software. Thus, attacks against this software can provide the ultimate form of deception jamming by manipulating an adversary's data systems.

The basic argument runs as follows:

- Computer viruses can be electronically injected into digital radio links.
- There are mechanisms for viruses thus injected, to be caused to execute.
- The existence of potential threats of this type significantly undermines the protection provided through normal Software Quality Assurance and through physical security measures.
- Consequently, a new approach is needed to assess vulnerabilities and to design protective measures.
- Viewing this problem from the perspective of Electronic Warfare provides a structure to evaluate these issues.

POSITION IN BRIEF

Current trends in the development of military electronic systems have created the opportunity for a new form of electronic warfare using computer viruses spread through radio transmission. The potential for this type of electronic attack significantly changes the nature of the computer virus problem beyond the elements controllable by software assurance and physical security.

Executive Summary

PREVENTING VIRUS INSERTION THROUGH SWITCHES

Ed Fulford

Manager, Information Security, Northern Telecom

ABSTRACT

Once, telephone switch vendors and users felt switch architecture was the primary deterrent to placing viruses inside the public network. Now, the availability of digital technology has increased the potential for virus attacks on switches, and has highlighted the need for improving user and resource management to negate these attacks.

CURRENT SYMPTOMS -- INDUSTRY-WIDE ISSUES

The implementation of aggressive virus detective and preventive measures within public networks is still hampered by the following:

- User awareness training on switch security software and practices has not been proactive. In the past, the common approach was to cover up possible security concerns, rather than address them with the user in order to enhance the overall network control and maintenance procedures.
- Telecommunications vendors have not fully standardized security controls based on governmental and industry requirements. These standards are only now being widely publicized, and vendors are dedicating more resources in their design and development areas to ensure compliance with these standards by implementing them in product security software and procedures.
- User access control is still based primarily on the reusable password. This control technique can be easily compromised and does little to provide actual user authentication.
- Use of encryption for protection for sensitive files and programs has not been readily adopted. Once access controls are breached, (through "social engineering" or some other method) it is often relatively easy to find out system management passwords and/or capabilities, due to the lack of additional safeguards.
- Software management tools, similar to those for identifying viruses in the personal computer environment, are largely non-existent. In the past, vendors may have assumed that the complexity of the switch's architecture and programming was a sufficient obstacle to the propagation of viruses; this is no longer a valid assumption.

SEEKING THE CURE -- ONE APPROACH

From a vendor's perspective, the threat of a virus within a product is terrifying and raises numerous questions. Why didn't we detect the virus when it infiltrated the switch? Can we find it? Can we identify who put it there? Can we remove it and fix any problems? Can we assure the user that this will not re-occur? These initial questions will surely lead to more complex and expensive questions. If the vendor can only react to this type of problem, the cost of a solution will quickly outstrip available resources, and will most likely alienate the users.

However, the scenario described above need not always be the norm. The appropriate response is pro-active; the vendor and user working in concert to identify and resolve these issues. The approach advocated to address this problem has several integral components:

- 1). **User Awareness.** Vendors must continue to stress the proper installation and management of the security tools provided with the switch. They can do this in a number of ways: by training user technical personnel on switch security, by pre-configuring the security software, by consulting with the user on security after the switch has been installed, and by sponsoring security awareness symposiums with user groups. While these are not all the techniques that could be used, a combination of them would help the vendor and user develop aggressive resource management practices, and provide warnings about the threat of viruses.
- 2). **Product Security Standards.** Since many of the switches in use today are digital computers and now extremely susceptible to virus attacks, computer security standards should be applied where appropriate. In reviewing computer security guidelines that have been published (by BELLCORE, the U.S. Government, and the telephone companies), many of the security requirements are consistent, and all address virus detection and prevention. A matrix of switch security standards can be developed by vendors, for us in standardizing security software and procedures across all telephone equipment products where applicable. Users would then be able to deploy and administer security on all products more efficiently, because of the common design functionality.
- 3). **User Authentication.** The technology to identify and verify users is available today, and will help limit the possibility of virus attacks on switches. Voice recognition is being tested by vendors to authenticate users by speech patterns and dialects (largely overcoming prior security concerns that a tape recording of a user's voice could be used to "fool" the security system). Encryption of passwords, using public key cryptography, is being developed by vendors to make reusable passwords more

secure. Time based access control algorithms and "one time" passwords can be used to provide gateways to the public switched network, which will also provide additional constraints to unauthorized access and virus attack. Vendors could provide any or all of these controls, within the constructs of the standards mentioned above.

4). Virus Detection Tools. Telephone switch architecture and software, while being based on the digital computer, is rather specialized. The programming languages used in switches are designed only for developing telecommunications applications, and relatively few people in the user population has access to them. As such, there were few virus attacks on switch operating systems. Now, many vendors are investigating the use of more generalized operating and programming systems (such as the UNIX operating environment and the C programming language) for the next generation of switches. The availability of these more widely used tools will make switches more susceptible to viruses. Vendors are now investigating image inventories, patch control systems and check sum audits on load modules. This will enable review of currently active software to determine if any unauthorized access or changes have taken place. This will also provide the basis for more sophisticated software management and tracking software for future deployment.

5). Partnerships. The most critical part of this process, however, is how cooperative efforts are formed. Vendors need to make sure that key parties - Research and Development, Marketing, Technical Support, and Manufacturing - embrace the need for security and are willing to devote the time and resources required to implement a corporate security direction. Users must do basically the same thing, but with government and industry groups. Finally, vendors and users must openly address common problems and have a defined strategy to solve them. Formal unauthorized telecommunication access programs and product security task forces will go far to ensure that both vendor and user needs and concerns are addressed.

As this approach is phased in, virus attacks on the public switch network will most likely decrease. This should not be seen as any more than a small triumph in a much larger battle. Technology and software will become more sophisticated and less expensive, and security controls will be more at risk. It is up to the vendors and users, together, to push the boundaries of switch security and provide an environment that significantly enhances detection and prevention of virus attacks in the face of these advances.

Nuclear Disaster and the Millennium Trojan Horse

Richard G. Lefkon
Assistant Professor, New York University
609 West 114th Street, New York, NY 10025
dklefkon@well.sf.ca.us
(212) 663-2315

ABSTRACT

As the Millennium is approached, military installations on all sides are urged to test the date dependencies of internal software in order to identify and address a possible date-related Trojan Horse.

