

*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 II**

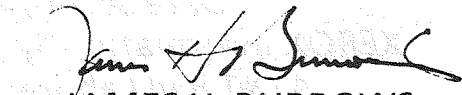
**Information Systems Security:  
Requirements & Practices**

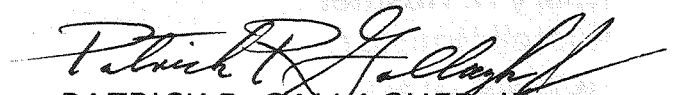
## Welcome!

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

# Conference

<b>Dr. Marshall Abrams</b>	<i>The MITRE Corporation</i>
<b>James P. Anderson</b>	<i>J.P.Anderson Company</i>
<b>Jon Arneson</b>	<i>National Institute of Standards and Technology</i>
<b>Devolyn Arnold</b>	<i>Department of Defense</i>
<b>James Arnold</b>	<i>Department of Defense</i>
<b>Al Arsenault</b>	<i>Air Force Academy</i>
<b>V.A. Ashby</b>	<i>The MITRE Corporation</i>
<b>David Balenson</b>	<i>Trusted Information Systems, Inc.</i>
<b>Dr. D. Elliott Bell</b>	<i>Trusted Information Systems, Inc.</i>
<b>James W. Birch</b>	<i>Secure Systems, Inc.</i>
<b>W.Earl Boebert</b>	<i>Secure Computing Technology Corporation</i>
<b>Dr. Martha Branstad</b>	<i>Trusted Information Systems, Inc.</i>
<b>Dr. John Campbell</b>	<i>Department of Defense</i>
<b>Lisa Carnahan</b>	<i>National Institute of Standards and Technology</i>
<b>R.O. Chester</b>	<i>Martin Marietta</i>
<b>David Chizmadia</b>	<i>Department of Defense</i>
<b>Dorothea deZafra</b>	<i>Public Health Service</i>
<b>Donna Dodson</b>	<i>National Institute of Standards and Technology</i>
<b>Karen Doty</b>	<i>CISEC</i>
<b>Dr. Deboah Downs</b>	<i>The AEROSPACE Corporation</i>
<b>Jared Dreicer</b>	<i>Los Alamos National Laboatory</i>
<b>Ellen Flahavin</b>	<i>National Institute of Standards and Technology</i>
<b>Daniel Gambel</b>	<i>Grumann Data Systems</i>
<b>L. Dain Gary</b>	<i>Mellon National Bank</i>
<b>Virgil Gibson</b>	<i>Grumann Data Systems</i>
<b>Dennis Gilbert</b>	<i>National Institute of Standards and Technology</i>
<b>Irene Gilbert</b>	<i>National Institute of Standards and Technology</i>
<b>Captain James Goldston, USAF</b>	<i>AFCSC</i>
<b>Dr. Joshua Guttman</b>	<i>The MITRE Corporation</i>
<b>Douglas Hardie</b>	<i>Unisys Corporation</i>
<b>Ronda Henning</b>	<i>Harris Corporation</i>
<b>Dr. Harold Highland, FICS</b>	<i>Compulit, Inc.</i>
<b>Jack Holleran</b>	<i>National Computer Security Center</i>
<b>Hilary H. Hosmer</b>	<i>Data Security, Inc.</i>
<b>Russell Housley</b>	<i>XEROX Information Systems</i>
<b>Howard Israel</b>	<i>AT&amp;T Bell Laboratories</i>
<b>Dr. Sushil Jajodia</b>	<i>George Mason University</i>
<b>Wayne Jansen</b>	<i>National Institute of Standards and Technology</i>

# Referees

Carole Jordan	Defense Investigative Service
Dr. Maria M. King	The AEROSPACE Corporation
Leslee LaFountain	Department of Defense
Steven LaFountain	Department of Defense
Paul A. Lambert	Motorola GEG
Dr. Carl Landwehr	Naval Research Laboratory
Robert Lau	Department of Defense
Dr. Theodore M.P. Lee	Trusted Information Systems, Inc.
Steven B. Lipner	Digital Equipment Corporation
Teresa Lunt	SRI International
Dr. William V. Maconachy	National Security Agency
Sally Meglathery	ISSA
Dr. Jonathan Millen	The MITRE Corporation
Warren Monroe	Hughes Aircraft
William H. Murray	Deloitte & Touche
Noel Nazario	National Institute of Standards and Technology
Ruth Nelson	GTE
Peter Neumann	SRI International
J.D. Nichols	Independent Consultant
Steven Padilla	SPARTA
Nick Pantiuk	Grumann Data Systems
Donn Parker	SRI International
Richard Pethia	Carnegie Mellon University
Dr. Charles Pfleeger	Trusted Information Systems, Inc.
Kenneth Rowe	Department of Defense
Professor Ravi Sandhu	George Mason University
Marvin Schaefer	Trusted Information Systems, Inc.
Dr. Roger R. Schell	GEMINI
Emilie J. Siarkiewicz	Rome Air Defense Center
Suzanne Smith	Los Alamos National Laboratory
Brian Snow	Department of Defense
Professor Eugene Spafford	Purdue University
Mario Tinto	Department of Defense
James Tippett	Department of Defense
Eugene Troy	National Institute of Standards and Technology
LTC. R. Vaughn, USA	U.S. Naval Academy
Grant Wagner	Department of Defense
Kenneth vanWyk	Carnegie Mellon University
Howard Weiss	SPARTA
Roy Wood	Department of Defense
Carol Worden	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. Fatcher, 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  
*Lorraine 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
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
Pfleger, 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, Lorraine J.	188	Wolcott, Dawn	25
Schanzer, Steven	655	Yasaki, Brian K.	78
Scherer, Joseph	630		
Schou, Corey D.	305		

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100

# A Method to Detect Intrusive Activity in a Networked Environment<sup>1</sup>

*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

---

<sup>1</sup> 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.

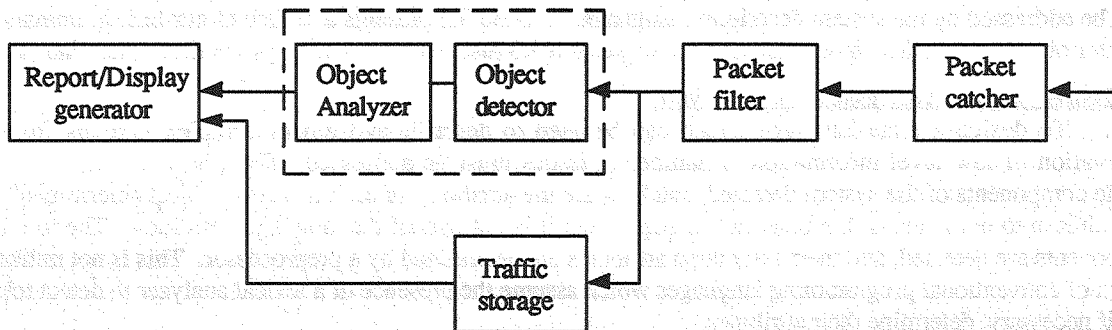


Figure 1

### 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.

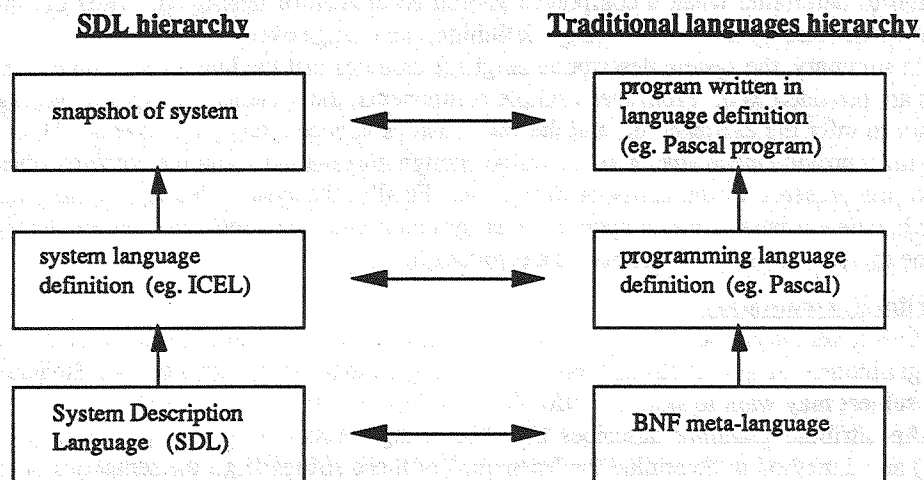


Figure 2

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 numbers<sup>2</sup> will be used to clarify the relationships between these three

---

<sup>2</sup> This example is taken from [12].

$N \rightarrow L.L$	$N \rightarrow L_1.L_2$	$v(N) = v(L_1) + v(L_2)/2^{l(L_2)}$
$N \rightarrow L$	$N \rightarrow L$	$v(N) = v(L)$
$L \rightarrow LB$	$L_1 \rightarrow L_2B$	$v(L_1) = 2v(L_2) + v(B), l(L_1) = l(L_2)+1$
$L \rightarrow B$	$L \rightarrow B$	$v(L) = v(B), l(L) = 1$
$B \rightarrow 1$	$B \rightarrow 1$	$v(B) = 1$
$B \rightarrow 0$	$B \rightarrow 0$	$v(B) = 0$
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.

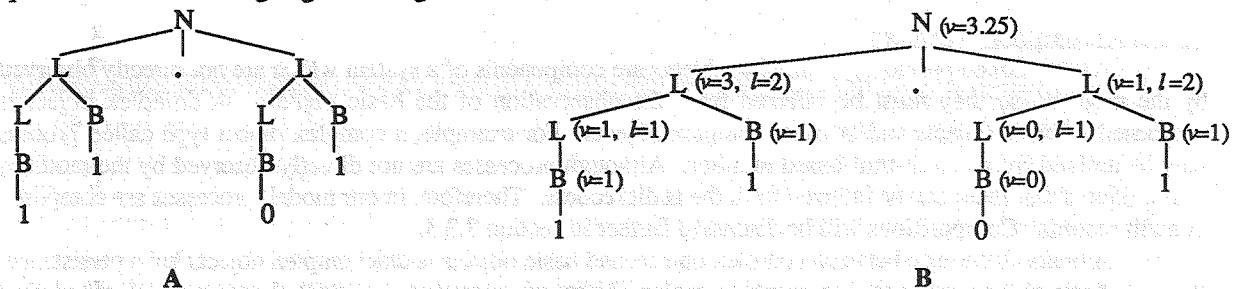


Figure 4

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,  $\Delta t$ , unless another object which is part of B's composition is created or observed. This specified  $\Delta t$  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  $A_1$ .

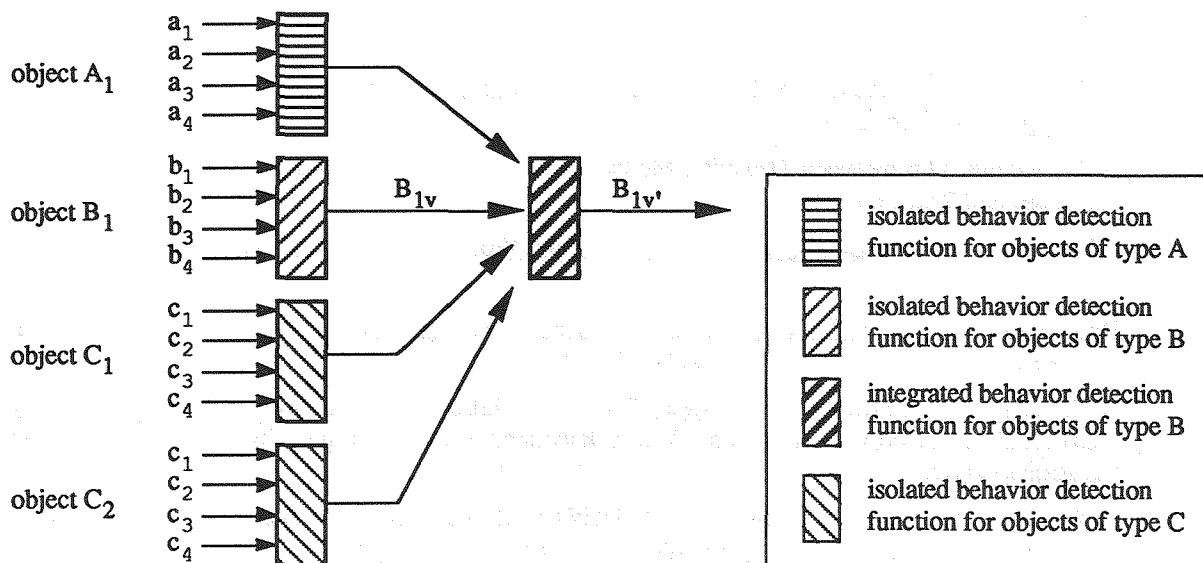


Figure 5

## 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 IDES 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.

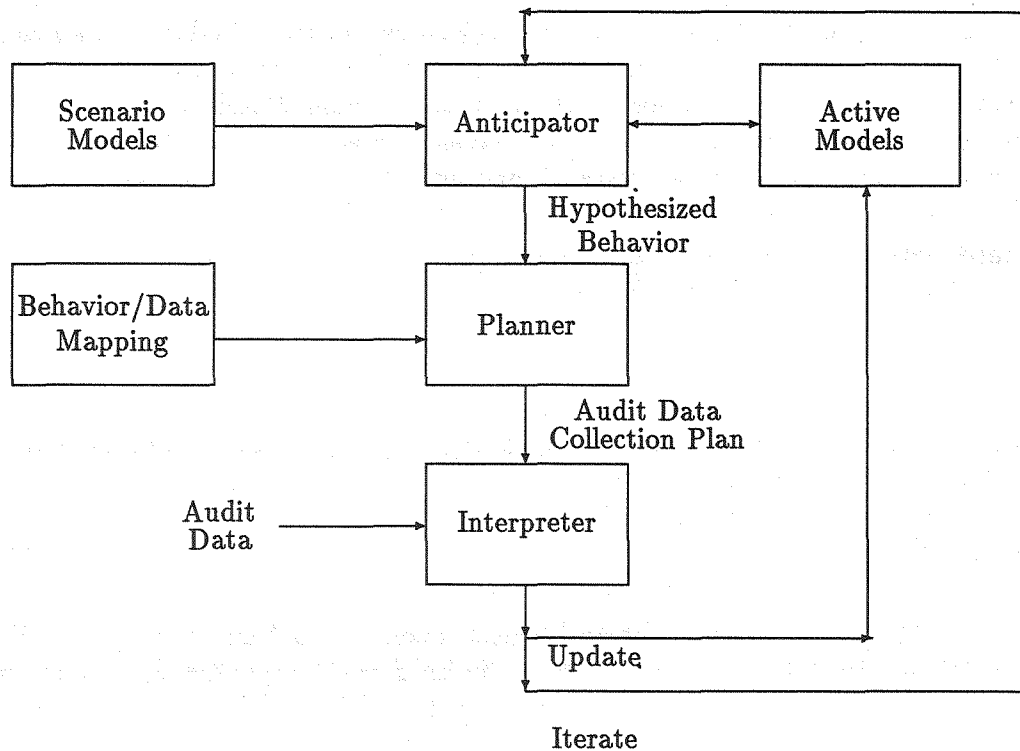


Figure 1: Model-Based Reasoning Approach

- 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 Gister<sup>1</sup> evidential reasoning system for the fusion and interpretation of evidence for hypothesized intrusions. This process takes place in the

<sup>1</sup>Gister 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 manyfold, 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,  $m_{\Theta}$ , is a mapping from subsets of a frame of discernment,  $\Theta$ , 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  $\Theta_A$  and  $\Theta_B$  is a set of pairs such that

$$\Theta_{A,B} \subseteq \Theta_A \times \Theta_B,$$

where every element of  $\Theta_A$  and every element of  $\Theta_B$  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:

$$m_{\Theta}^3(A_h) = \frac{1}{1-k} \sum_{A_i \cap A_j = A_h} m_{\Theta}^1(A_i) m_{\Theta}^2(A_j), \quad (2)$$

$$k = \sum_{A_i \cap A_j = \phi} m_{\Theta}^1(A_i) m_{\Theta}^2(A_j).$$

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  $\Theta_A$  to frame  $\Theta_B$  using the compatibility relation  $\Theta_{A,B}$  is defined by:

$$m_{\Theta_B}(B_j) = \sum_{\substack{C_{A \rightarrow B}(A_k) = B_j \\ A_k \subseteq \Theta_A, B_j \subseteq \Theta_B}} m_{\Theta_A}(A_k), \quad (3)$$

where  $C_{A \rightarrow B}(A_k) = \{b_j | (a_i, b_j) \in \Theta_{A,B}, a_i \in A_k\}$ .

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

$$m_{\Theta}^{\text{discounted}}(A_j) = \begin{cases} \alpha \cdot m_{\Theta}(A_j), & A_j \neq \Theta \\ 1 - \alpha + \alpha \cdot m_{\Theta}(\Theta), & \text{otherwise} \end{cases} \quad (4)$$

where  $\alpha$  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.

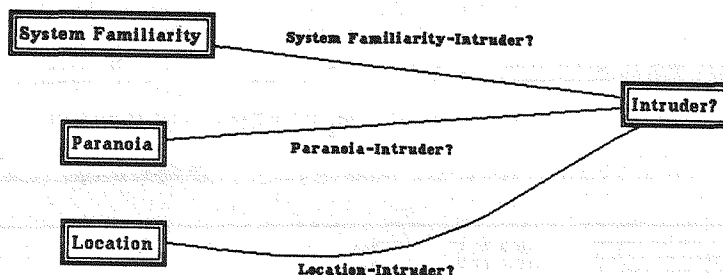


Figure 2: Frames and compatibility relations.

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.