EARLY MILITARY COMPUTING

In the beginning of the computer age, business applications and home amusements were the farthest thing from the major users' minds. Eniac and its siblings were used primarily for making trigonometric computations. The precise sines, cosines and tangents resulting from their calculating loops, went into plotting projectile trajectories.

The projectiles generally were artillery shells, with explosives, in warfare. Some subsequent early use of computers took place for what today are referred to as nuclear missile silos. Movies such as "Dr. Strangelove" may not have been far from the truth in depicting rocketry launches triggered in part by computer decision-making.

Historically, most programs did their logical reasoning by arithmetic comparison: Is A greater than B; if so, do such-and-such. Reverse the sign of the numbers, and of course the outcome would change as well.

It is hypothesized that some nuclear missile silos of early construction are present in much their original form today, including the original computer decision-making programs. Further, that at least some of these programs use the current date in part of their reasoning.

DATES AND THE MILLENNIUM TROJAN HORSE

Many of today's LANs and PCs ask the user to input the date in the form YYMMDD. This conference begins on 911001 and ends on 911004. It lasts $(B-A) + 1$, or $3 + 1$, which equals four days. The Thirteenth NCS Conference took place in 1990, one year ago: $1991 - 1990 = 1$ year.

A surprising computational result occurs between the 23rd and 22nd NCS Conference: $2000 - 1999$ equals 1 year. But using the standard YYMMDD format, $001001 - 991001 = - [negative] 990000$. The date difference is negative, and wherever it occurs all the decisions may be backwards - including the decision to arm and launch.

This idea is not so farfetched as it may seem. Recently a financial company's business users discovered to their chagrin that, say, bonds held in 1991 but maturing in 2011 had a profit/loss calculation exactly four times as large - and backwards - the twenty-year span results expected. That even happened using programs written in the 1980's, not the 1950's.

LIMITATIONS OF SOFTWARE QUALITY ASSURANCE

It is a commonplace in commercial programming, that the older a system is, the more likely its source code has been lost or otherwise does not match the stored executable binary. Thus while source code scans and analyses may be helpful they do not constitute a complete solution.

Ballistics launch software, in either well-known or obscure weapons systems and locations, needs to be exercised judiciously to determine its usage of the calendar date.

POSITION IN BRIEF

An appeal is made to defense ministries around the world to seek out the full spectrum of computers in their nuclear weapons installations. As each computer is identified, a controlled test of software can be made, such as bringing the date forward in steps, to observe what happens as the Millennium line is crossed.

Executive Summary

REDUCED DEFENSE SPENDING INCREASES THE NEED FOR TRUSTED SYSTEMS

Carole S. Jordan
Defense Investigative Service
Industrial Security Directorate
1900 Half Street SW
Washington D.C. 20324-1700

Department of Defense budget cuts are increasing the need for defense contractors to use trusted computer systems in their facilities. The Defense Industrial Security Program includes nearly 12,000 contractors that are qualified to work on contracts that use government classified information. Several thousand of these contractors process classified information on automated information systems (AISs) that have been accredited for such processing.

Large defense contractors typically perform on several dozen contracts at any one time. The greater the number of accredited AISs that are used for processing, the more opportunity there is to separate the processing so that data belonging to several different, unrelated contracts do not have to reside on the same AIS. Contracts involving Special Access Program (SAP) data, often have a specific requirement to isolate SAP processing from other processing. For these reasons, most accredited AISs in contractor facilities have operated in the dedicated security mode. (In this mode, all users have a personnel security clearance and a need-to-know for all of the classified information in the AIS). In the dedicated mode of processing, there is very little risk of unauthorized disclosure of classified information, therefore, there is no requirement to meet a level of trust per DoD 5200.28-STD, "DoD Trusted Computer System Evaluation Criteria".

However, broad defense cuts as well as specific budget reductions in DoD procurement are having an impact on companies that contract with the Department of Defense. Fewer contracts are being let, and several large contracts for weapon systems have been cancelled. Defense contractors have reacted to these changes through personnel cutbacks, reorganizations, and in some instances office closings. In some segments of the industry the adjustments have been extreme, underscoring the need for cost-effective solutions.

Along with contractor work force reductions, there have been significant consolidations in their computer operations. Consolidating both operating locations and AIS systems may save money initially, however, moving classified processing onto a smaller number of remaining AISs can have an adverse impact on AIS security.

The trend to reduce the numbers of AISs and combine the processing of unrelated contracts on a remaining AIS can greatly increase the risk of unauthorized disclosure of classified information. The increased risk comes from the result of having some users who are not authorized to access all of the data, once it has been combined on one AIS. E.G., consolidating two dedicated-mode AISs can result in the need for a system high, partitioned or multilevel mode AIS. Each of these modes requires a particular level of trust to be met.

The use of new or existing technology to reduce more effectively costs is increasingly important to contractors who must control operational expenditures. Contractors need precise security solutions in the form of trusted products and subsystems in circumstances of serious vulnerability. Computer hardware and software vendors need to meet the increased demand by continuing to produce a wide variety of cost-effective trusted products and subsystems.

1991: A YEAR OF PROGRESS IN TRUSTED DATABASE SYSTEMS

John R. Campbell

National Security Agency

9800 Savage Road

Fort George G. Meade, Maryland 20755-6000

(301) 859-4387

1991 has seen some significant gains in database security. This panel will discuss some of these gains. Because of the number of these gains, and the limited time for this panel, the presentations will be short. However, the panelists will enjoy discussing these topics further with you after the panel is completed.

The first significant gain is the availability of commercial products. By the time of this panel, the user should be able to choose systems designed to the TCSEC C2 and B1 levels from a variety of vendors. These vendors include ARC, Informix, Oracle, Sybase and Teradata. Other trusted systems are being developed. We are fortunate to have panelists from three leading companies to discuss some of these products. All three led the security efforts in their respective companies. Jim Pierce of Teradata Corporation will discuss his modular, massively parallel database machines and will share with you future security plans of his company. Linda Vetter of Oracle Corporation will talk about her highly flexible products designed for the TCSEC C2 and B1 levels and of the two architectures of the B1 systems. She will also briefly discuss the distributed features of Version 7. Helena Winkler-Parenty of Sybase Corporation will discuss her client-server architecture and Sybase's future security plans.

A second significant gain in 1991 is the completion of the Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria. This completion required five years of work and precipitated many good debates on key issues in

database security. The lavender or near-purple color of the cover of the Interpretation is appropriate as the writers did much penance to complete this quality work. Mario Tinto, who lead the final effort and was active throughout the development of the Interpretation will discuss this work.

Gain #3 is that the evaluation of trusted database systems has been started by the National Computer Security Center. As of this writing, two products were under evaluation; others are in preparation for evaluation. It is my experience that users want trusted database systems that the Center has approved. Mike Hale, Chief of the Branch responsible for these evaluations, will discuss these evaluations.

Gain #4 is that database security is maturing somewhat as a discipline. Some very tough issues are being examined and understood. For example, we know a lot more about the causes of, the problems associated with and the potential solutions for polyinstantiation now, than when we put it in a contract to force people to look at the problem. An international panel, composed of the top researchers in this area, was held at the Workshop on the Foundations of Computer Security at Franconia, to argue this subject. Catherine Meadows, of the Naval Research Laboratory, who chaired this lively panel, will discuss the results with you

To make database security a success, we need good research and development in this area. There has been such research and development in 1991 and this is the last Gain that we wish to discuss. For example, Rome Labs is sponsoring the development of a B2 system, Oracle is examining the relationship between integrity and confidentiality and we are supporting the development of a trusted database system with A1 MAC. Bhavani Thuraisingham, of MITRE, will bring us up to date on such topics as distributed, multimedia, and object-oriented database systems.

RECENT DEVELOPMENTS IN SOME TRUSTED DATABASE MANAGEMENT SYSTEMS

Bhavani Thuraisingham, Ph.D.

The MITRE Corporation, Burlington Road, Bedford, MA

INTRODUCTION

Applications such as C3I, multimedia information processing, AI, CAD/CAM, and process control are becoming an essential part of many military operations. While relational database management systems have been adequate for present-day applications, complex operations of the future would require the power of representation of object-oriented database management systems as well as the reasoning power of deductive database systems. In addition, many military applications are being used in an increasingly distributed environment, requiring the operation of distributed database management systems. Due to the sensitivity of the data processed by military applications, it is essential to provide multilevel security for the database systems that are used in such applications. Distributed database systems, object-oriented database systems, and deductive database systems that are currently available have yet to incorporate multilevel security.

Some of our recent work in database security has been focussed on investigating multilevel security issues for these new generation database systems. Our other activities include research on trusted distributed database management systems. The ultimate objective of our research is to be able to develop intelligent database management systems which can operate in a multilevel secure distributed environment.

In this paper is given a brief overview of our work in trusted deductive database management systems, trusted object-oriented/multimedia database systems, and trusted distributed database systems. The motivation for this work as well as the background are also given.

TRUSTED DEDUCTIVE DATABASE MANAGEMENT SYSTEMS

Ever since Colmerauer and Kowalski pioneered the use of predicate logic as a programming language, Mathematical Logic has been applied to various areas of computer science such as database systems. It has not only been used as a framework to study their properties, it has also been used as a basis for developing powerful intelligent database systems. The first workshop on Logic and Databases held in France in 1977 discussed the formalisms of first order logic for database systems, which subsequently led to the formalization of relational database concepts using the proof and model theoretic results of first order logic. Further research activities contributed significantly to the development of advanced logic programming languages, inference engines for database systems, treatment of integrity constraints, and in handling negative, partial, and uncertain information. As a result, complex deduction and decision making processes have been incorporated into commercial intelligent data/knowledge base management systems available today. Such systems are called deductive database systems.

In the meantime, the recommendations of the Air Force Summer Study led to the design and development of multilevel secure relational database management systems. In such database systems, users cleared at different security levels can access and share a database with data at different sensitivity levels without violating security. Despite these advances, logic programming language research and research activities in multilevel secure database management systems remained largely separate. That is, a logic for reasoning in a multilevel environment or a logic programming system for multilevel environments is not currently available. Thus, multilevel secure database management systems lack several important features that have been successfully incorporated into conventional database management systems. They include constraint processing, deductive reasoning, and handling efficient proof procedures.

We made an early attempt in 1988 to view multilevel databases through first-order logic. Although not entirely successful, this approach helped gain an insight into utilizing formal logic to develop multilevel systems. That is, classical first-order logic, being monotonic, was found to be an inappropriate tool for formalizing concepts in multilevel databases. This is because it is possible for users at different security levels to have different views of the same entity. In other words, statements that are assumed to be true at one security level can very well be false at a different security level. Another contention is that first-order logic deals with only one universe (or world). In a multilevel database environment, there is a world corresponding to each security level. In other words, the universe in a multilevel environment is

decomposed into multiple-worlds, one for each security level. Considerations such as these have led us to believe that a special logic is needed for reasoning in a multilevel environment. From an examination of the various nonstandard logics described in the literature, none appeared capable of being used for multilevel systems. Therefore, during the past year, we have developed a logic for not only formalizing multilevel database concepts, but also for developing multilevel deductive database systems [1].

The logic that we have developed for multilevel databases is called Nonmonotonic Typed Multilevel Logic (NTML). It extends typed first-order logic to support reasoning in a multilevel environment. We have also formalized multilevel databases using NTML. In particular, the proof theoretic and model theoretic approaches for viewing multilevel databases have been studied. We have regarded security constraints, that are rules which assign security levels to the data, as integrity constraints for multilevel database systems. Techniques for integrity constraint processing have been adapted for security constraint processing. Also, the essential points towards developing a logic programming language based on NTML for developing intelligent multilevel database systems have been investigated. In addition, extensions to NTML for knowledge-based applications have also been proposed. We believe that this work provides the foundations for developing trusted deductive database management systems.

TRUSTED OBJECT-ORIENTED/MULTIMEDIA DATABASE MANAGEMENT SYSTEMS

Object-oriented systems are gaining increasing popularity due to their inherent ability to represent conceptual entities as objects, which is similar to the way humans view the world. This power of representation has led to the development of new generation applications such as CAD/CAM, Multimedia information processing, Artificial Intelligence and Process control systems. However, the increasing popularity of object-oriented database management systems should not obscure the need to maintain security of operation. That is, it is important that such systems operate securely in order to overcome any malicious corruption of data as well as to prohibit unauthorized access to and use of classified data. For many applications, it is also important to provide multilevel security. Consequently, multilevel database management systems are needed in order to ensure that users cleared to different security levels access and share a database with data at different security levels in such a way that they obtain only the data classified at or below their level.