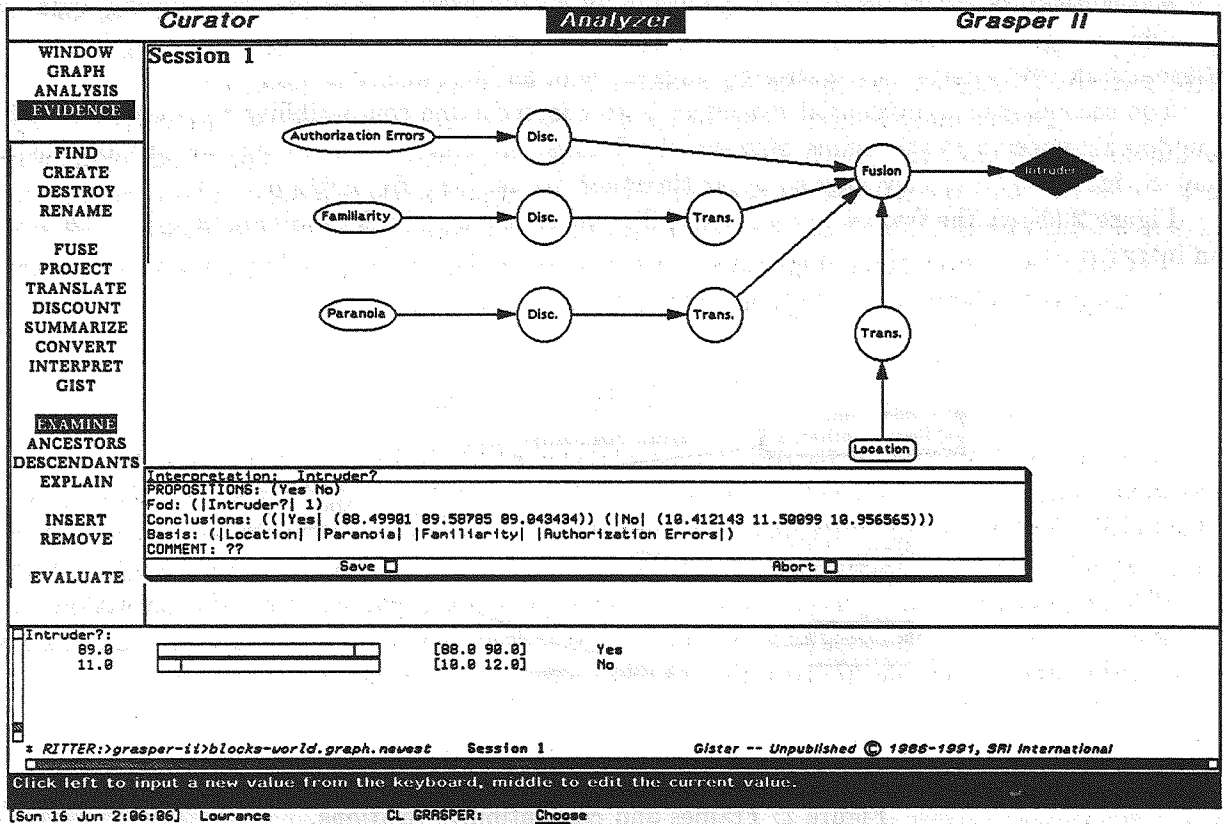


Figure 3: An intrusion analysis within Gister.

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. IDIS: 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's<sup>1</sup> 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.

---

<sup>1</sup>Open 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 Excel<sup>2</sup> and Lotus 1-2-3<sup>3</sup>). 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.

---

<sup>2</sup>Microsoft Excel is a trademark of Microsoft Corporation

<sup>3</sup>Lotus 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.  $\langle T_{a,s} \rangle_{K_s}$  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

$DT_a = \langle A, B, S, t_a, \text{duration}_a \rangle_{K_{a,ts}}$  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.

$DR = \langle B, S, t_b, \text{duration}_b, \langle B, S, t_b, \text{duration}_b \rangle_{K_{b,ts}} \rangle$

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)  $AS \rightarrow S : \langle AS, S, \langle t_s+1, K_{b,s}, \langle K_{b,s}, t_b \rangle_{K_{b,ts}} \rangle_{K_s} \rangle$

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)  $S \rightarrow B : \langle S, B, \langle t_b+1 \rangle_{K_{b,s}}, \langle K_{b,s}, t_b \rangle_{K_{b,ts}} \rangle$

To complete the process, S sends the session key to B, extracted from (4). Its encryption under  $K_{b,ts}$  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

$MT_b = \langle T_{b,s}, T_{a,s}, \langle A,B,S,t_a,duration_a \rangle \rangle_{K_s}$

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<sup>†</sup>

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

<sup>†</sup>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,

$$Var(\hat{\mu}) = \sigma^2 p \left( \frac{(n+3)}{(n+1)^2} + \frac{1}{(n-1)} - \frac{2}{n} \right) + \frac{\sigma^2}{n}$$

where  $\sigma^2$  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  $\delta$  be defined as follows:

$$\delta = \sqrt{\sum_{i=1}^N (y_i - x_i)^2 / N}$$

where  $N$  is the size of the query.  $\delta$  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 Tidsskrift*, 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] Schlorer, 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] Vemulapalli, 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 UNIX<sup>TM</sup> 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 <sup>TM</sup> 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\_dbasa") 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.

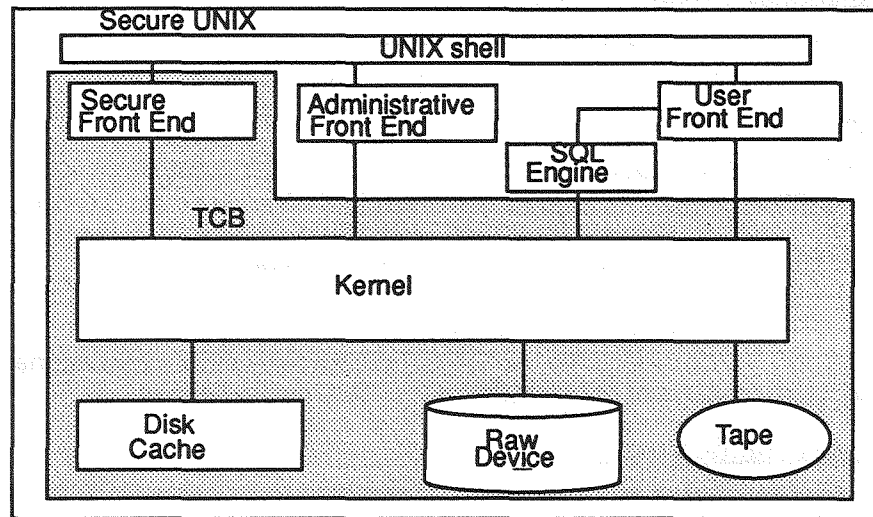


Figure 1 INFORMIX-OnLine/Secure System Architecture

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\_dbasa.

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.

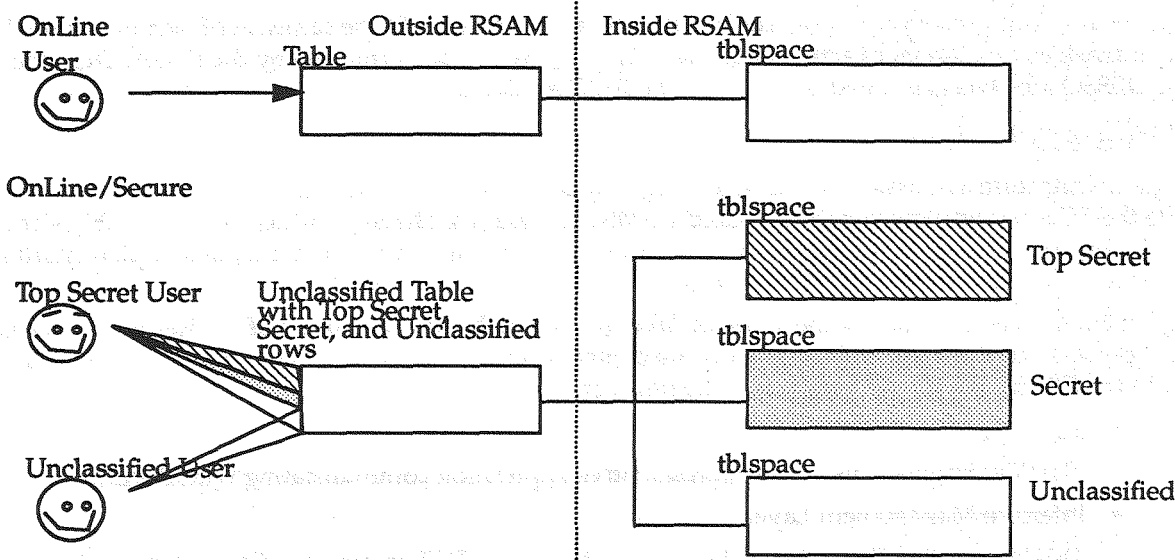


Figure 2 Tables and Table Abstractions in OnLine and INFORMIX-OnLine/Secure

### 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.

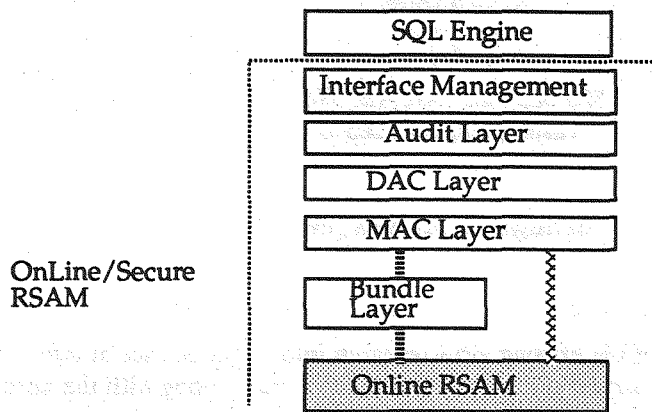


Figure 3 Interface Call Handling in INFORMIX-OnLine/Secure

### 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 integrity<sup>1</sup>. 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.**

$\text{checksum}(\text{file}) \Rightarrow \text{integrity}(\text{file})$   
 $\neg \text{checksum}(\text{file}) \Rightarrow \neg \text{integrity}(\text{file})$

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 factor<sup>2</sup> 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.

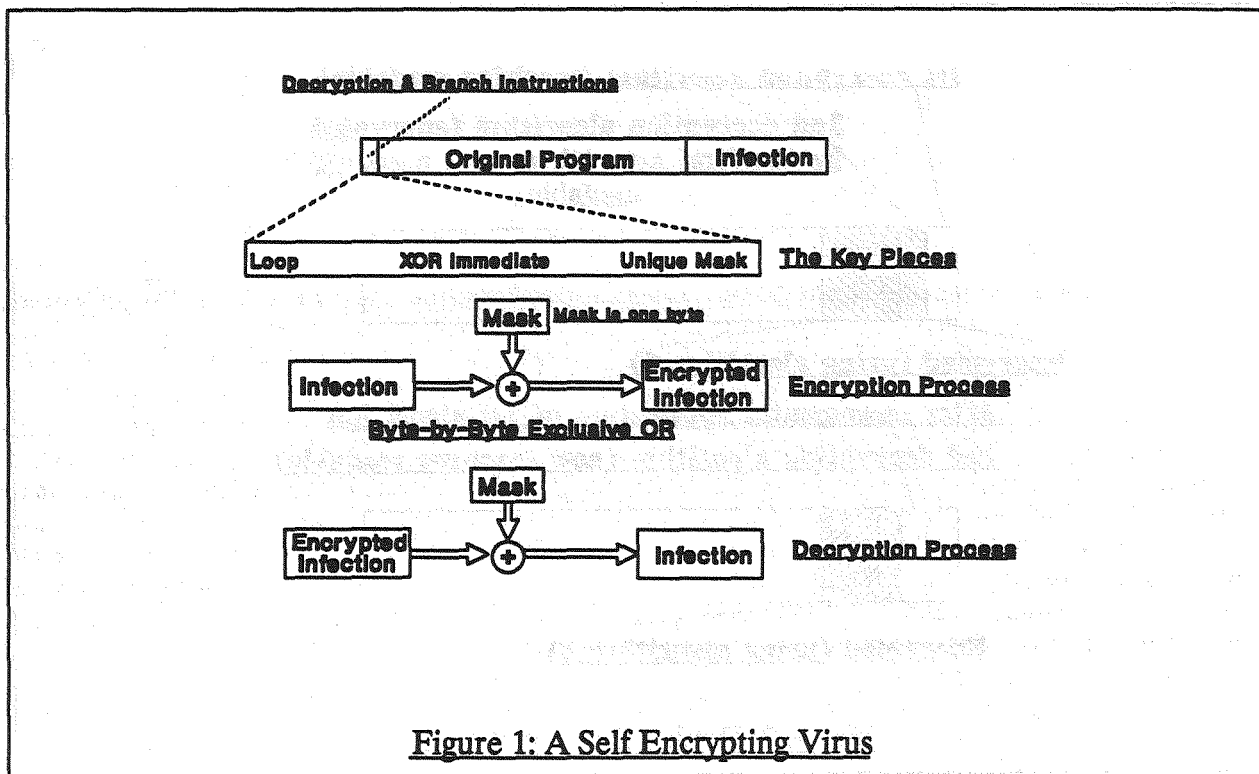
$\text{executable}(\text{file}) \wedge \text{disassemble}(\text{file}) \Rightarrow \text{assembly}(\text{file})$

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 an assembly listing. Today, many debug utilities include a 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.**

$\text{encrypted}(\text{file}) \wedge \text{applied\_to}(\text{first\_instruction}, \text{file}) \wedge \text{disassemble}(\text{file}) \Rightarrow \text{assembly}(\text{file})$

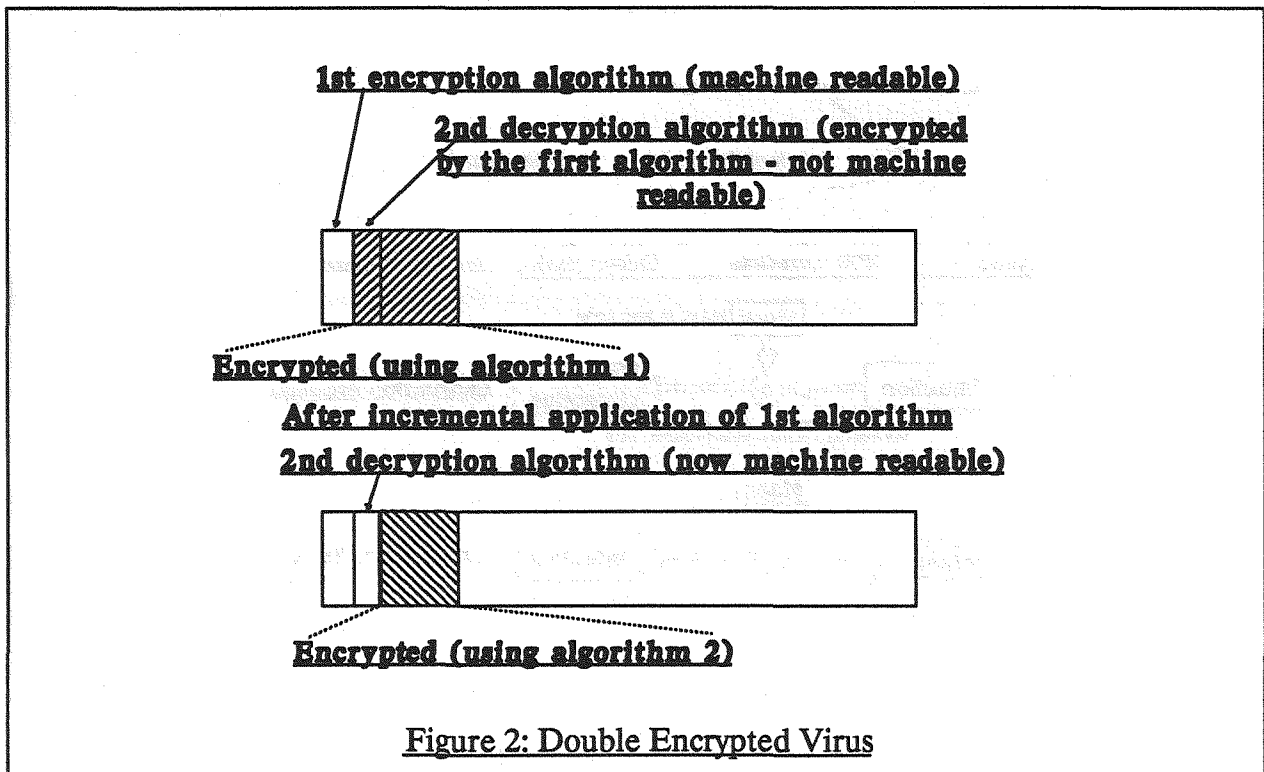
The function performed in Rule 5 decrypts the cipher thereby restoring the code to a machine readable format. The resulting machine readable code can then be disassembled by applying Rule 6.