Much of the research on trusted object-oriented database management systems has focussed on developing multilevel secure object-oriented data models. However, the data models that have been developed consider only the simple attributes of an object. For example, the title, author, publisher, and date of publication are simple attributes of a book. Such attributes can also be easily represented by a relational model. In contrast, the book cover, preface, introduction, various chapters, and references form the components of a book and cannot be treated as simple attributes of an object. The book, consisting of these components, has to be collectively treated instead as a *composite object*. Composite objects involve the IS-PART-OF relationship between objects. This relationship is based on the notion that an object is *part of* another object. Note that it is not possible to treat composite objects using a relational model without placing a tremendous burden on the application program in order to maintain the structure of the complex structures, thus conferring upon the object model another advantage over the relational model.

Multimedia systems, CAD/CAM systems, and knowledge-based systems are inherently more complex by their very nature and, therefore, can be handled effectively only if their components are treated using composite objects. For example, in multimedia systems, each document is a collection of text, graphics, images, and voice, and needs to be treated as a composite object. In a CAD/CAM system, the design of a vehicle consists of designs of its components, such as chassis, body, trunk, engine, and doors. Knowledge-based systems are being applied to a wide variety of applications in medicine, law, engineering, manufacturing, process control, library information systems, and education. These applications need to process complex structures. Therefore, support for composite objects in complex applications is essential.

Another feature that needs to be supported by an object-oriented data model is versioning, which has been neglected until now in secure models. In many object-oriented applications, such as multimedia systems and CAD/CAM, it is necessary to maintain documents and designs that evolve over time. In addition, alternate designs of an entity should also be represented because of the need for choice. If security has to be provided for these applications, then some form of version control should be supported by the model. Another advantage to providing version control for secure applications is the uniform treatment of 'cover stories' and versioning. Note that for many secure applications it may be necessary to support cover stories where users at different security levels have different views of the same entity. The version control feature supported by the model could be extended to support cover stories also.

Our recent work in trusted object-oriented database management systems is involved with developing a multilevel secure object-oriented data model with support for composite objects and versioning. In addition, we have also investigated issues on concurrency control and security constraints for trusted object-oriented systems [2]. We have specified extensions to the multilevel object-oriented data model for supporting multimedia data such as voice, text, graphics, images, and video. While the work that we have carried out is only the first step towards the development of trusted object-oriented database systems with multimedia data handling capability, it has incorporated all of the essential features of the object-oriented approach which will enable a useful trusted object-oriented database system to be developed.

TRUSTED DISTRIBUTED DATABASE MANAGEMENT SYSTEMS

The rapid growth of the networking and information processing industries has led to the development of distributed database management system prototypes and commercial distributed database management systems. In such a system, the database is stored in several computers which are interconnected by some communication media. The aim of a distributed database management system (DDBMS) is to process and communicate data in an efficient and cost-effective manner. It has been recognized that such distributed systems are vital for the efficient processing required in military as well as commercial applications. For many of these applications, it is especially important that the DDBMS should operate in a secure manner. For example, the DDBMS should allow users who are cleared at different levels access to the database consisting of data at a variety of sensitivity levels without compromising security. DDBMSs that provide multilevel user/data handling capability are called trusted distributed database systems (TDDDBMS).

Recently we have been conducting research and development activities in trusted distributed database management systems based on the relational data model. Much of our work has been focussed on a homogeneous environment [3]. This work includes (i) the design of a system architecture for a TDDDBMS, (ii) the development of a mandatory security policy for a TDDDBMS, (iii) designing approaches for multilevel data distribution, (iv) designing strategies for secure distributed query processing, (v) implementing a prototype secure distributed query processor, (vi) research on secure distributed transaction management, (vii) simulation of secure distributed concurrency control algorithms, and (viii) design and development of a prototype distributed security constraint processor.

We have also conducted a preliminary investigation on security issues for heterogeneous (also called federated) database systems. Our focus here has mainly been on schema integration issues [4]. We are also investigating other types of heterogeneity, such as handling different accreditation ranges and security policies

ACKNOWLEDGEMENTS

We gratefully acknowledge NSA, Rome Labs, SPAWAR, and CECOM for sponsoring our work in trusted deductive database systems, trusted object-oriented/multimedia database systems, and trusted distributed database systems.

REFERENCES

- [1] Thuraisingham, B., "A Nonmonotonic Typed Multilevel Logic for Secure Data/Knowledge Base Management Systems," MTR 10935, The MITRE Corporation, Bedford, MA June 1990 (a version is published in the proceedings of the 4th IEEE Computer Security Foundations Workshop, Franconia, NH, June 1991)
- [2] Thuraisingham, B., "Issues on Developing a Multilevel Secure Object-Oriented Data Model," MTP 384, The MITRE Corporation, Bedford, MA, October 1990 (a version has been accepted for publication in the *Journal of Object-Oriented Programming*)
- [3] Thuraisingham, B., "Multilevel Security Issues in Distributed Database Management Systems," MTP 297, The MITRE Corporation, Bedford, MA, July 1990 (a version has been accepted for publication in *Computers and Security Journal*).
- [4] Thuraisingham, B., "Schema Integration in Secure Heterogeneous Database Systems," Accepted for publication in *Database Programming and Design*, 1991.

EXECUTIVE SUMMARY

Oracle and Security: Year in Review 1990-91

Linda L. Vetter, Director, Oracle Secure Systems

Oracle Secure Systems, formed in February 1989, is chartered with spearheading Oracle Corporation's efforts to research, design, build and deliver high security relational database management system (RDBMS) products and services to commercial and government organizations worldwide. The past year for the Secure Systems division has been marked by a number of major milestones. During the year, Oracle made substantial progress in improving and refining its multilevel secure (MLS) relational database management system, Trusted ORACLE RDBMS Version 1.0, in preparation for its upcoming commercial release. In addition, Oracle Secure Systems contributed significantly in the area of standards creation, MLS application development, and other areas.

Research and Development Contracts

Oracle Corporation continues to participate in a number of secure RDBMS projects within the federal government. Oracle is working with Gemini Computers of Monterey, California, to complete its MLS RDBMS contract with the NCSC. Oracle and Gemini are in the process of porting Trusted ORACLE to the Gemini A1-targeted platform. Work is progressing satisfactorily on this challenging task. In conjunction with this contract with the NCSC, Oracle's MLS RDBMS was the focus of a day long Technical Review Group (TRG) meeting in Bethesda, Maryland, in January 1991. Updated versions of MLS Oracle systems, documentation, and test facilities have been delivered to NCSC during the year.

Oracle, SRI, and Gemini also have continued efforts related to their Air Force RADC SeaView contract. This effort was publicly presented at a peer review meeting in Oakland in May 1991. This work also has been proceeding well with various interim deliverables completed during the year.

Standards, Portability and Interoperability

1991 has been a noteworthy year for the development of standards for multilevel secure database management systems. The Trusted Database Interpretation (TDI) of the Trusted Computer System Evaluation Criteria was published in April 1991. Oracle played an active role during the development of the TDI by participating in the various TDI working groups and fora over the past few years. In addition, Oracle has actively participated in the review of the Information Technology Security Evaluation Criteria (ITSEC), the harmonized criteria of France, Germany, Holland and the United Kingdom through the submission of written comments on each draft and attendance at ITSEC workshops in Brussels, Belgium.

The Secure Systems group also currently participates in standards committees concerned with multilevel secure DBMS application portability and interoperability issues. Primary among these is the POSIX 1003.6 Security Working Group and the Trusted Systems Interoperability Group (TSIG). The POSIX 1003.6 effort defines an interface for security functions for a portable operating system. The TSIG focuses on interoperability issues in trusted network environments. Oracle also closely follows and contributes to other standards initiatives in order to conform with as many standards as possible. For example, Oracle's submission of a group access control feature called "roles" has been accepted as a part of a future ANSI SQL standard.

Market Requirements

The acceptance of advanced security software products on a broad basis will require products that are easy to use while providing high functionality and security. Oracle has spent substantial effort trying to gauge the needs of potential users of MLS DBMS products to identify the requisite mix of functionality

and security needed by the marketplace.

The Oracle Security Advisory Committee (ORASAC) was formed this year to provide a channel of communication between Oracle and potential users of MLS RDBMS products. ORASAC is a committee composed of members of Oracle Secure Systems and representatives from government and industry who meet on a periodic basis to exchange information on the development and implementation of Trusted ORACLE RDBMS and related applications. ORASAC has proven to be a successful vehicle for requirement gathering, implementation analysis, and educational exchange for both Oracle Corporation and ORASAC members.

Evaluation

Trusted ORACLE RDBMS Version 1.0 (target Class B1) and ORACLE RDBMS Version 7.0 (target Class C2) were accepted into the NCSC's Trusted Product Evaluation Program in June of 1991. Oracle Corporation is pleased to be participating in the program and is proud to have two products under evaluation. The initial evaluation platform is Hewlett-Packard's HP-UX BLS 8.04 multilevel secure UNIX operating system. Multiple meetings between Oracle Secure Systems and the NCSC evaluation team already have been held and extensive documentation delivered to team members.

Technology

Development of Trusted ORACLE V1.0 has progressed tremendously over the past year as it prepares to enter its beta testing phase. Users will have the option of implementing Trusted ORACLE database applications using one of two modes: operating system constrained mandatory access control (MAC) enforcement or RDBMS/trusted subject-enforced MAC.

Trusted ORACLE RDBMS has many advantages regardless of which run-time mode is chosen, and in both cases users see data classifications maintained at the individual row level. Trusted ORACLE minimizes redundancies between the operating system (OS) and RDBMS, for example, user identification and authentication is defined at the OS level and is not duplicated within the RDBMS. In addition, valid sensitivity labels and their dominance relations are defined and modified via MLS OS facilities, thus eliminating the need to re-define or re-implement such functions within the RDBMS. In general, Trusted ORACLE provides efficient integration with OS security mechanisms, maximum portability, hardware configuration flexibility, standards compliance, and functionality.

The OS MAC mode requires explicit isolation of each component of the system, the secure operating system, DBMS, or network for example, with each component enforcing a specific portion of the overall security policy. This implementation requires that the security mechanisms of a component be constrained from bypassing or re-implementing any of the security mechanisms of any more primitive (underlying) component of the secure system - i.e., the DBMS must run without any special OS security privileges. This mode is designed to meet the TDI requirements for "Two TCB Subsets Which Meet the Conditions."

There are certain environments in which the OS MAC mode can be particularly advantageous: where there is a high proportion of low-level to high-level data; where there is a high proportion of single-level applications or users; where there is a small number of data sensitivity labels; where requirements for high assurance MAC apply; and/or where pre-certified, heterogeneous hardware configurations exist.

Trusted ORACLE RDBMS Version 1.0 also provides selective MAC enforcement within the RDBMS itself. Trusted ORACLE must operate as a "trusted subject" when configured to run in this mode; that is, Trusted ORACLE must operate with one or more OS security privileges enabled (e.g., to allow apparent "down-grades").

In this mode, the data sensitivity labels are physically stored within the database and are provided by Trusted ORACLE upon subsequent use of the data. Trusted ORACLE still enforces discretionary access controls on database named objects as before, but it uses the data sensitivity labels to enforce mandatory access controls itself, only relying on the secure operating system to provide label and dominance definitions. Trusted ORACLE logical database storage objects still map directly to one or more physical storage objects, however, when running in this mode the storage objects may contain multilevel data; OS storage objects (e.g., entire files) are labeled at the highest level of data contained within the object. In other words, a "database high" file will contain multilevel labeled data, for example rows at secret, confidential and unclassified.

There are certain environments in which the DB MAC mode will be particularly advantageous: where large numbers of data sensitivity labels are needed; where numerous applications or users require multiple levels of data simultaneously; and/or where multilevel referential and entity integrity enforcement justifies partial relaxation of strict MAC enforcement.

Application Development Analysis and Technology Transfer

Oracle Secure Systems this year also continued analysis of the considerations and implications of application development in an MLS RDBMS environment. One example of this effort was well received in a research paper presented at the Fourth RADC Workshop in Database Security which described the conflicts between enforcing strict mandatory access controls and enforcing multilevel integrity in an MLS RDBMS. Topics discussed included entity integrity, referential integrity, transaction integrity and value constraints enforcement and tradeoffs to consider between integrity and strict MAC security enforcement. Information of this type should help application developers achieve more satisfactory results in initial MLS application design.

Oracle has been developing multiple ways to ensure the successful transfer of new MLS RDBMS technology to users. Secure Systems has created and taught introductory courses on Trusted ORACLE RDBMS and on MLS application design to internal staff and customers around the world during the past year. In addition, new requirements for support staff involved in sensitive security work are being addressed.

Overall, Oracle Corporation has continued its multi-faceted approach to database security, making significant progress during the year in addressing research and development, standards, requirements, product evaluations, and technology transfer issues.

1991 SYBASE Secure Products: Executive Summary

Helena B. Winkler-Parenty

**Sybase, Inc.
6475 Christie Avenue
Emeryville, CA 94608**

Overview

During the past year Sybase has continued its long standing commitment to building trusted products. In addition to supporting the B1-targeted SYBASE Secure SQL Server™ and SYBASE Secure SQL Toolset™, which have been generally available for two years, Sybase has been developing two other secure products. Sybase is currently working on both a C2-targeted upgrade to the standard SYBASE SQL Server™ and a second and considerably more powerful release of the B1-targeted Secure SQL Server.