**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.



**Rule 10: If a binary difference from DIFF disassembles, the likelihood of a random error is low.**

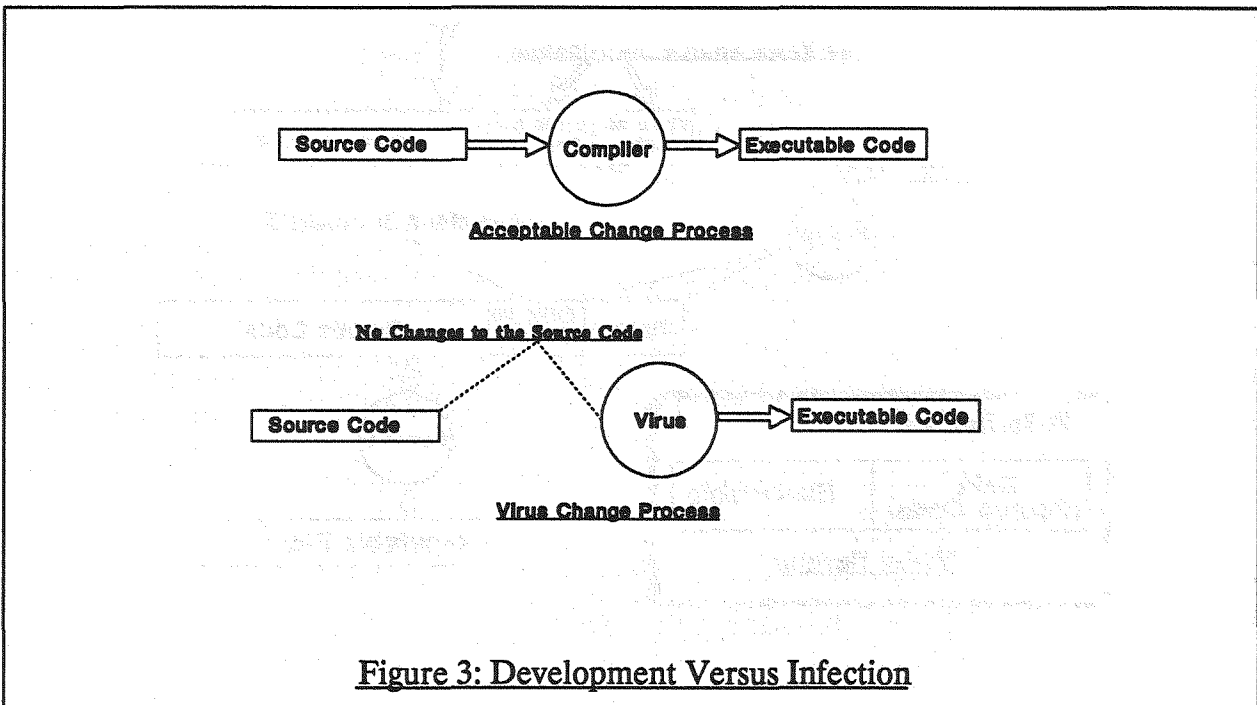
$\text{assemble}(\text{file}) \Rightarrow \text{file}(\text{executable})$

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

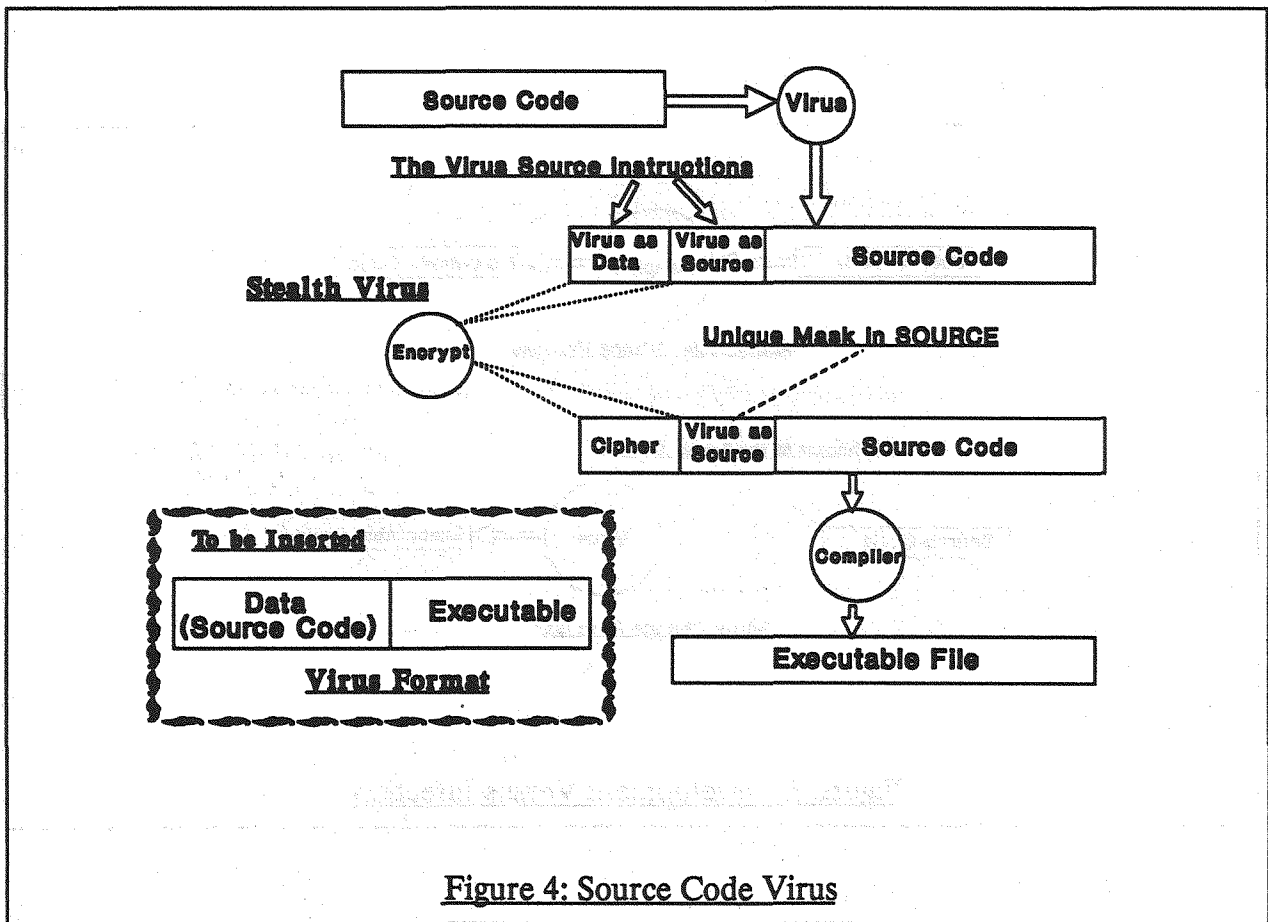


Figure 4: Source Code Virus

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.

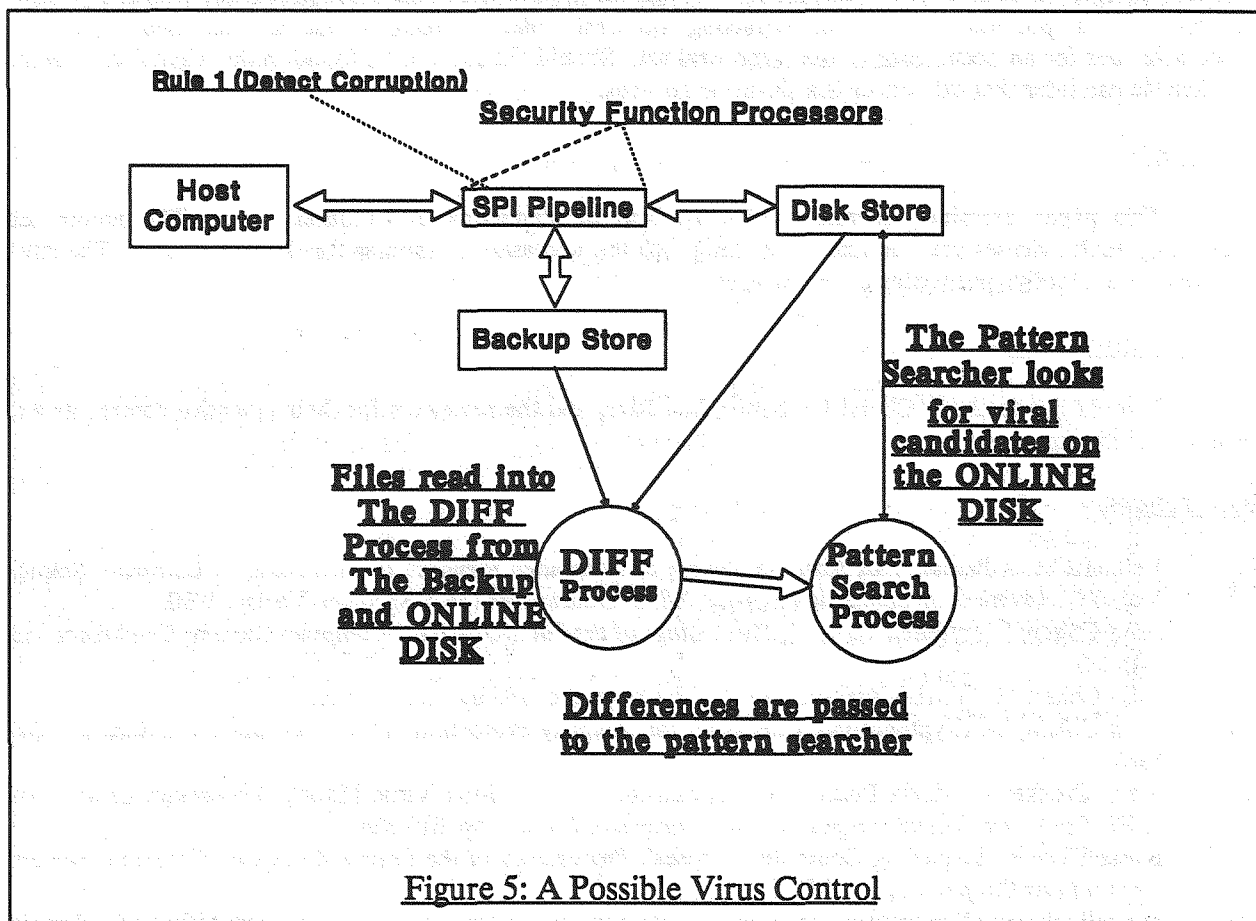


Figure 5: A Possible Virus Control

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.

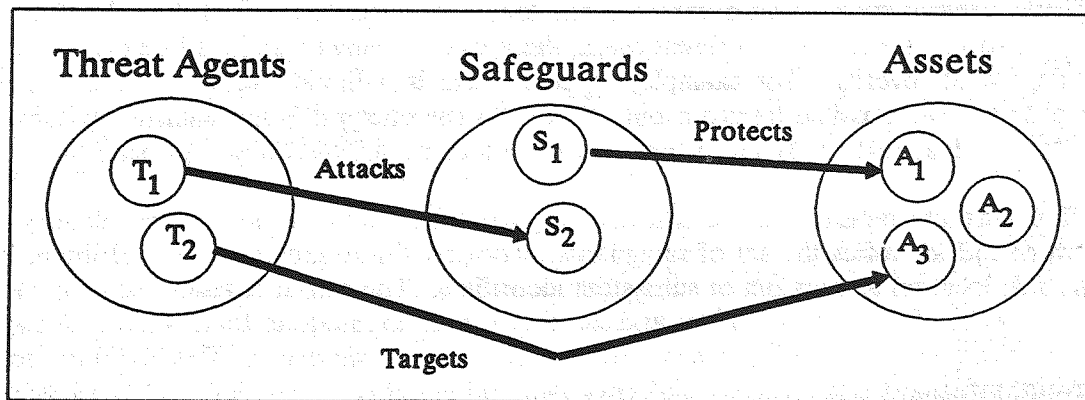


Figure 1: Sample Instance of the Threat Model

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  $O_i(\text{id})$  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.

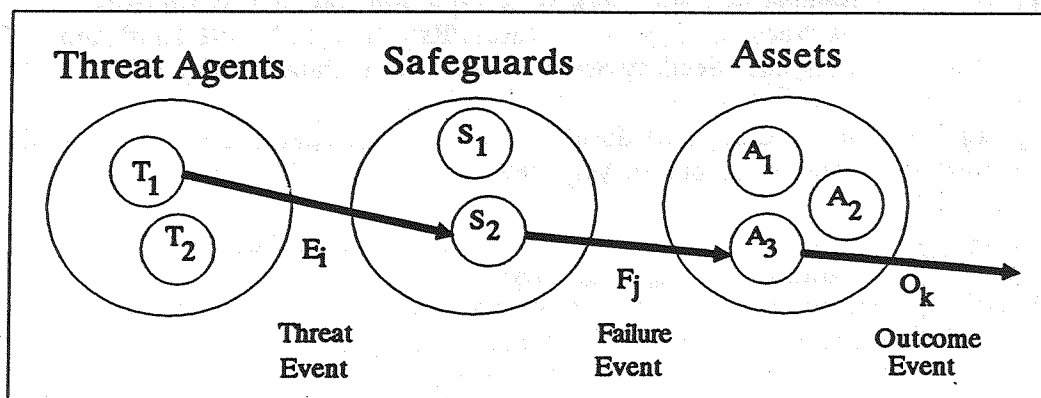


Figure 2: Sample Instance of the Risk Model

## 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  $E_i$  causes  $F_j$  which then causes outcome event  $O_k$ . The contribution to total risk by this scenario is given by the following equation:

$$X = E_i(\text{freq}) \times F_j(\text{prob}) \times O_k(\text{prob}) \times O_k(\text{sev})$$

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  $P \Rightarrow S$ .<sup>1</sup> 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

---

<sup>1</sup> 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  $P_x^e$ , and the new antecedent is given by  $P_x^e \ \& \ M$ .<sup>2</sup> For the example, the desired new invariant  $G \ \& \ B \ \& \ C \ \& \ D \Rightarrow S$  is given by  $(A \ \& \ B \ \& \ C \ \& \ D) \hat{A} \Rightarrow S$ . 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  $x_i$ .

<sup>2</sup> The notation  $P_x^e$  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:  $F \& \neg(F^{x_i})$ , or (b) it is initially false and changing the value of  $x_i$  makes it true:  $\neg F \& (F^{x_i})$ . 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  $P_x^e$ , i.e.  $P \equiv P_x^e$ .

**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  $dP_x^e$ , is  $P \oplus P_x^e$ .

**Lemma 1.**  $dP_x^z = 0$  iff  $P$  is independent of the value of  $x$ .

*Proof.*

Assume ("if" direction)  $P \equiv P_x^z$  {definition of independence}

Then  $(P \Leftrightarrow P_x^z)$

$\equiv (P \oplus P_x^z = 0)$  {definition of  $\oplus$ }

$\equiv (P \oplus P_x^z = 0)$

Assume  $(P \oplus P_x^z = 0)$  ("only if" direction)

$\equiv (\neg(P \Leftrightarrow P_x^z) = 0)$  {definition of  $\oplus$ }

$\equiv (P \Leftrightarrow P_x^z)$

(End of proof.)

**Definition 4. Unconditional Dependence:**  $P$  is unconditionally dependent on  $x/e$  if  $P$  has the opposite truth value of  $P_x^z$ , i.e.  $(P \Leftrightarrow \neg P_x^z) \& (P_x^z \Leftrightarrow \neg P)$

**Lemma 2.**  $dP_x^z = 1$  iff  $P$  is unconditionally dependent on the value of  $x/e$ .

*Proof.*

$(P \Leftrightarrow \neg P_x^z) \& (P_x^z \Leftrightarrow \neg P)$

$\equiv \neg(P \equiv P_x^z)$

$\equiv (P \oplus P_x^z) = 1$

(End of proof.)

If  $dP_x^z$  is not 0 and not 1, then the resulting formula can be solved for 1 to determine the conditions under which  $P_x^z$  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

$$dF/d(F_i)_{e^j} = dF_{\neg F_i}^{F_i} \& dF_i^{x_j}$$

#### 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  $dI/dP_x^z$  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  $P_z$ , 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  $\neg P \ \& \ P_z \ \& \ \neg S \equiv P_z \ \& \ \neg S$ .

*Proof*

$$\begin{aligned} dI/dP_z &= (P \oplus P_z) \ \& \ \neg S && \{\text{Definition 3 and 5}\} \\ &\equiv (P \ \& \ \neg P_z \ | \ \neg P \ \& \ P_z) \ \& \ \neg S && \{\text{Definition of } \oplus\} \\ &\equiv \neg P \ \& \ P_z \ \& \ \neg S && \{\text{assumed: } (P \Rightarrow S) \equiv \neg(P \ \& \ \neg S)\} \\ &\equiv P_z \ \& \ \neg S && \{(P \Rightarrow S) \Rightarrow (\neg P \ \& \ \neg S \equiv \neg S)\} \end{aligned}$$

*(End of proof.)*

The form  $\neg P \ \& \ P_z \ \& \ \neg S$  may be more useful if we expect the change  $x/e$  to maintain the invariant, because showing either  $P_z \ \& \ \neg P = 0$ , or  $P_z \ \& \ \neg S = 0$  is sufficient to show that  $P_z \Rightarrow S$ . If  $P_z \ \& \ \neg P$ , 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  $P_z \ \& \ \neg S = 0$ , which is equivalent to  $P_z \Rightarrow S$ .

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  $(P_z) \ \& \ M$  maintains the invariant. The first is to show directly that  $P_z \ \& \ M \Rightarrow S$ . 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  $P_z \ \& \ M$  implies the negation of the partial predicate difference  $dI/dP_z$  i.e.:  $P_z \ \& \ M \Rightarrow \neg [dI/dP_z]$ . Proving this is equivalent to proving  $P_z \ \& \ M \Rightarrow S$  directly. This result is proved formally below.

**Theorem 2.**  $[(P \Rightarrow S) \ \& \ [M \Rightarrow \neg [dI/dP_z]]] \Leftrightarrow [(P \Rightarrow S) \ \& \ (P_z \ \& \ M \Rightarrow S)]$

*Proof*

$$\begin{aligned} &(P \Rightarrow S) \ \& \ [M \Rightarrow \neg [dI/dP_z]] \\ &\equiv (P \Rightarrow S) \ \& \ (M \Rightarrow \neg (P_z \ \& \ \neg S)) && \{\text{Theorem 1}\} \\ &\equiv (P \Rightarrow S) \ \& \ (M \Rightarrow (\neg P_z \ | \ S)) \\ &\equiv (P \Rightarrow S) \ \& \ (M \Rightarrow P_z \Rightarrow S) \\ &\equiv (P \Rightarrow S) \ \& \ (M \ \& \ P_z \Rightarrow S) \end{aligned}$$

*(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

```
chpasswd(input_val)
{
  /* if security officer, then change password to input value */
  if (s_auth) pw := input_val
}
```

This *chpasswd* function is modeled by

$(s\_auth \Rightarrow pw' = \text{input\_val}) \ \& \ (\neg s\_auth \Rightarrow pw' = pw)$

A proof is done to show that the state invariants plus the effect of the *chpasswd* function ensure the state transition invariant (the function must also maintain the invariants, but this is not shown for conciseness).