C2 Targeted DBMS

Sybase is currently modifying its standard SQL Server to comply with the Trusted Database Interpretation (TDI) at the C2 level. The standard SQL Server already contains a mechanism which allows users to define Discretionary Access Controls on objects that they own. Database owners can grant or revoke the privilege to use a database or create tables in it. Table and view owners can grant and revoke the privilege to Select, Update, Insert or Delete rows from a table. In addition, Select and Update protections can be applied to individual columns within a table. Owners of stored procedures determine who has execute permission on their stored procedures. The DBMS validates each user's request against the permissions that appear in the access control lists that are associated with each database, table, view, and stored procedure.

SYBASE provides three distinct roles: System Security Officer (SSO), System Administrator (SA), and Operator (Oper). These roles allow multiple users to be given SSO, SA, or Oper privileges, without losing individual accountability. By dividing the system privileges into three categories, viz. security relevant, system administration, and backup, SYBASE allows for more finely grained control than is traditionally provided.

Auditing is an important component of a trusted system. An auditing mechanism is being incorporated into the SQL Server that is tailored to the requirements of a relational DBMS. Through this, security relevant system activity is recorded in an audit trail, which can be used to detect attempted misuse or penetration of the system. The SQL Server and the Secure SQL Server have extensive auditing capabilities. Events are audited at the discretion of the SSO, permitting auditing to be customized to the needs of individual installations. Examples of auditable events are: specific user's queries, and all user access to specified databases or tables. The SSO can employ the full power of Transact-SQL™, Sybase's extended SQL language, and the Secure SQL Toolset to review the audit trail, greatly reducing the effort usually associated with this task.

B1 Targeted DBMS

The next release of the Secure SQL Server will provide all of the capabilities of the standard SQL Server plus the additional requirements of the B1 level of trust. This release builds on Sybase's customer experience with the previous release of the Secure SQL Server, and provides significantly more powerful capabilities. The next release of the Secure SQL Server will contain all of the features discussed above for the C2 targeted SQL Server, in addition to the B1 specific features mentioned in this section.

SYBASE augments a multilevel secure (MLS) operating system's Trusted Computing Base (TCB) with the trusted subject Secure SQL Server. The Secure SQL Server enforces DBMS mandatory access control by labeling all DBMS subjects (processes) and storage objects (rows), and mediating all accesses between DBMS subjects and objects based on their security labels. The MLS operating system, on which the Secure SQL Server is running, provides mandatory access control for operating system objects, typically files or segments and protects the DBMS itself.

The Secure SQL Server's security policy is based on the widely accepted Bell-LaPadula Model. In order to select data, the user's security level must dominate the security level of the rows being accessed, otherwise they will not be retrieved. Updated and inserted rows inherit the security level of the user performing the operation. Sybase's mandatory access control goes beyond the B1 level and applies to all objects, even the data dictionary, so that authorized users will not even be aware of the existence of tables or databases that they are not authorized to see.

The auditing mechanism of the SQL Server is enhanced in the B1-targeted product with the inclusion of security labels and mandatory access control. Row access can be audited based on either the identity of the user performing the access or the table in which they are contained. To minimize the number of rows that are audited a minimum row security level can be specified and only the access to rows with at least this classification will be audited.

Conclusion

Sybase has pioneered the Client/Server Architecture, Server Enforced Integrity, and the Trusted Subject Architecture. In 1991 Sybase is developing trusted products at two different levels of trust, the standard SQL Server and the Secure SQL Server. These are designed to meet the C2 and B1 levels of trust respectively. Sybase is expanding upon its original Secure SQL Server product to better meet the needs of industry and government.

Executive Summary

Panel: Requirements and Experiences

**Dennis Gilbert, Moderator
National Institute of Standards and Technology**

Panel Members:

Kenneth Cutler, American Express

David Ferraiolo, NIST

Michael Ressler, Bellcore

Aylen Hasagawa, Allstate

Hal Tipton, Rockwell International

Until recently, the U.S. government's view of "trusted" technology for computer and communications systems related largely to preserving national security. The view heavily emphasized the security requirement of confidentiality--preventing unauthorized disclosure. Recently, however, the government is paying increasing attention to other computer security requirements, such as integrity and availability. In addition, trusted technology is being explored for protection of unclassified information of various types in civil agencies. Efforts are in progress to further broaden the notion of trust to include safety and reliability. There are signs that such requirements are increasingly important to users of both government and commercial systems.

System users need standards and guidance that move beyond the current DoD Trusted Computer System Evaluation Criteria (TCSEC or Orange Book) approval. They look for standards and guidance which support the production of more robust, trustworthy systems which address the full range of security requirements.

NIST conducted a study to help it better understand and meet federal needs for protecting computer-based information. In the project, which involved the cooperative effort of people from over two dozen government and industry organizations, NIST looked at technical information protection methods used in computers or application systems. The study explored organizations' experience in developing trust or reliance on information systems which are important to the organization's mission, including safety-critical systems. Participants represented a wide variety of perspectives, environments, application, and system architectures.

A primary goal of the project was to identify requirements for new federal standards and guidance documents on protection of sensitive and critical information in systems of the 1990's. The project drew upon the significant experiences of many organizations in specifying, implementing, and using computer-based information system protection mechanisms. These experiences are helping NIST identify security requirements and develop near-term guidance on the effective use of commercial security products. NIST expects that commercial and other private sector organizations, having given significant input to the requirements, will consider adoption of the standards when they are developed.

Another primary aim of the project was to determine whether a core set of broadly-applicable information protection objectives and technical requirements exists. These requirements would form the basis for trusting the security capabilities of systems and products that implement them. This is true when the requirements are implemented in commercial products and federal systems and supported by appropriate methods for determining their correctness and effectiveness.

In a similar vein, the National Research Council's System Study Committee, in its report "Computers at Risk," recommended the promulgation of comprehensive generally accepted system security principles (GSSP). The GSSP would be "a basic set of security-related principles that are so broadly applicable and effective for the design and use of systems that they ought to be part of any system with significant operational requirements." Efforts by NIST and others are underway to explore these and related issues, and to coordinate these activities.

This session presents the results of the NIST study of organizations' requirements and experiences described above. It also brings together several participants actively attempting to define the core set of information protection requirements. They present a status report and discuss the significant issues and challenges.