$(u\_auth \Rightarrow t\_auth) \ \&$

$(t\_auth \Rightarrow pw \neq 0) \ \&$

$(s\_auth \Rightarrow pw' = \text{input\_val}) \ \&$

$(\neg s\_auth \Rightarrow pw' = pw)$

$\Rightarrow (pw' \neq 0 \ \& \ pw = 0 \Rightarrow s\_auth)$

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 *chpasswd* function specification then becomes:

$((s\_auth \mid u\_auth) \Rightarrow pw' = \text{input\_val}) \ \& \ (\neg(s\_auth \mid u\_auth) \Rightarrow pw' = pw)$

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

$(u\_auth \Rightarrow t\_auth) \ \&$   
 $(t\_auth \Rightarrow pw \neq 0) \ \&$   
 $(u\_auth \mid s\_auth \Rightarrow pw' = inval) \ \&$   
 $(\neg(u\_auth \mid s\_auth) \Rightarrow pw' = pw) \ \&$   
 $\neg(pw' \neq 0 \ \& \ pw = 0 \Rightarrow s\_auth) \equiv 0$

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.,  $T_1 \Rightarrow S \ \& \ T_2 \Rightarrow S \ \& \ \dots \ \& \ T_n \Rightarrow S$ . 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  $d(T \Rightarrow S)_e^x$ . 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  $dP_e^x \Rightarrow dP_f^z$  ( $x$  may equal  $z$  and  $e$  may equal  $f$ ). Also define  $x/e < z/f$  if  $dP_e^x \Rightarrow dP_f^z$  but not  $dP_f^z \Rightarrow dP_e^x$ . 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  $dP_e^x = 0$ , as shown in Lemma 1.

If  $dP_e^x \Rightarrow dP_f^z$  then it could be said that  $P_e^x$  differs less from  $P$  than does  $P_f^z$ . 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  $P_e^x$  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  $d(a \mid b)_e^a$  is  $c \ \& \ \neg b \mid b \ \& \ \neg c$ , and  $d(a \mid b)_e^a|_c$  is  $\neg a \ \& \ \neg b \ \& \ c$ . So  $d(a \mid b)_e^a|_c \Rightarrow d(a \mid b)_e^a$ , 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  $(P_1)_t^z \ \& \ \neg S_1 \mid (P_2)_t^z \ \& \ \neg S_2 \mid (P_n)_t^z \ \& \ \neg S_n$ , 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:

$$(\neg(w | t | u) \& (t \& d \& p | w)) \equiv 0$$

and

$$(\neg w \& (t \& d \& p | w) \& \neg t) \equiv 0.$$

Note that since

$$(\neg(w | t | u) \& (t \& d \& p | w)) \equiv 0$$

it is not necessary to compute

$$\neg(w | t | u) \& (t \& d \& p | w) \& \neg(p \& d \& l).$$

## 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 UNIX<sup>1</sup> 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 key-space 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 key-space 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.<sup>2</sup> Unfortunately, users often select passwords that do not exploit the large key-space 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 weak<sup>3</sup> passwords has led to a number of system break-ins,

---

<sup>1</sup> UNIX is a trademark of Unix System Laboratories, Inc.

<sup>2</sup> 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 key-space would require over 27,480 years to complete.

<sup>3</sup> 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 key-space. 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 Project<sup>4</sup> 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,  $\phi$ , equal

---

<sup>4</sup> Obvious 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  $\phi$ , 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 keyspace; words in monocase, 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.<sup>1</sup>

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 unavailability 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

---

<sup>1</sup> 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,<sup>2</sup> 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.<sup>3</sup>

## 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

---

<sup>2</sup> 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.

<sup>3</sup> 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 deprecations 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)  $MARK-ADI$ .<sup>4</sup> 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- $\underline{e}$ -access mode, there would be no necessary relationship between  $MARK-T$  and  $MARK-AI$ . On the other hand, if invocation includes  $\underline{r}$ -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  $MARK-VDI$ .<sup>5</sup> If a ULM instance allows a pure- $\underline{e}$  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  $t = MARK-T$ .<sup>6</sup>

---

<sup>4</sup> This solution of the binding problem is derived from the solutions found in [LEE88], [KARG88], and [SHOC87].

<sup>5</sup> If the altering access mode implies a viewing capability, then  $MARK-T$  would have to imply  $MARK-ADI$ .

<sup>6</sup>  $MARK-ADI$  becomes  $MARK-T \wedge MARK-VDI$ .

Authorized invokers have the lattice-value  $i = \text{MARK-AI}$  "added" to their lattice-value.<sup>7</sup> 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 T1 from invoking T2 using either discretionary or non-discretionary features of a ULM. The discretionary approach would be to put the invoker of T1 onto a NACL for T2 as part of the execution of T1. That is, the action of subject  $S$  invoking T1 would cause T1, running as a subject, to set the NACL of T2 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 *MARK-EXCLUDE-T<sub>i</sub>* that is added to the invoker's current lattice-value on invocation (that is, the invoker of T1 has *MARK-EXCLUDE-T2* added to its current lattice-value) and the logic for invocation of T2 is altered to be "lattice-value(subject) includes *MARK-T2* and does not include *MARK-EXCLUDE-T2*".<sup>8</sup>

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

---

<sup>7</sup> The lattice-value  $i$  is added in the sense of having it ANDed onto whatever other lattice-value is already existing.

<sup>8</sup> 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.<sup>9</sup> 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.<sup>10</sup> 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.

---

<sup>9</sup> 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.

<sup>10</sup> 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 Noforn** and **SECRET EI Noforn**, respectively. A Lower Volgan national cleared to **SECRET** is allowed to read information designated either **SECRET EILV Only** or **SECRET LV Noforn**; analogously for a Irish national cleared to **SECRET**. A Lower Volgan is allowed to create **SECRET LV Noforn** 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:

$p$	→ "EILV Only"	$p, q$	→ "LV Only"
$q$	→ "LVYU Only"	$q, r$	→ "YU Only"
$r$	→ "EIYU Only"	$r, p$	→ "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.<sup>11</sup> 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.<sup>12</sup> Any analyst may request access to any restricted data item. Requesting access to restricted data items is implemented as the invocation of a bound

---

<sup>11</sup> 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.

<sup>12</sup> 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*.<sup>13</sup>

**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.<sup>14</sup> 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

---

<sup>13</sup> 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.

<sup>14</sup> The case of individuals can of course be included by viewing each individual as a group of one.

form *MARK-P* or *MARK-Q*).<sup>15</sup> 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.<sup>16</sup> 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

---

<sup>15</sup> 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.

<sup>16</sup> 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.<sup>17</sup>

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.<sup>18</sup> [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-

---

<sup>17</sup> Clearly roll-back can be used to extricate the system from an anomalous state when there is a difference of opinion about the release.

<sup>18</sup> 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

$$\bigwedge_{a \in A} s(a), \quad \text{where } s(a) \text{ is either } a \text{ or } \neg a \text{ for } a \text{ in the alphabet } A.$$

[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's<sup>19</sup> 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

---

<sup>19</sup> "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  $2^{2^{|A|}}$ .<sup>20</sup> [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.

---

<sup>20</sup> 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:

- |                 |  |
|-----------------|--|
| <b>Window</b>   | A <i>window</i> is an abstraction of a displayable region on the workstation screen.   |
| <b>Pixmap</b>   | 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> . |
| <b>Property</b> | 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.  |
| <b>Atom</b>     | 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:

- |                          |   |
|--------------------------|---|
| <b>Sensitivity label</b> | A classification and compartments set that is used as the basis for mandatory access control decisions.   |
| <b>Information Label</b> | 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. |
| <b>User ID</b>           | An integer which uniquely identifies a user. The server may share the screen with multiple users.   |
| <b>Privileges</b>        | 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.

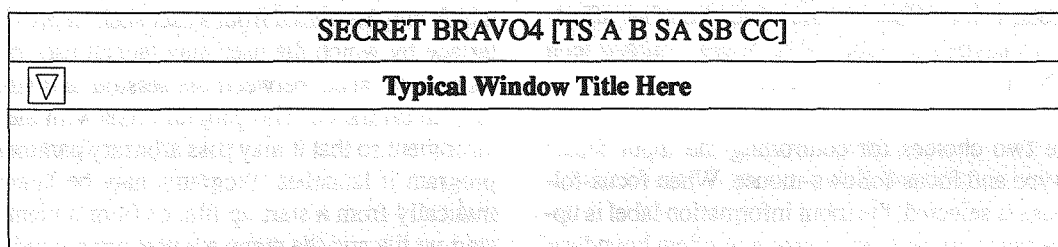
**Logintool** also starts a user thread of execution for unprivileged processes to be initiated with the user's environment. The processes started from this thread are started with the same sensitivity label as the users's home directory. This label is called the *session low* label.

#### 4.2 The Window Manager

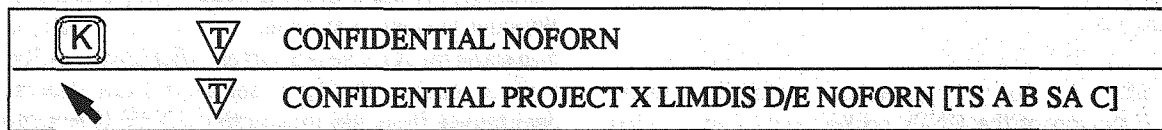
The window manager is based on **olwm**, the OPEN LOOK Window Manager in Open Windows. In addition to the functions normally performed by a window manager, **olwm** displays a CMW label (a combined information and sensitivity label) for each top-level window, which accurately reflects the data in the 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**



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] LINX - 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. Schauer, 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)<sup>1</sup> 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 73<sup>2</sup> 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)<sup>3</sup>. 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 Model<sup>4</sup> 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
<b>Confidentiality</b>	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.
<b>Control examples</b>	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.
<b>Loss examples</b>	Disclosure, inference, espionage	Piracy, trade secret loss, espionage	Piracy, espionage	Espionage, theft, reverse engineering	Violation of privacy
<b>Authenticity</b>	Genuine, accepted by conformity to fact, valid	Genuine, unquestioned origin	Genuine, unquestioned origin	Genuine, unquestioned origin	Unquestioned identity
<b>Control examples</b>	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
<b>Loss examples</b>	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
<b>Integrity</b>	Unimpaired, complete, whole	Unimpaired, all present, ordered	Unimpaired, all present, ordered	Unimpaired, all present	Honest, forthright, loyal, ethical
<b>Control examples</b>	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
<b>Loss examples</b>	Larceny, taking, changing, fraud, concatenation	Theft, fraud, change, concatenation	Theft, fraud, change, concatenation	Sabotage, parts theft	Negligence, injury, ethical violations
<b>Utility</b>	Fit, created for use	Fit, executable	Fit, executable	Workable, assured of action	In good mental and physical health, motivated, assured of action
<b>Control examples</b>	Testing, backup, recovery plans	Testing, maintenance	Testing, maintenance	Testing maintenance	Medical exams, environment protection, safety
<b>Loss examples</b>	Change, sabotage, converted	Sabotage	Sabotage	Change, failure, sabotage	Accidents, injury, injunction
<b>Availability</b>	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
<b>Control examples</b>	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
<b>Loss examples</b>	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 model<sup>5</sup> 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 Model<sup>6</sup> 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 Criteria<sup>7</sup> (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<sup>8, 9, 10</sup>. 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 mystery<sup>11</sup>. 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 face<sup>12</sup>.

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<sup>13,14</sup>.

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	10 9 29	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]<sup>11</sup>. 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.fidonet.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].

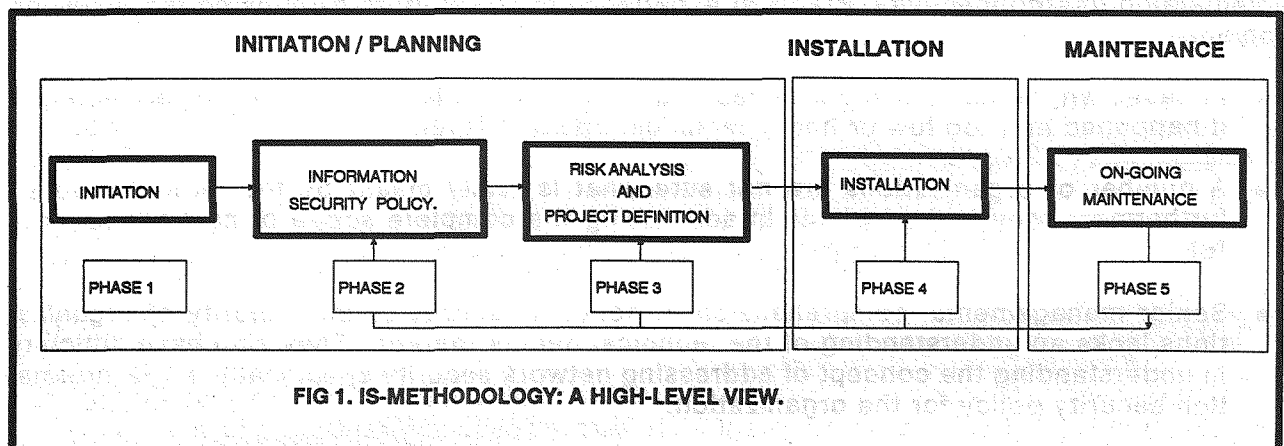


FIG 1. IS-METHODOLOGY: A HIGH-LEVEL VIEW.

**(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

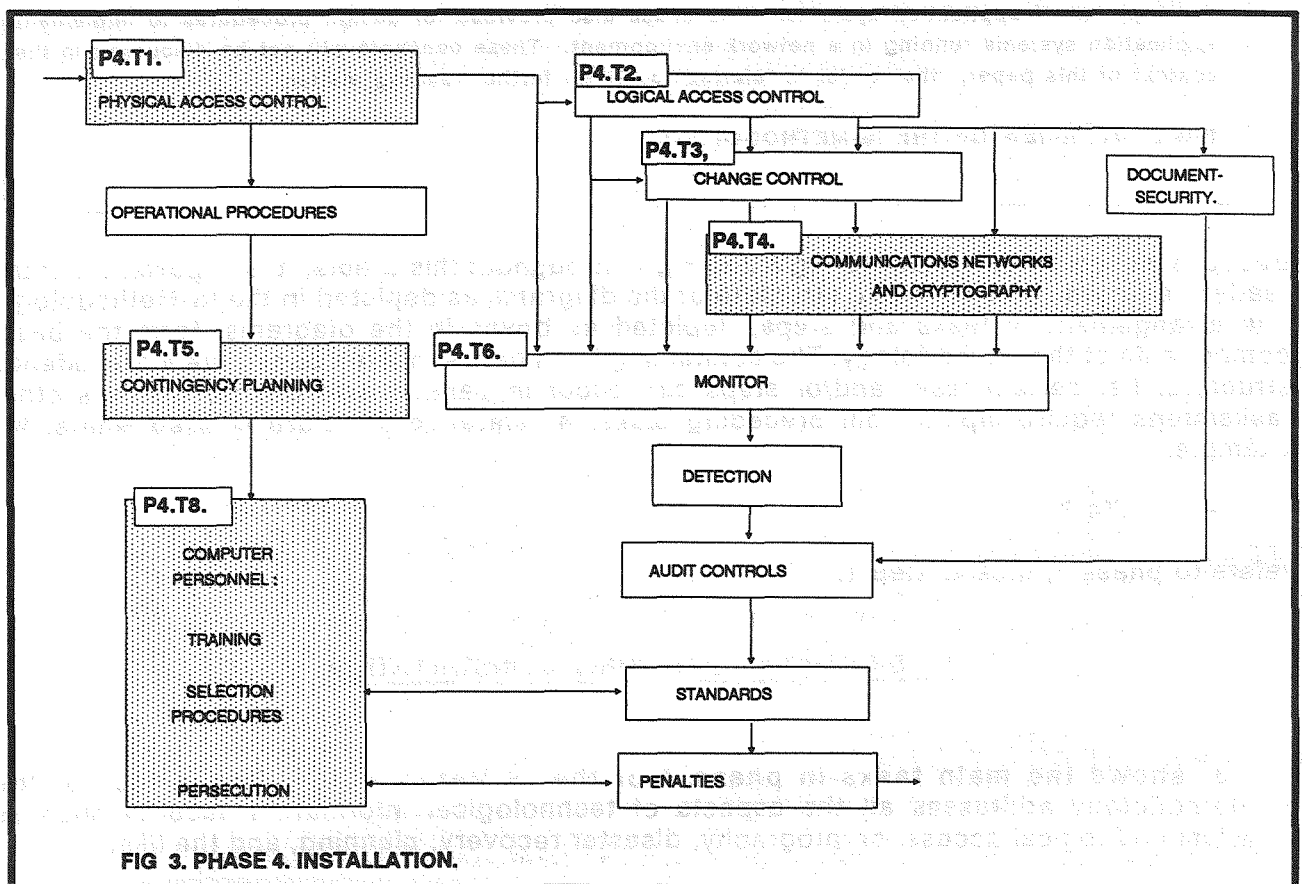
refers to phase 4, task 4, step 1.

## 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.T8):** 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.



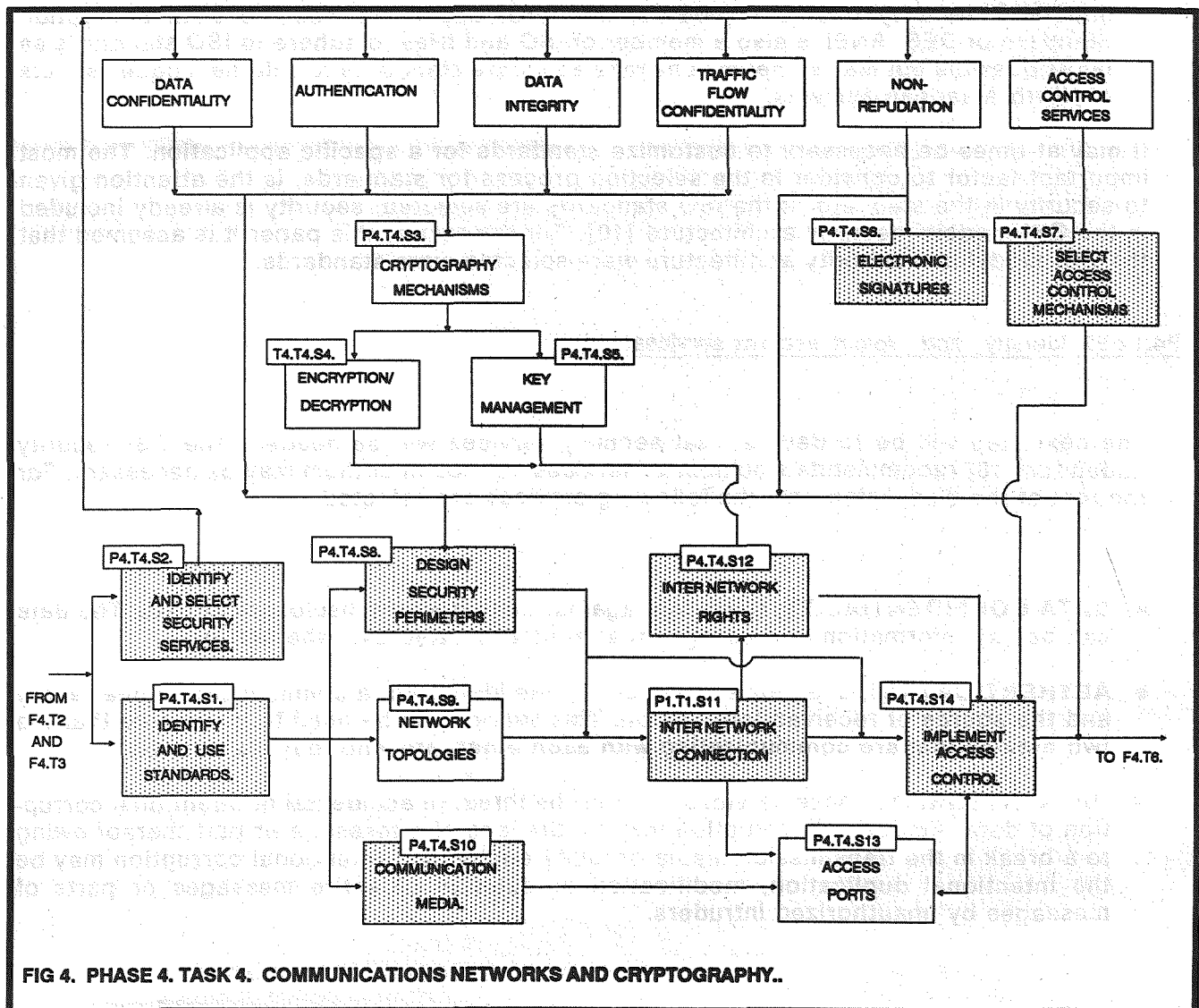
### 3. ACHIEVING NETWORK SECURITY - IS-METHODOLOGY : PHASE 4, TASK 4.

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:

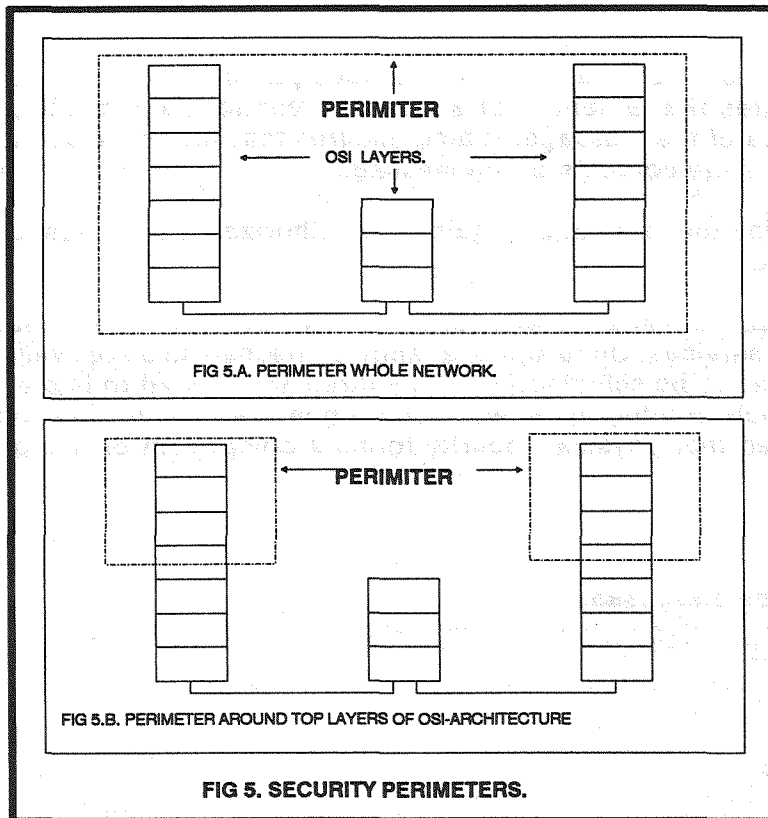
- It must be totally unforgeable.
- It must be authenticatable. This means that it must be possible to prove that the sender made the signature and that he is the owner of the signature.

Electronic signatures are a very interesting field but rather more study is needed before electronic signatures will be accepted as fail-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 objects "Security profile". The software and hardware components making up the access path is called the "Baggage". If a subject tries to access a 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 needs 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, Pabrai 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. 2987.
- [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 number<sup>1</sup>.

#### 4.2 Authentication and Access Control

In any distributed system, if the point at which a security subject<sup>2</sup> 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)<sup>3</sup>. 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

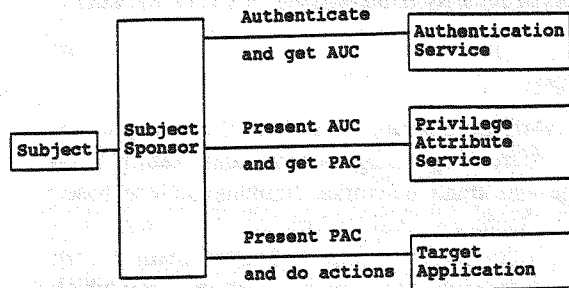


Figure 1. Functional Components

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 Ticket<sup>4</sup>. 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 key<sup>5</sup>. 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 it<sup>6</sup>.

#### 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 keys<sup>7</sup>. 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

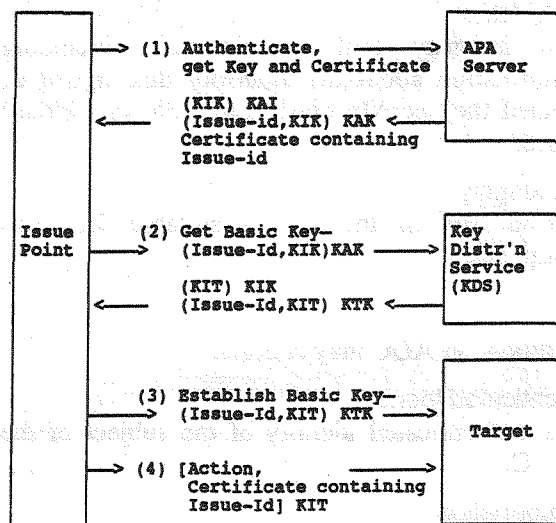


Figure 2. Symmetric Key Protocol for Non-proxiable Certificates

(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 protocol<sup>8</sup>.

#### 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.

---

<sup>8</sup> 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 preferred<sup>9</sup>. 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

<sup>9</sup> 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.

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. below<sup>10</sup>.

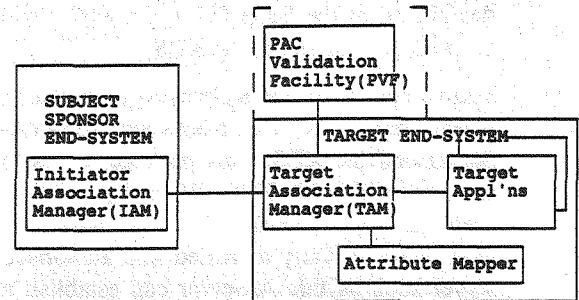


Figure 3. Target Machine Structure

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.

<sup>10</sup> 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 connection<sup>11</sup>, 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 KDS<sup>12</sup>.

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.

$t_I$	$q_W$	$q_R$	$t_I$	$q_W$	$q_R$	$t_I$	$q_W$	$q_R$
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.

$k$	$t_I$	$G_R$	$G_W$
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,  $s_3 : (U - \{x\}) \cup V \rightarrow \mathcal{N}$ , 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.

$m$	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.

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_TTM	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.

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 anomoly 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 precedural 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 CONFERENCE<sup>1</sup>**

**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

---

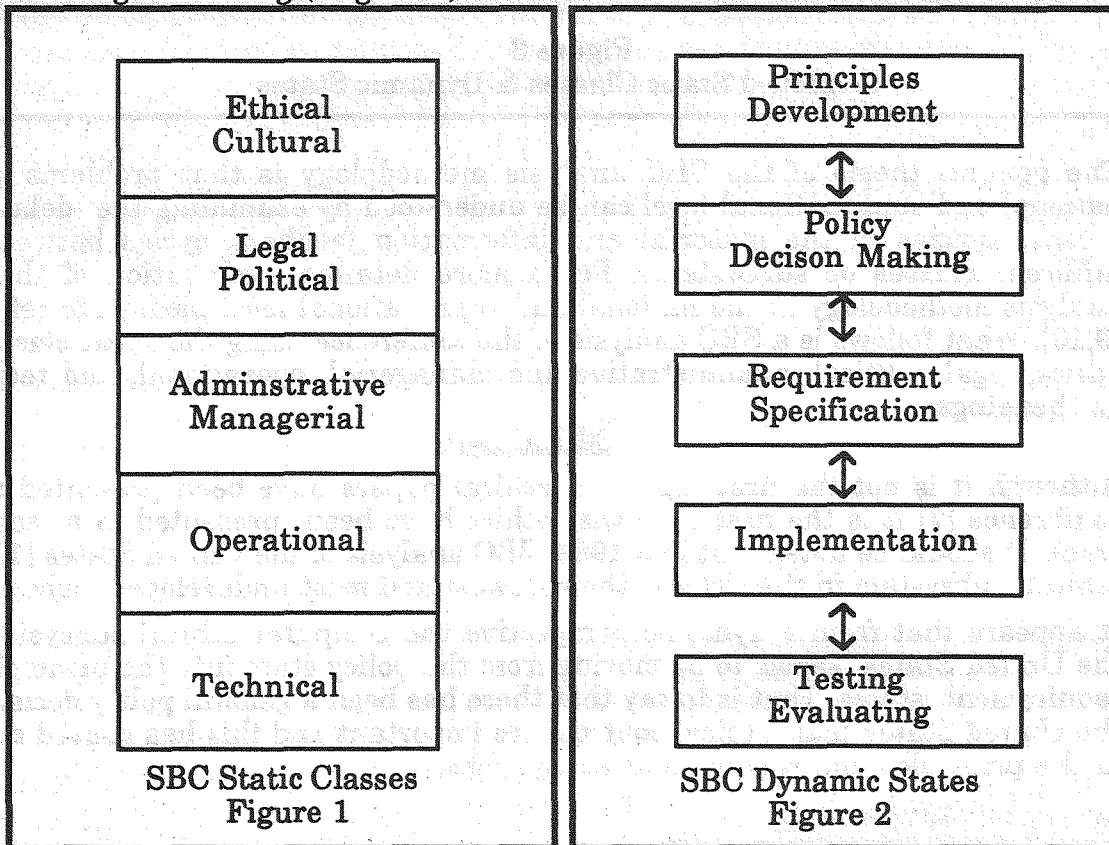
<sup>1</sup> A version of this paper was originally published in Computers & Security Volume 10 No 3 1991 and is being published here with the full permission of the publishers, Elsevier Advanced Technology, Mayfield House, 256 Banbury Road, Oxford, OX2 7DH, UK. This paper has been funded by the Swedish 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).

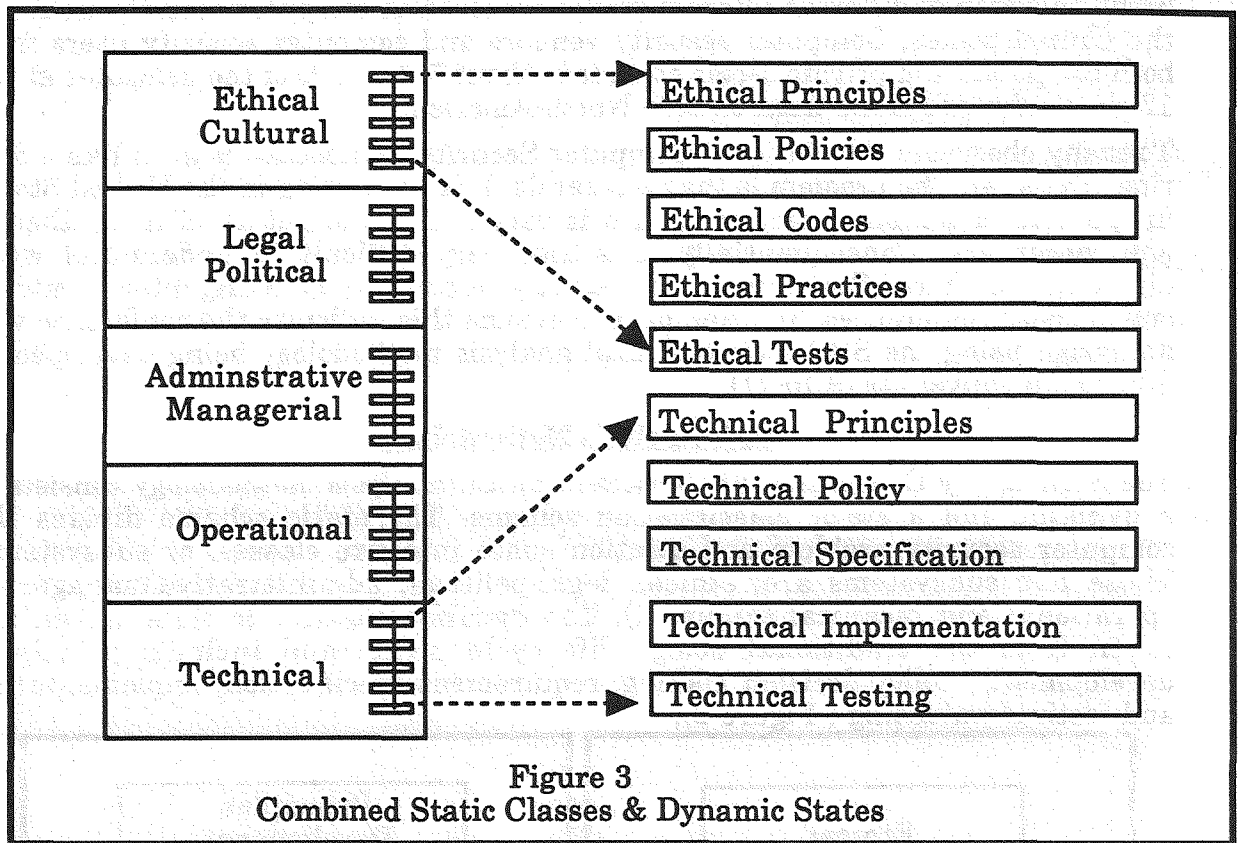


Figure 3  
Combined Static Classes & Dynamic States

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.

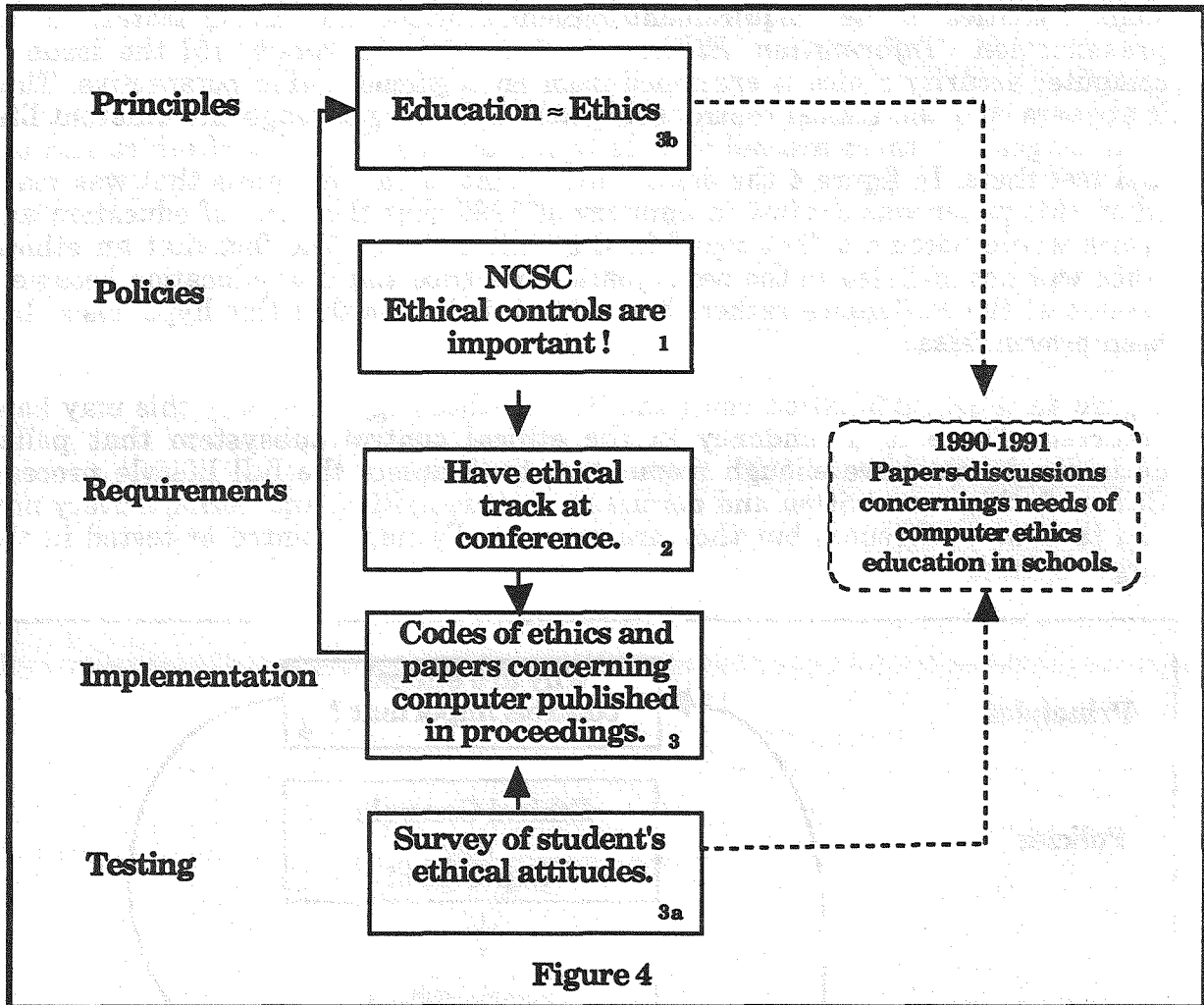
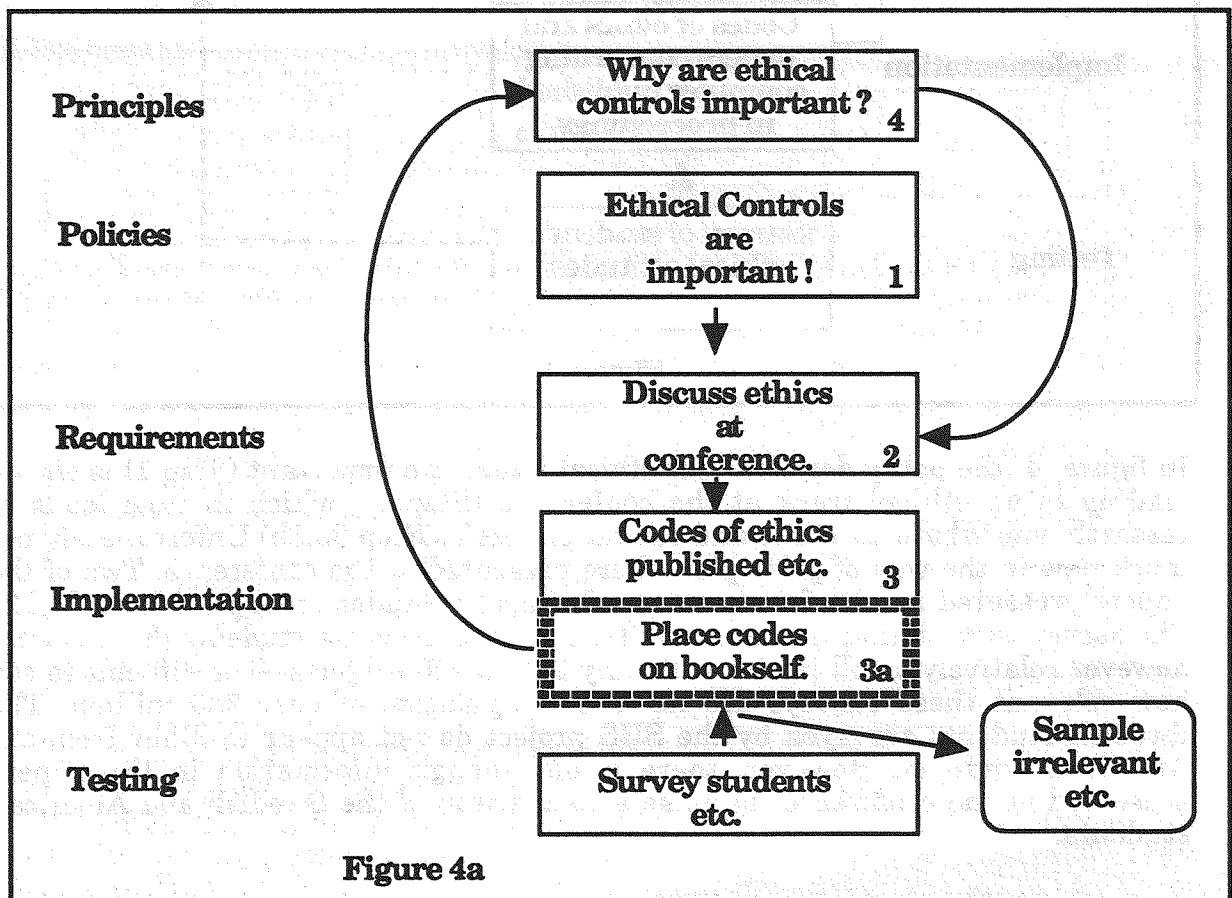