Executive Summary

Panel: Risk Management

Irene Gilbert, Moderator

National Institute of Standards and Technology

Panel Speakers

Suzanne Smith, Los Alamos National Laboratory

Deb Bodeau, The MITRE Corporation

H. Carol Bernstein, IBM Laboratory Council

The operation, protection, and management of automated information systems has become critical in the 1990's. Business and organizations are increasingly recognizing the importance of protecting information systems as evidenced by recent laws, policy, directives, and guidelines. We must not only ensure that appropriate security controls are in place, we must also address business categories that have a large impact on the survivability of our organizations.

This panel will discuss the legal aspects of computer security, general liability concerns, and insurance issues in the 90s. The greatest return on limited financial resources and manpower can be realized only when we carefully select and implement appropriate controls as they apply to the following business categories:

- Legal
 - Compliance
 - Policy
 - Directives
 - Federal law
 - State and Local statutes
- Liability
 - Financial
 - Safety
 - Reliability
- Insurance
 - Hardware
 - Facility
 - Warranties
 - Operation
 - Service
 - Availability

PANEL: Specifying, Procuring, and Accrediting MLS System Solutions

Joel E. Sachs
Arca Systems, Inc.
2841 Junction Ave., Suite 201
San Jose, CA 95134
408-434-6633

Panel Overview

Both the availability of MLS products and attempts at procuring MLS system solutions have increased in recent years. Several of these procurements have already been deemed less than successful. A number of reasons have been suggested: integration of these products is not straight forward, defining and mapping solution requirements to them is difficult, and certification and accreditation are hard and not uniform. Procuring an MLS system solution that results in an accreditable secure solution is not simple; moreover, there is debate and confusion as to what should be specified during the initial phases of a procurement that will help all parties involved throughout the life of the program. This panel will explore issues associated with developing a specification, statement of work, and evaluation criteria for procuring an MLS System Solution successfully. The critical deliverables and their role in certification and accreditation will also be examined.

The panel will explore these issues by role playing the various parties in the procurement process, as opinions vary depending on one's position within the process. Each of the seven panelists will act on the behalf of an identified role. These roles are: End-User Organization, Program Management Office, Advising Security Agency (and also Certification Body), Designated Approving Authority, Systems Integrator, Security Engineering Subcontractor, Vendor. The panel will discuss the issues associated with the pre-draft RFP, pre-RFP, pre-award, and post-award phases of an MLS System Solution procurement. A specific example problem will be used as a case study. The panelists will discuss and debate their needs and concerns regarding the development of a MLS System Solution, with respect to the role that they are playing. Specific questions will be asked of the panel relative to each procurement phase.

Information is provided in the following sections to aid the audience with a preliminary understanding of the topics and issues of specifying, procuring, and accrediting MLS System Solutions. These Sections include a description of the example MLS problem to be considered by the panel, example issues and concerns of the various parties, example critical questions for the panel, as well as a paper entitled "*A Framework For Developing Accreditable MLS AISs*".

MLS Case Study Problem Description

The panel will consider the following problem: An end-user organization would like to have automated support for their analysis, planning, and operations activities. The users are distinguished by the jobs they are authorized to perform, i.e., analysts, planners, and operations personnel. All users have at least a Secret clearance; some have a Top Secret clearance. This system is to be developed and fielded in two phases. In the first phase, these three activities are to be done using segregated processing in order to keep these activities and their results separated from one another. The analysis data is Top Secret. The planning and operations data are Secret but must be kept separate.

The various users are spread throughout a closed facility. The majority of the data lends itself to be handled by a DBMS. Moreover, the data content usually stays constant as it evolves from the analysis to the operations stage. However, some data does not move to the next stage and other data is added at the next stage. In addition, the system must support the ability to make external connections to Top Secret systems to allow the import of Top Secret information for analysis.

The second phase of this system is to provide the capability for a single user to simultaneously do either a) analysis and planning, or b) planning and operations, but to disallow both analysis and operations to be conducted together in a single session. The purpose here is to permit selected planners to review new analysis information to update current plans and to allow selected operations personnel to update plans based on operational status. In addition to these changes, the second phase must also support bidirectional communications on external connections to permit the export of plans and operations as well as the import of analysis data.

As additional considerations, i.e., options, the end-user organization is interested in two things. One is a simplified downgrading process, e.g., a "single button" to move a developed analysis stripped of strictly Top Secret data into a plan. The other is to utilize existing ADP resources in the new system.

Panel Roles, Descriptions, and Areas of Concern

End-User Organization

The end user organization has a requirement for a system solution. The results of this procurement will be delivered to this organization for their use.

Their main concerns are how to ensure that they get what they want, that it will be accreditable, and how much will it cost? They usually understand functional requirements reasonably well but often do not understand security and assurance requirements and security issues.

Program Manager's Office [PMO]

The PMO is responsible for writing the RFP, awarding the contract, and supervising its execution. (Typically, a separate organization might be used to develop a system specification for the SOW. For the purposes of this panel, the specifier will be considered merged with the PMO.)

The PMO's main concerns are system specification, cost, schedule, accreditation, and measuring the prime contractor's progress and compliance. The PMO understands the functional requirements as communicated by the end-users, but may not fully understand the security requirements, issues, and assurance needs that result from the mission and threat context.

Advising Security Agency / Certification Body

The Advising Security Agency is the End-User's and/or PMO's security arm that helps monitor the progress of the program to ensure that security within the program is adequately addressed. The Certification Body gathers the assurance evidence and performs risk analyses on the system. (For the purposes of this panel, these two roles have been combined as often happens in practice.)

Their main concern is whether the delivered system meets the security requirements specified in the RFP, security functionality and assurance. The certification body must provide enough evidence to allow the DAA to make a proper decision regarding its accreditation.

Designated Approving Authority [DAA]

The DAA is the individual responsible for the operational aspects of the system. It is this individual's responsibility to approve the system for operation.

The DAA's main concern is whether the system meets its operational requirements and its operational risk has been reduced to an acceptable level. Based on the evidence provided during the certification process, the DAA must make a decision whether the operational risk is acceptable given the evidence provided and the system's mission, and accredit or fail the system for

operation. The DAA's accreditation of the system is his indication that he feels the risk is low enough or the operational need is so high to allow the system to operate.

Systems Integrator

The Systems Integrator is responsible for the development and integration of the end-system as well as the management of all the subcontractors involved in the effort.