Figure 4

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.



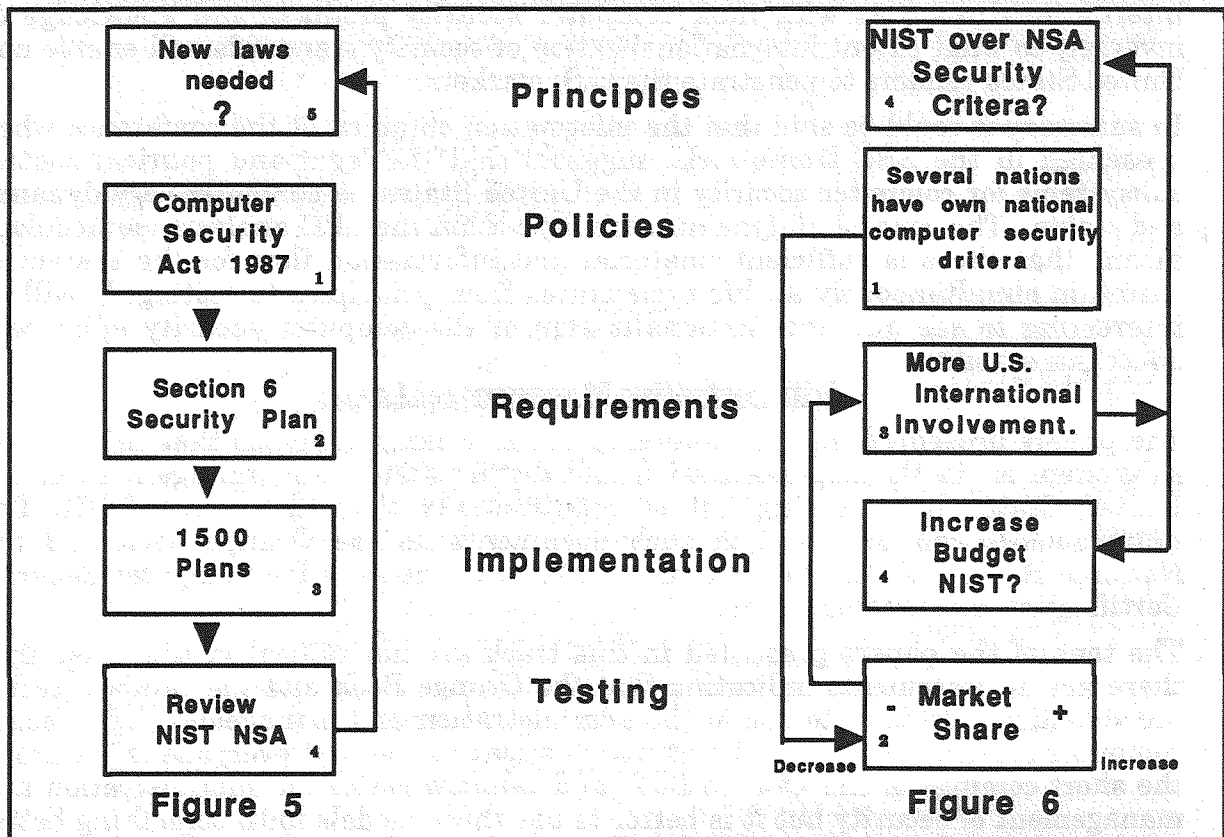
At the 12 th. annual conference the flag had been raised in the United States on the importance of ethical control to the computer security problem. Only time will tell if it has been raised high enough for the individuals in the computer security community to take notices and bring the ethical control 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 subsytem 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.

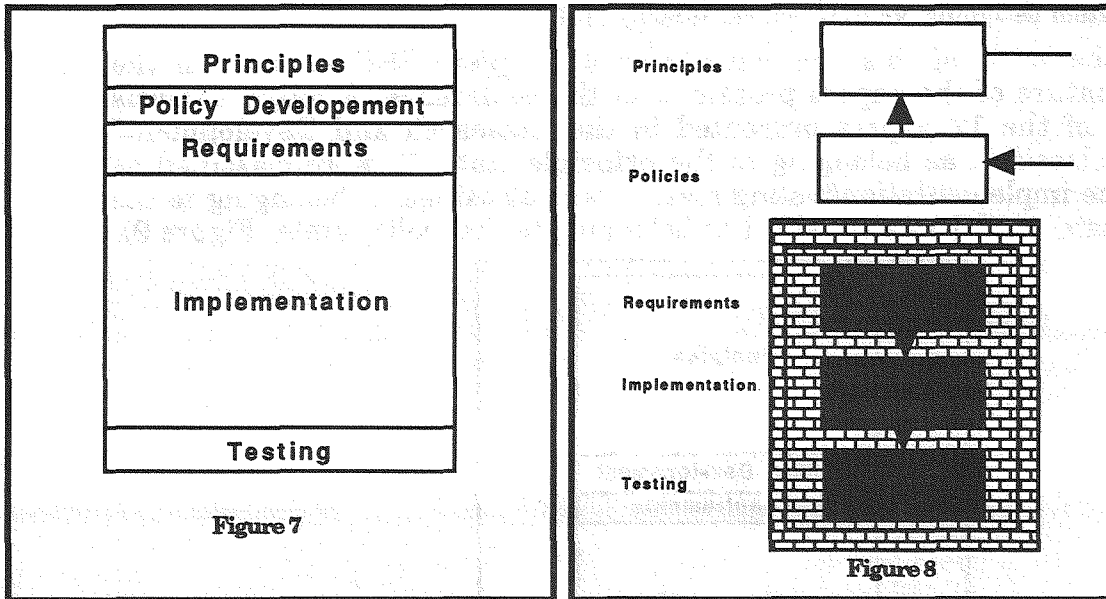
In summary it could be said that the information obtained at the conference when presented in the SBC framework suggests that the legal and political control subsystems for computer security in the United States is currently very dynamic and stable. The concept of dynamic stability within the SBC analysis 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.



### 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].

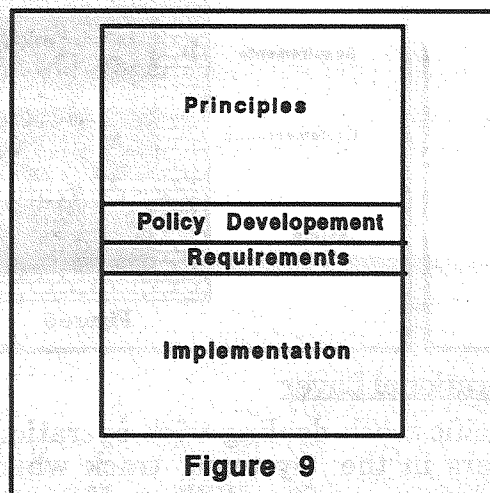
What this would indicate in a SBC framework analysis is that the principles and policies are poorly coupled to the requirement and implementation subsystems (Figure 8 above). A weak coupling between principles and policies, and requirements and implementation, means that what is written on paper at headquarter 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 Informations säkerhet 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 probably 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 additional 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

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 Entropy 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. Weiss

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 risks<sup>1</sup> (vulnerabilities) for applied security dollars.

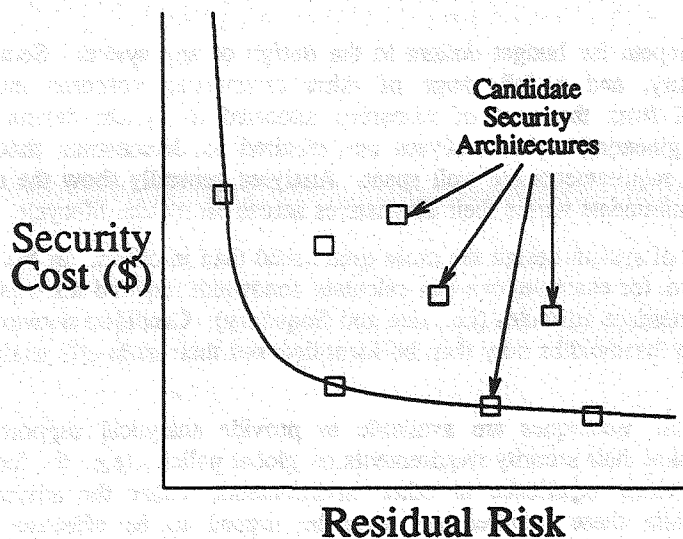


Figure 1. SSE Goal: Security Dollars Spent on Key Security Risks

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.

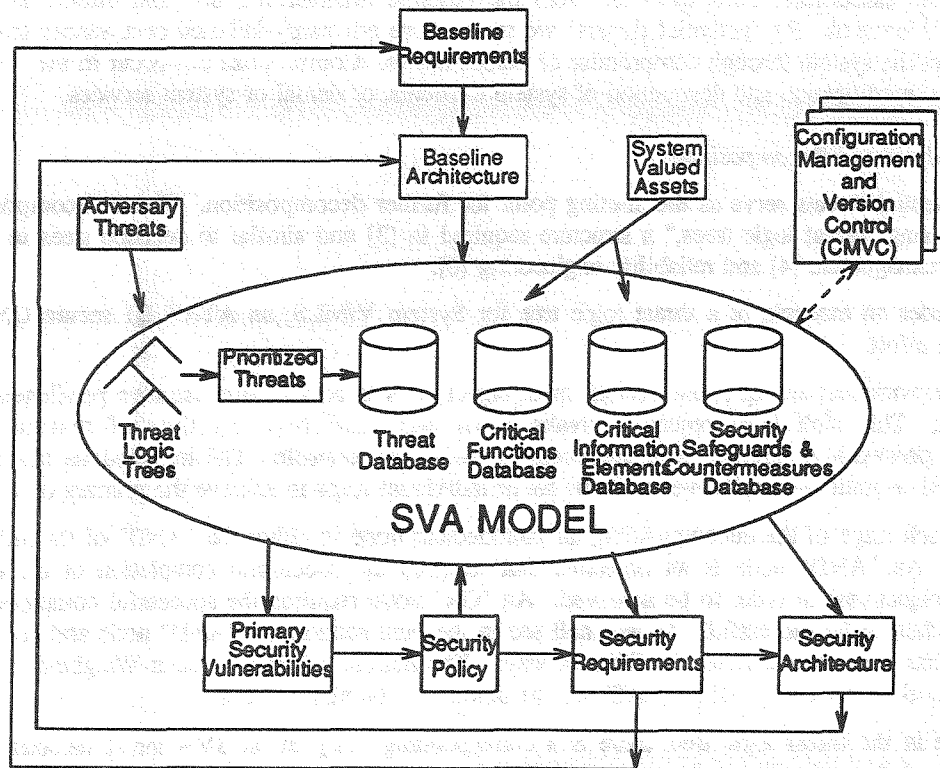


Figure 2. SVA Model

### 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)<sup>®</sup> 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<sup>®</sup> 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

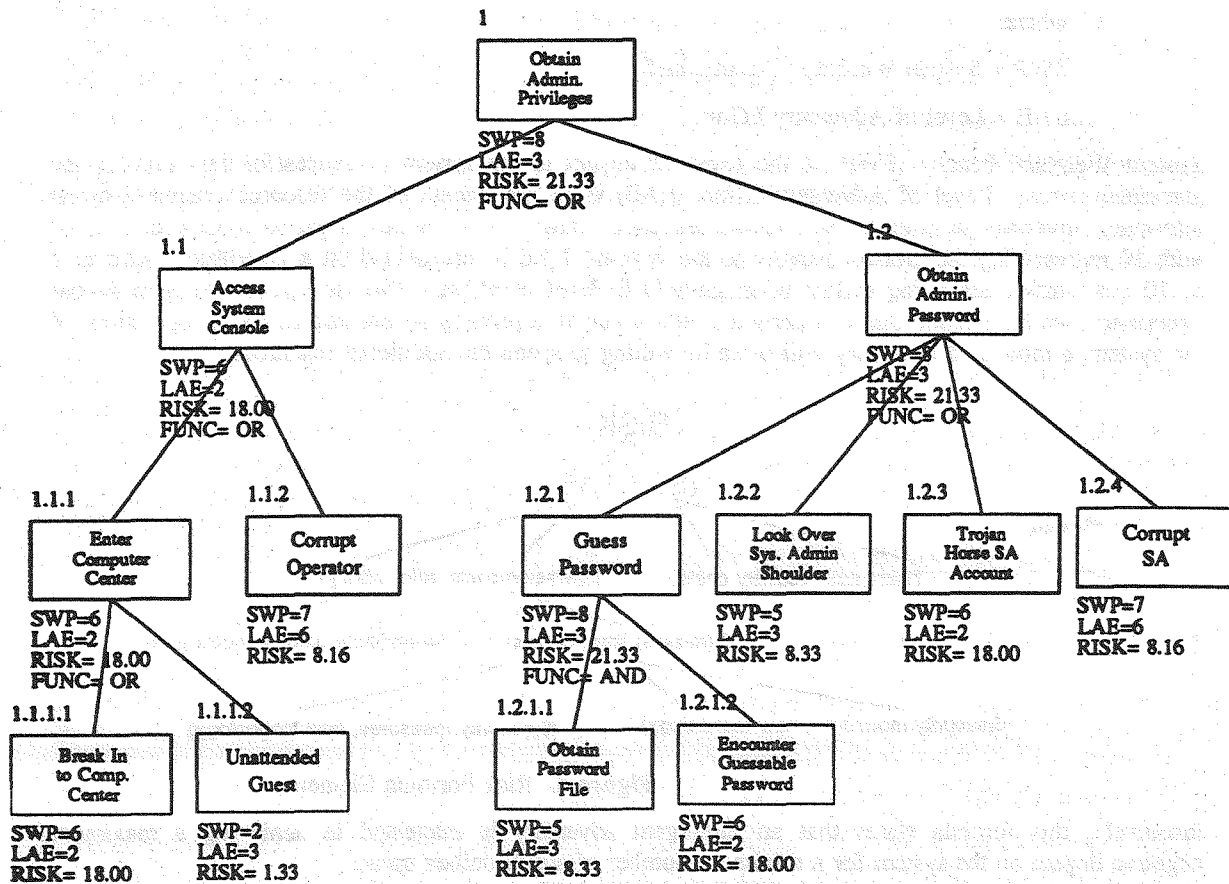


Figure 3. Threat Logic Tree Example - UNIX System Design

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.

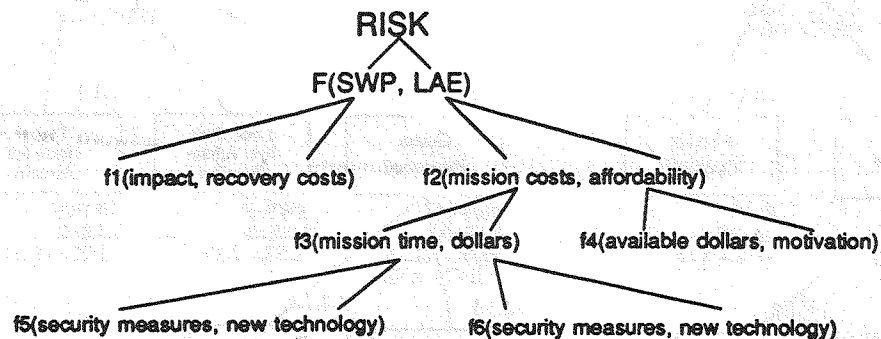


Figure 4. Risk Formula Elements

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	$\sum_{i=1}^n Max\ lae_i$	$lae_{maxR}$

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.

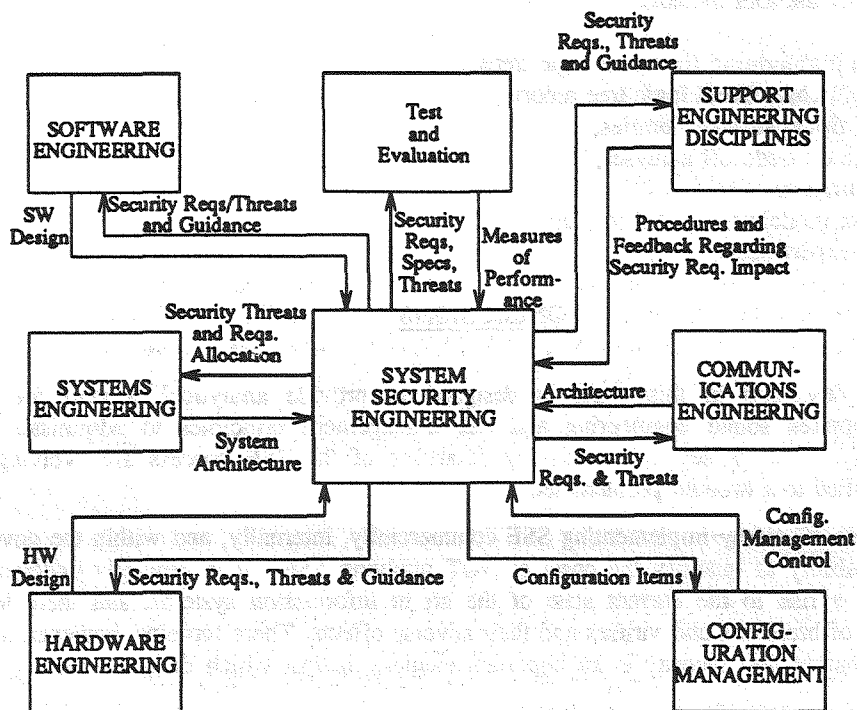


Figure 5. Example System Security Engineering Interfaces

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 Lina 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.<sup>1</sup> 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

---

<sup>1</sup> 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 in 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.<sup>2</sup>

(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.

---

<sup>2</sup> 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.<sup>3</sup>

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

---

<sup>3</sup> 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.<sup>4</sup>

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

---

<sup>4</sup> 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
	<b>INTRODUCTORY MATERIAL</b>
1	Introduction - Course overview and rules; What is Computer System Security? Confidentiality, Integrity, Availability
2	Computer Systems Security, Privacy, and Ethics
	<b>RISK ANALYSIS</b>
3	Risk Analysis - Threats, Vulnerabilities, and Countermeasures
4	Risk Analysis

	<b>THREATS, VULNERABILITIES, AND COUNTERMEASURES</b>
5	Threats and Vulnerabilities: Malicious code
6	Malicious Code: Trojan Horses, Time Bombs, Trap Doors
7	Malicious Code: Worms and Viruses
	<b>SECURE COMPUTER SYSTEMS: OVERVIEW</b>
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
	<b>SECURITY MECHANISMS</b>
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
	<b>SECURITY ASSURANCE</b>
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
	<b>NETWORK AND DATABASE SECURITY</b>
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
	<b>WRAP-UP AND SUMMARY</b>
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 veritabily 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

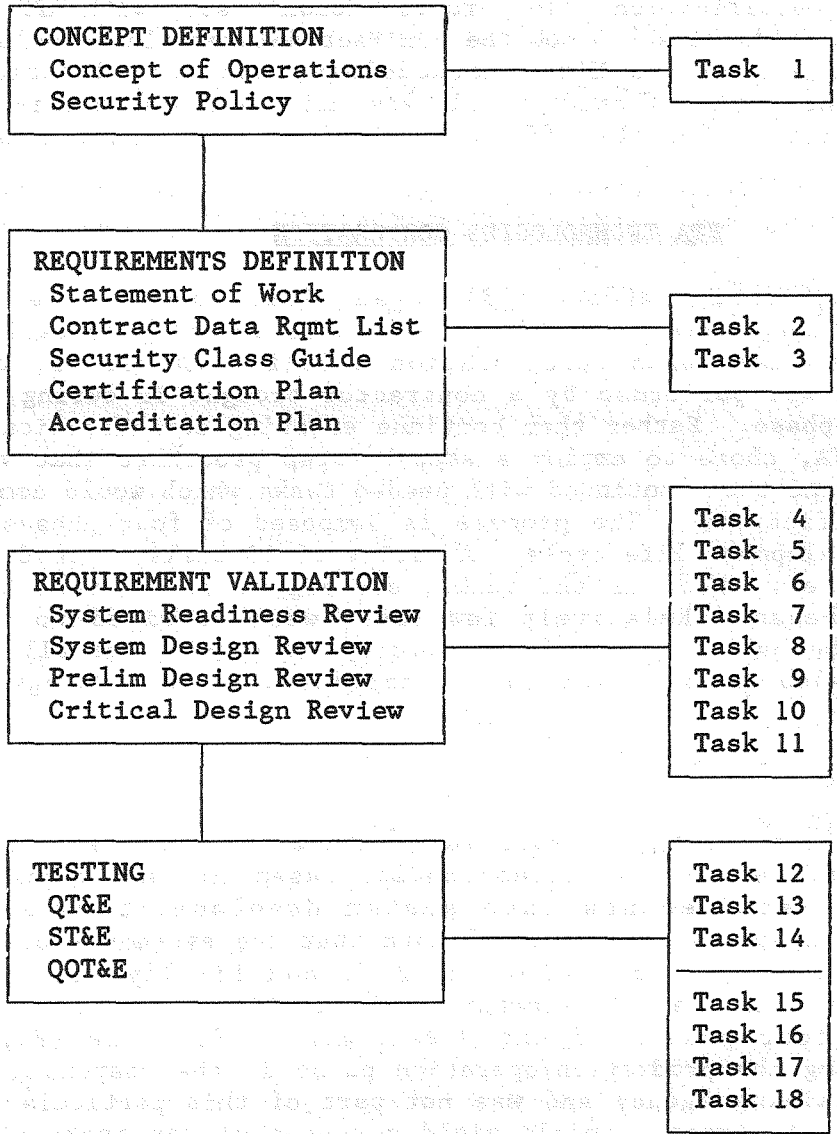
The ETA certification effort [13] began after some of the affected systems (for munitions storage and management) were actually installed. The system's developers had begun certification itself much before, but this particular effort was performed by a contractor brought in during the full scale development 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

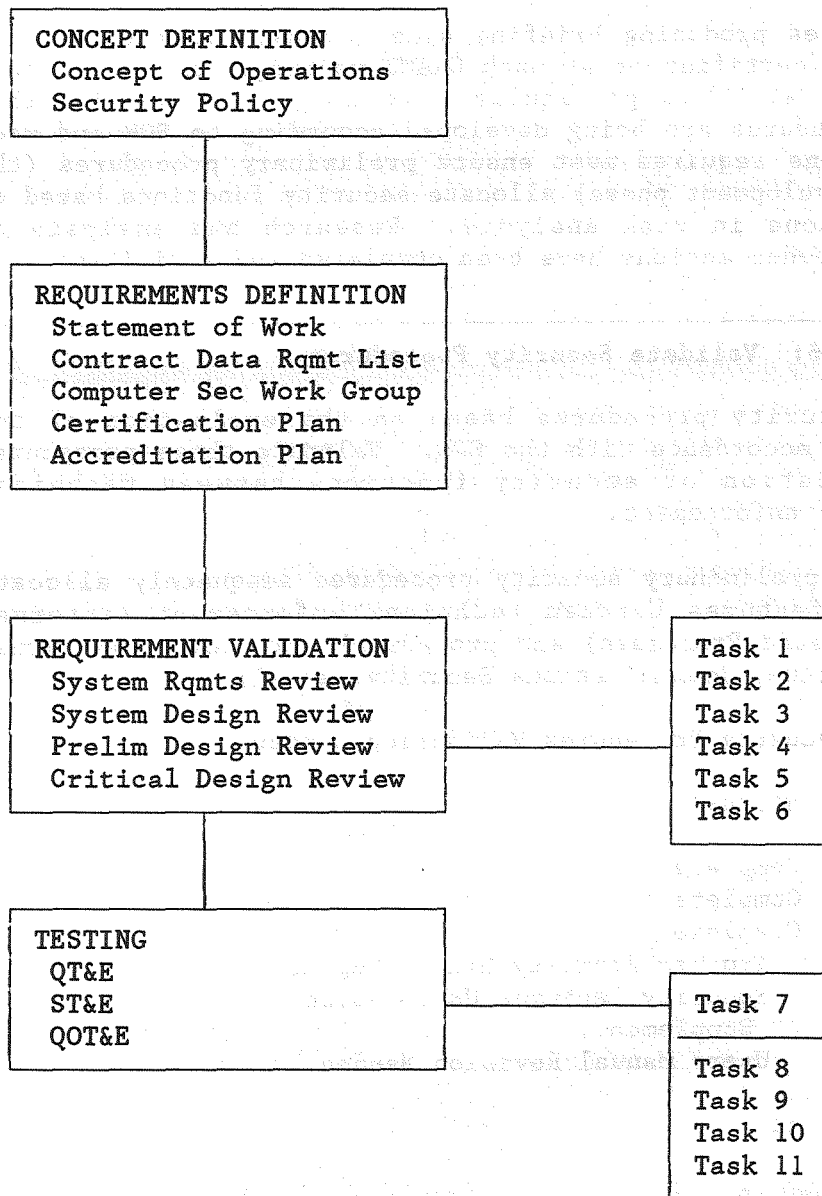


Figure 3. Accreditation Overview



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 Exploration<sup>1</sup> 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.

---

<sup>1</sup>This work was funded by Rome Air Development Center contract number F30602-87-D-0093/0006.

## 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

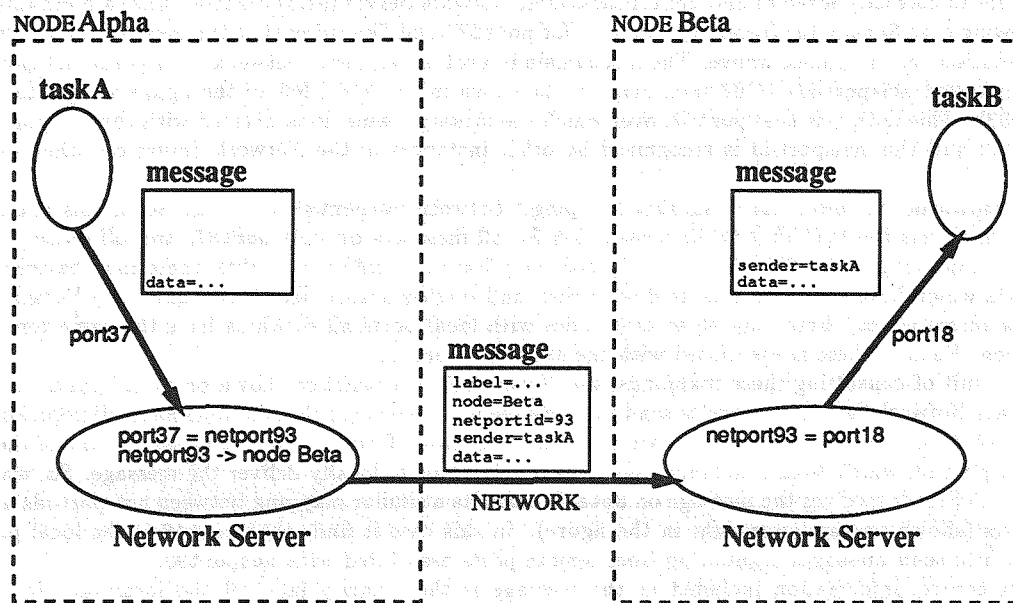


Figure 1: DTMach Message Scenario

a virtual circuit between tasks on different nodes. How this arrangement is set up and maintained is 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 strategies: 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,<sup>2</sup> 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,

<sup>2</sup>Both of these mappings are needed, because both can change independently of one another.

can be removed from the DTMach 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 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 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.

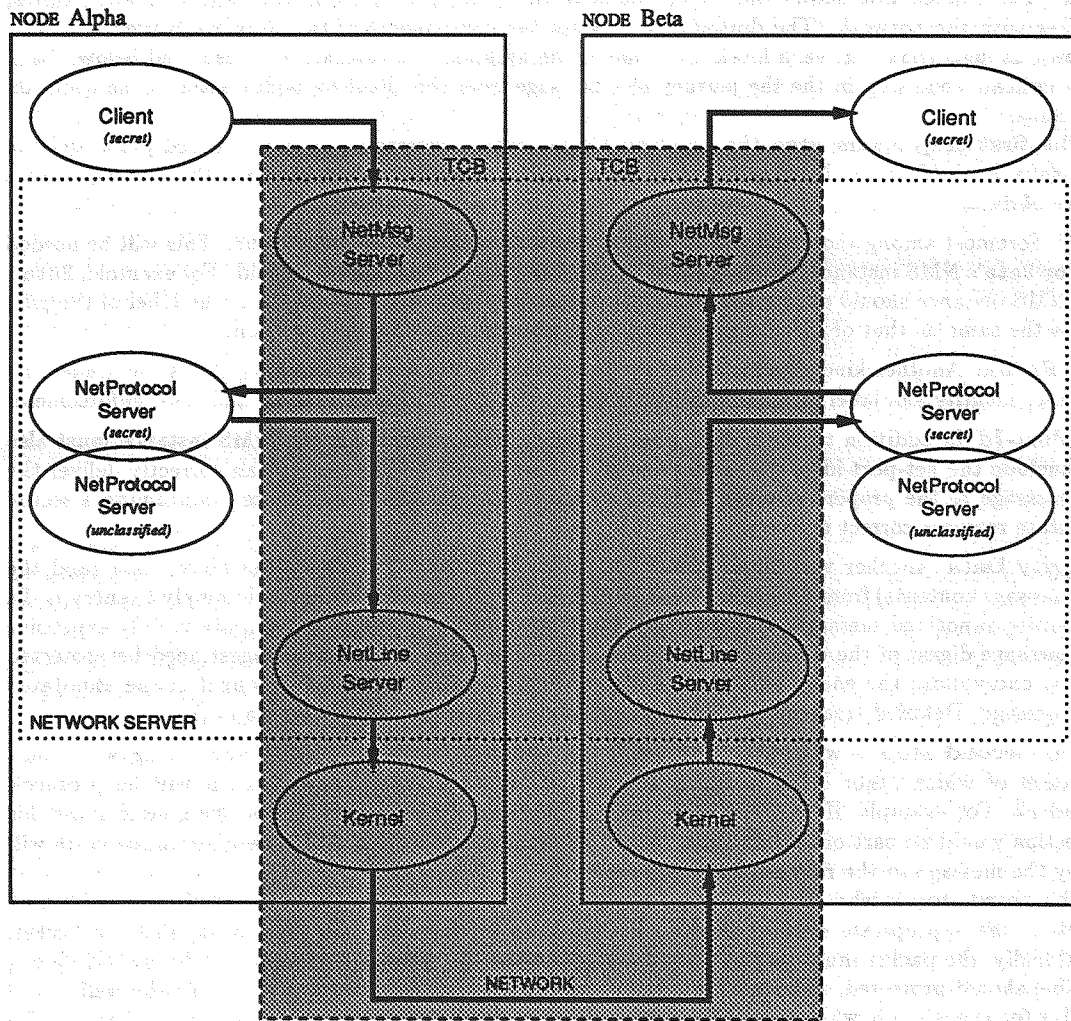


Figure 2: Message Transmission

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 encryption<sup>3</sup> (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,

<sup>3</sup>It 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.

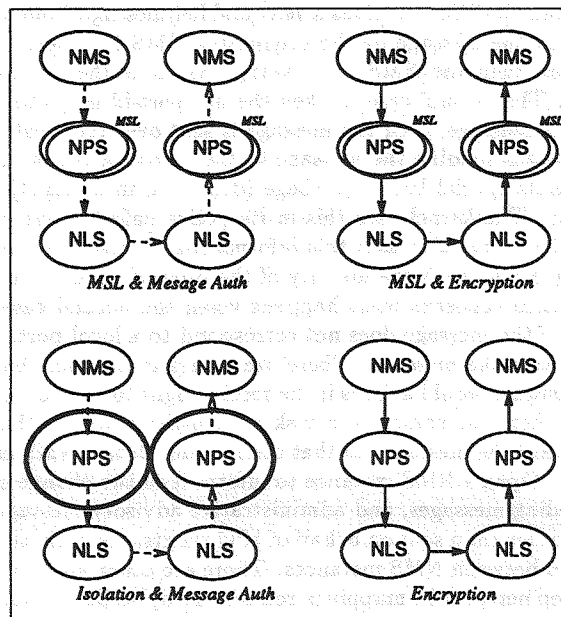


Figure 3: Four NPS Architectures

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 10<sup>th</sup> National Computer Security Conference, September 1987.
- [8] Greg King, "Considerations for VSLAN<sup>TM</sup> Integrators and DAAs," Proceedings of the 13<sup>th</sup> 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 System<sup>1</sup> (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 UNIX<sup>2</sup> 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 VMS<sup>3</sup>) 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.<sup>4</sup>

## 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."

<sup>1</sup>X Window System is a trademark of the Massachusetts Institute of Technology.

<sup>2</sup>UNIX is a registered trademark of AT&T.

<sup>3</sup>VMS is a trademark of Digital Equipment Corporation.

<sup>4</sup>The 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.