Their main concerns are how to provide the required functionality, security, and assurance within the budgetary and time constraints stipulated in the integrator's proposal. Other areas of concern include how to manage the security engineering effort to produce a functional and useable system and how to handle requested changes to the end-system.

Security Engineering Group/Subcontractor

Security Engineering is responsible for the security portion of the overall system development. This team is composed of internal systems integrator personnel, a security subcontractor, or a combination of both.

This team's main concerns are: how to relate component policies to the overall system policy, the trust requirements for each component, how to integrate trusted and untrusted systems, how to integrate multiple products into a single secure solution, and how to provide required assurance evidence. They may also be involved in determining the security requirements and policy, determining the appropriate assurance level, and how to provide assurance evidence.

Vendor

Vendors provide products that are used as part of end-user system solutions.

Their main issues are: how to relate their product features to the desired functionality and assurances needed within an MLS system solution and how to advise the systems integrator on the best use of these features.

Example Questions for Panel

Pre-Draft RFP Questions:

- 1) Should SOW explicitly state detailed security requirements, e.g., require either a compartment for the planning data or DAC, or just simply state need to segregate planning from operations data?
- 2) How should the SOW handle the migration of analysis data to planning data to operational data (i.e, the downgrading / transmission issue)? Should a trusted application be explicitly required?
- 3) What can be done at this stage to ease the certification / accreditation process? Who should do it? How should it be requested?
- 4) How should threats be determined and documented? What information about threats should be provided to prospective bidders?
- 5) Who should identify or develop the following
 - Assurance Requirements
 - Assurance Deliverable Schedule
 - Security Architecture
 - MLS Concept of Operations

- System-Wide Security Policy
- Certification and Accreditation Plan
- System-Wide Security Policy Model
- System Threat List and Risk Analysis

Who provides inputs, who writes, who reviews, who is the intended audience? When should these be done? Should the SOW be explicit? What should the DIDs require?

Considerations: a) It's more work for either the Specifier, PMO, Certification Body, or Systems Integrator; b) Not everything is known upfront; c) If not done up front, bidders get to decide what is required, and some may use this flexibility to undercut other bidders by potentially deriving insufficient requirements.

Pre-RFP Questions:

- 6) Who should develop/determine the MLS Concept of Operations? The PMO, Advising Security Agency, or Systems Integrator? When?
- 7) What steps can be taken to ensure that an MLS system solution is proposed, not just an MLS operating system?
- 8) When should the Advising Security Agency, Certification Body, or DAA become involved? How and to what degree? At different stages who are they helping and to whom are they responsible? Should this be reflected in the RFP and SOW? How?
- 9) How and when should the overall assurance requirements be given? How should they be determined?
- 10) Should a Certification and Accreditation Plan be included in the RFP? If not, when should it be developed? How should it be specified that the system must be certifiable or accreditable?

Pre-Award Questions:

- 11) Should Certification and Accreditation be addressed in the Proposal? How?
- 12) Which factors should be considered in the proposal evaluation criteria?
 - a) the Technical approach? methodologies? architectures? trade-offs?
 - b) the Assurance / Certification and Accreditation approach?
 - c) the Participating Personnel?
- 13) As engineering process capability testing becomes routine, should security tests and exercises be administered as part of the evaluation of the bidders? If so, how should tests be given? If so, who should take the test? Should it be a group test?

Post-Award Questions:

- 14) How should the DAAs of the external systems to which the proposed system connects be dealt with?
- 15) How should the detailed security requirements be determined? How and when should they be delivered?
- 16) At what times within the development / certification process should assurance evidence be provided? Who is to review this evidence? How should it be developed?
- 17) How should component policies be related to an overall system policy?

- 18) How should assurance evidence be generated for an MLS System Solution that is composed of multiple trusted and untrusted products?**
- 19) How can vendors provide functional capability to assist in the integration of their products into the system solution?**
- 20) What assurance evidence can a vendor provide that enhances a product's appeal for use in a secure system solution for the System Integrator, Security Subcontractor, or Certifier / DAA?**

Executive Summary

TRUSTED APPLICATIONS IN THE REAL WORLD

Stephen T. Walker, Moderator

Trusted Information Systems, Inc.

3060 Washington Road (Rt. 97)

Glenwood, MD 21738

(301) 854-6889

Panelists

Sam Doncaster, Digital Equipment Corporation

Mal Fordham, Grumman Data Systems

Helmut Stiegler, Siemens Nixdorf

Clark Weissman, Unisys

After ten years of trusted system development by computer vendors and system integrators, it is time to gather together our thoughts and experiences into a set of lessons learned and common sense guidance.

This session will highlight the practical insight of a highly experienced set of system implementors and vendors from the U.S. and Europe and identify where things have gone well or badly and why. The session will begin with short summaries of each speaker's experiences and will then move to a panel discussion with questions and comments from the audience to highlight our collective wisdom from efforts of the past ten years.

Individuals seeking insight into practical experiences with applying trusted systems and those with experiences to contribute are encouraged to attend this session.

This session expands upon the issues and topics developed in the **Fielding COTS Multilevel Security Solutions: The Next Step** which occurs 1400-1530 on 4 October 1991 in the Blue Room.

Executive Summary

WINNING STRATEGIES IN INFORMATION SYSTEMS SECURITY EDUCATION, TRAINING, AND AWARENESS

A panel discussion of programs which have met with success in implementing the education, training and awareness provisions of PL 100-235, the Computer Security Act of 1987.

Moderator: W.V. Maconachy, Ph.D.
Chairman, National Computer Security Educators' Group

Program Summary

This program is sponsored by The National Computer Security Educators' Group (NCSEG). The program will serve as a forum for practitioners in computer security education, training, and awareness (ETA) to present their views on workforce ETA. The panel participants represent a cross section of government and private sector experts who are implementing ETA programs in their organizations. During the discussions, the panel members will illustrate how they are reaching their respective workforce with COMPUSEC ETA programs. The discussions will be open to the audience for debate, additional information, and other points of view.

Discussion

It has been several years since the passage of the Computer Security Act of 1987. The act prescribes certain measures be taken by federal agencies to ensure the security of computers and computer systems which contain government information. This mandate from Congress has resulted in plethora of activity by federal agencies as they each, independently, respond to the spirit as well as the letter of the law. However, lots of activity may not equate to movement; or at least movement in a specified direction. One of those unspecified directions is the area of providing COMPUSEC ETA to the federal workforce. This program is one of a series of activities sponsored by the NCSEG that strives to provide the thread of continuity needed in the federal community to guide those implementing the ETA requirements of PL100-235.

[