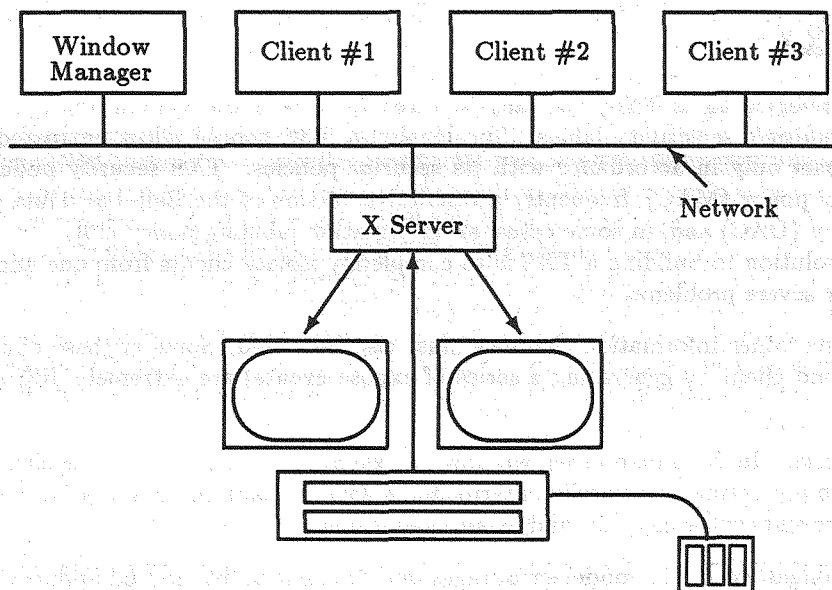


Figure 1: X Architecture

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/Motif<sup>5</sup> [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.<sup>6</sup> 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

<sup>5</sup> OSF/Motif is a trademark of the Open Software Foundation.

<sup>6</sup> Vendors 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.<sup>7</sup> Since clients choose resource IDs, it is possible to use write-up as a broad signaling channel.<sup>8</sup> For this reason, most TXSs typically support read and write equal, and a slightly constrained read down policy on their local resources.

<sup>7</sup>We 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.

<sup>8</sup>Consider 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.<sup>9</sup> 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

---

<sup>9</sup>This 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."<sup>1</sup> 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.<sup>2</sup> 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.

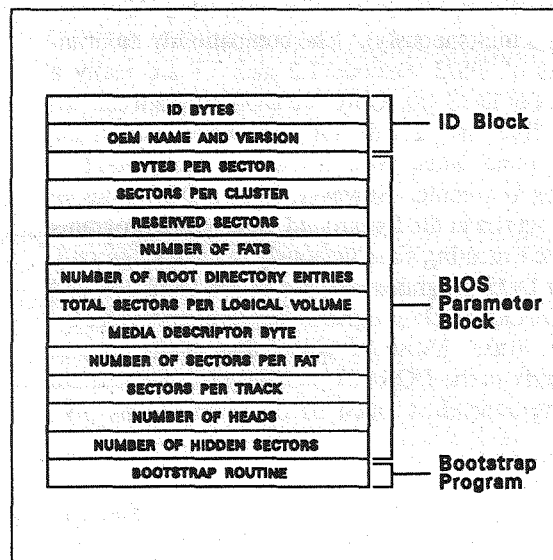


Diagram 1 - Layout of a DOS Boot Sector

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

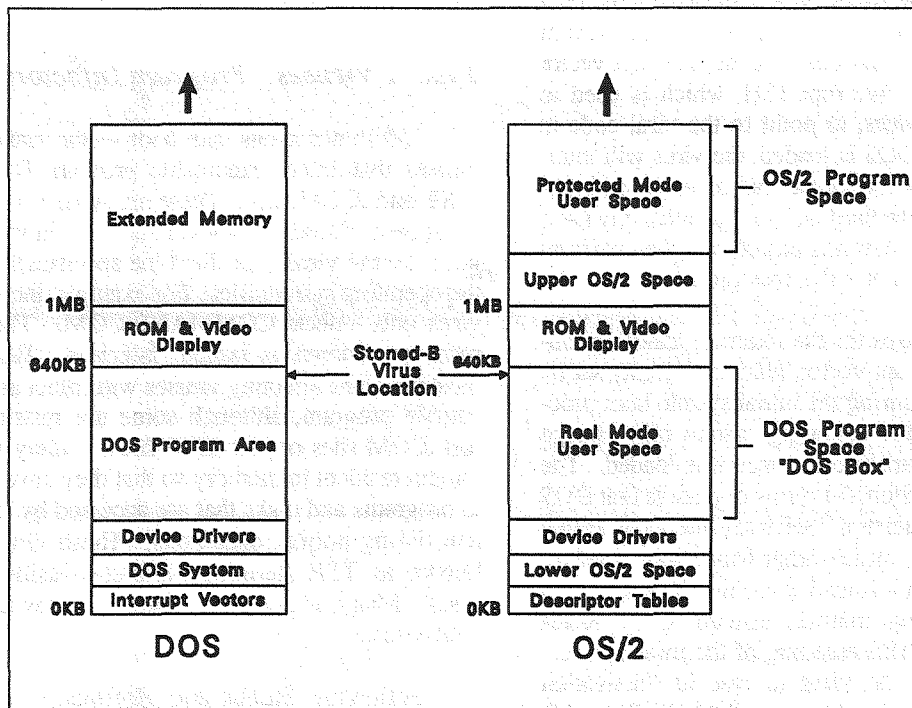


Diagram 2 - Generalized DOS and OS/2 Memory Maps

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 microprocessor 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.<sup>3</sup>

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 Hoffman<sup>4</sup>, 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,

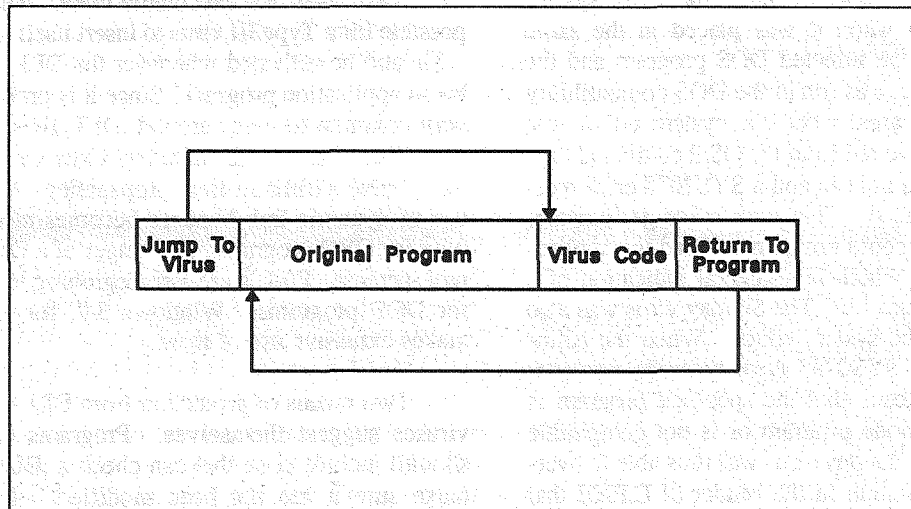


Diagram 3 - Structure of a Nonoverwriting Virus

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.<sup>5</sup> 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"**

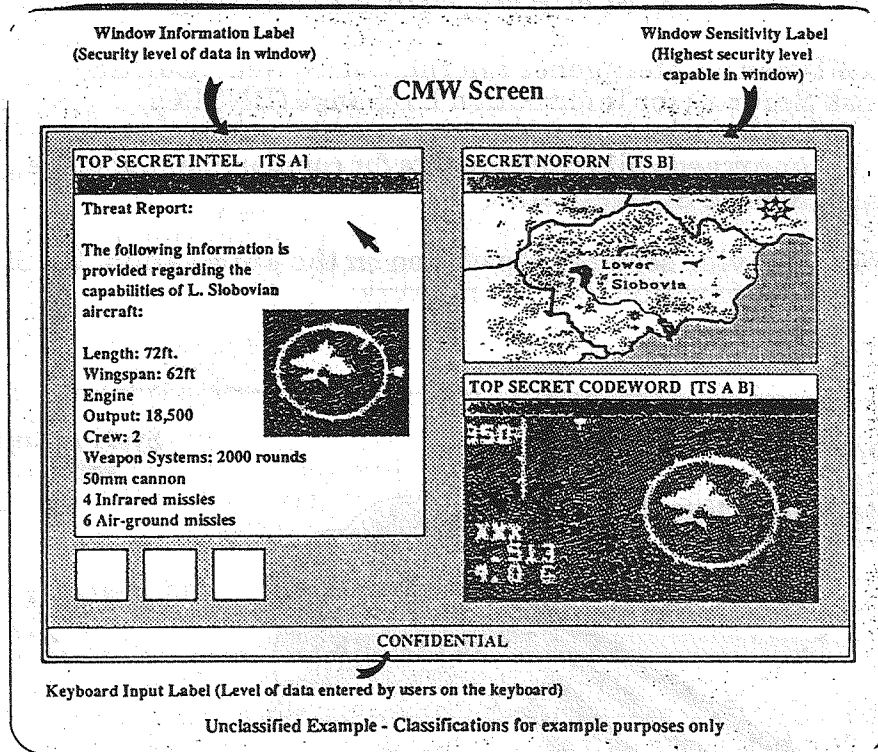


Figure 1  
CMW Program

**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

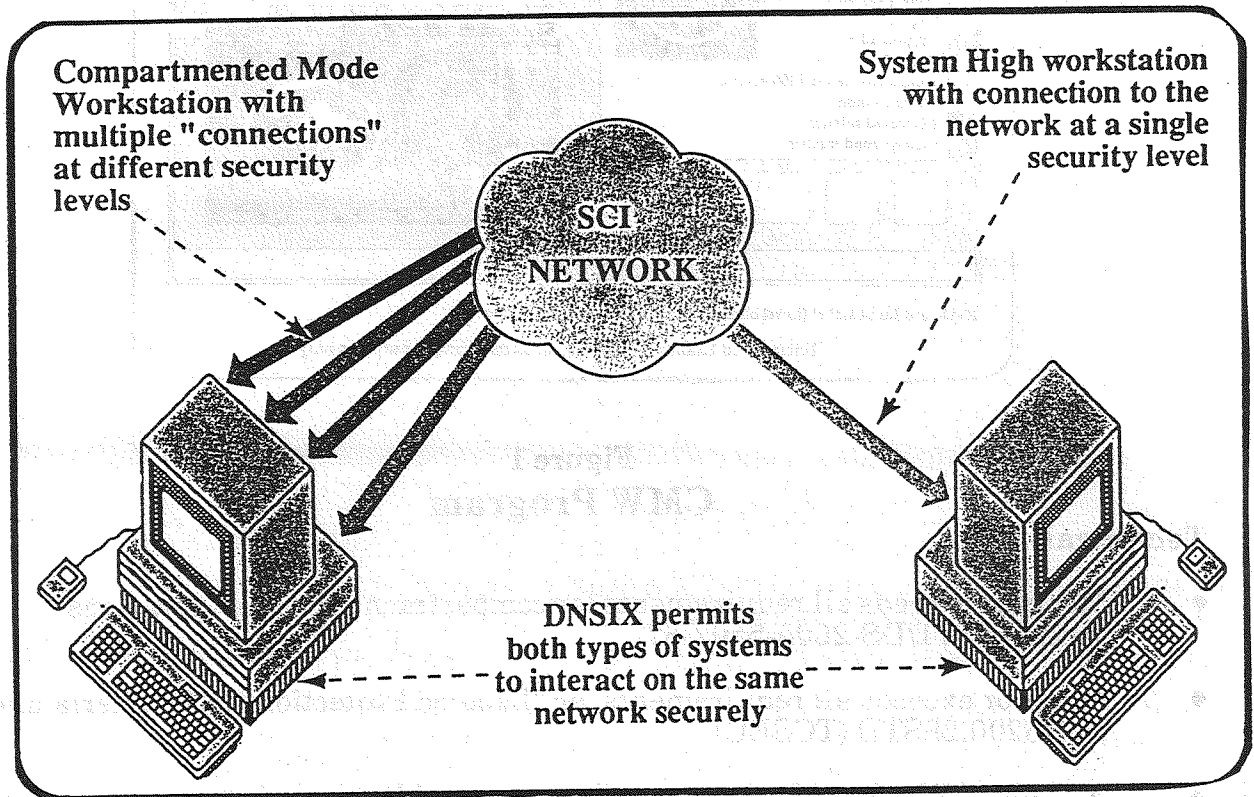
**Programmatic:**

- 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/ZVEI Working Group based on a suggestion by the EUROBIT<sup>2</sup> 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).

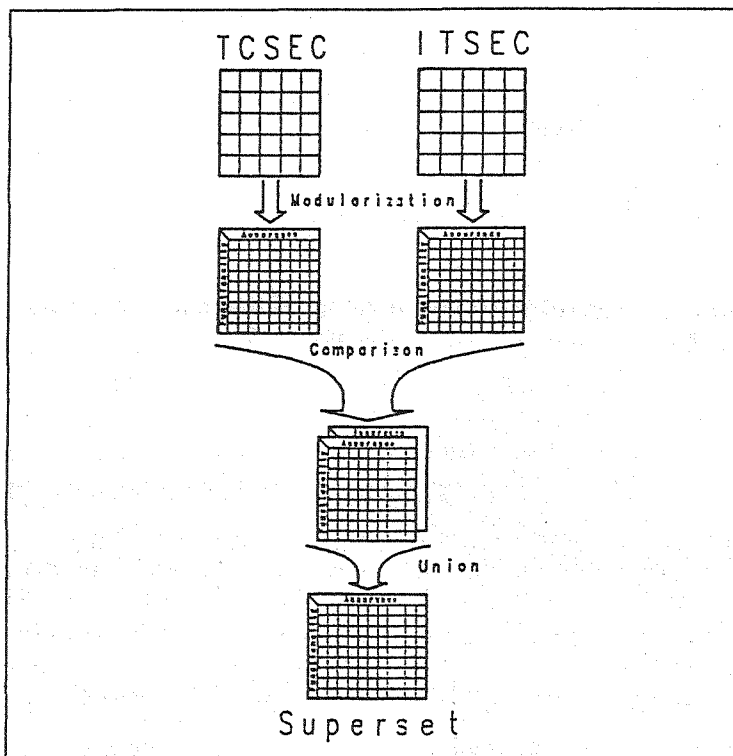


Figure 1. Concept

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.

Figure 2.  
Superset: Functionality

**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 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 study<sup>1</sup> that examined Embedded Weapon System (EWCS) vulnerability to malicious code.

## **INTRODUCTION**

The study of ECWS 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 ECWSs it is important to understand the functions of a typical weapon system. An ECWS is a computer or group of computers that is a component of a larger system used to perform a specific military mission. The ECWS 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.

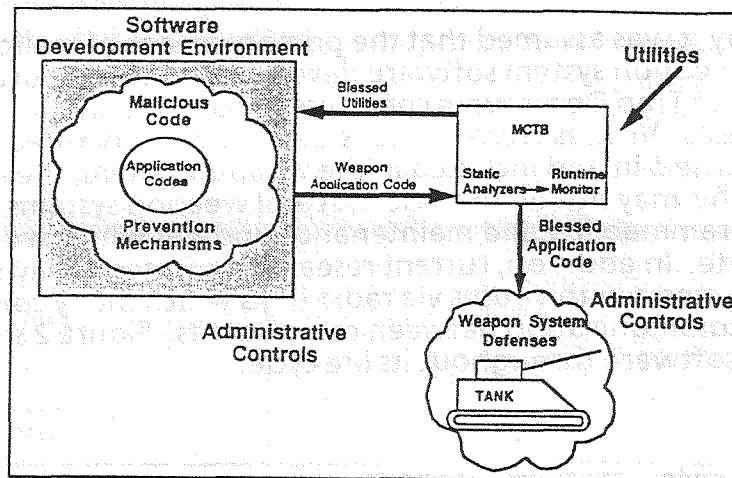


Figure 3. Malicious Code Resistant Security Architecture (MCRSA)

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.

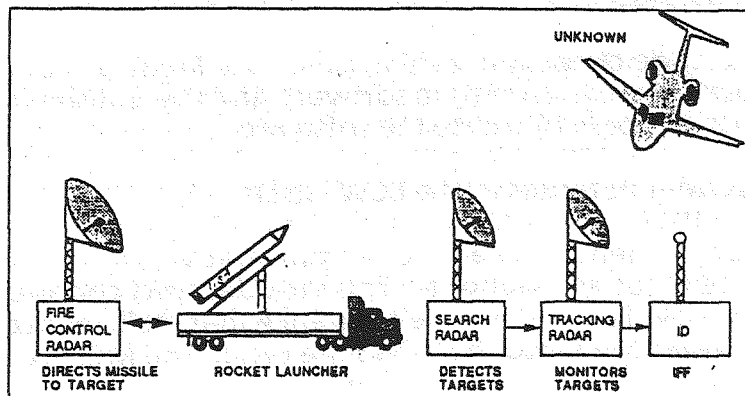


Figure 1. General Weapon System Functions

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.

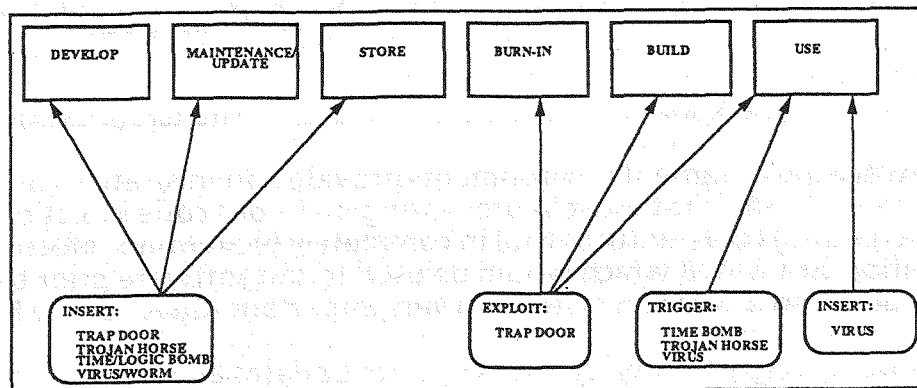


Figure 2. Software Life Cycle Vulnerability

### 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.



## *Executive Summary*

# **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.



## **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.

## **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 creditable, 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
- System-Wide Security Policy Model
- Certification and Accreditation Plan
- 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.