A TAXONOMY OF COMPUTER ATTACKS WITH APPLICATIONS TO WIRELESS NETWORKS

by

Daniel Lowry Lough

Dissertation submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Engineering

Approved:

Nathaniel J. Davis IV
(Chairman)

Ezra A. Brown

Mark T. Jones

Randy C. Marchany

Scott F. Midkiff

Charles E. Nunnally

April 2001

Blacksburg, Virginia

A Taxonomy of Computer Attacks with Applications to Wireless Networks

by

Daniel Lowry Lough

Committee Chair: Nathaniel J. Davis IV

Computer Engineering

Abstract

The majority of attacks made upon modern computers have been successful due to the exploitation of the same errors and weaknesses that have plagued computer systems for the last thirty years. Because the industry has not learned from these mistakes, new protocols and systems are not designed with the aspect of security in mind; and security that is present is typically added as an afterthought. What makes these systems so vulnerable is that the security design process is based upon assumptions that have been made in the past; assumptions which now have become obsolete or irrelevant. In addition, fundamental errors in the design and implementation of systems repeatedly occur, which lead to failures.

This research presents a comprehensive analysis of the types of attacks that are being leveled upon computer systems and the construction of a general taxonomy and methodologies that will facilitate design of secure protocols. To develop a comprehensive taxonomy, existing lists, charts, and taxonomies of host and network attacks published over the last thirty years are examined and combined, revealing common denominators among them. These common denominators, as well as new information, are assimilated to produce a broadly applicable, simpler, and more complete taxonomy. It is shown that all computer attacks can be broken into a taxonomy consisting of improper conditions: **V**alidation **E**xposure **R**andomness **D**eallocation **I**mproper **C**onditions **T**axonomy; hence described by the acronym **VERDICT**.

The developed methodologies are applicable to both wired and wireless systems, and they are applied to some existing Internet attacks to show how they can be classified under VERDICT. The methodologies are applied to the IEEE 802.11 wireless local area network protocol and numerous vulnerabilities are found. Finally, an extensive annotated bibliography is included.

Keywords:

VERDICT, computer attack taxonomy, IEEE 802.11, wireless security, integrity flaws, computer security

Dedication

This work is dedicated to the **One Almighty God**, who has called me to be His servant.

> ...for all have sinned and fall short of the glory of God...
>
> *Romans 3:23*

> ...He chose us in Him before the foundation of the world, that we should be holy and blameless before Him. In love He predestined us to adoption as sons through Jesus Christ to Himself, according to the good pleasure of His will...
>
> *Ephesians 1:4–5*

> For while we were still helpless, at the right time Christ died for the ungodly. For one will hardly die for a righteous man; though perhaps for the good man someone would dare even to die. But God demonstrates His own love toward us, in that while we were yet sinners, Christ died for us.
>
> *Romans 5:6–8*

> For God so loved the world, that He gave His only begotten Son, that whoever believes in Him should not perish, but have eternal life.
>
> *John 3:16*

Who has believed our message? And to whom has the arm of the LORD been revealed? For He grew up before Him like a tender shoot, and like a root out of parched ground; He has no stately form or majesty that we should look upon Him, nor appearance that we should desire Him. He was despised and forsaken of men, a man of sorrows, and acquainted with grief; and like one from whom men hide their face, He was despised, and we did not esteem Him. Surely our griefs He Himself bore, and our sorrows He carried; yet we ourselves esteemed Him stricken, smitten of God, and afflicted. But He was pierced through for our transgressions, He was crushed for our iniquities; the chastening for our well-being fell upon Him, and by His scourging we are healed. All of us like sheep have gone astray, each of us has turned to his own way; but the LORD has caused the iniquity of us all to fall on Him. He was oppressed and He was afflicted, yet He did not open His mouth; like a lamb that is led to slaughter, and like a sheep that is silent before its shearers, so He did not open His mouth. By oppression and judgment He was taken away; and as for His generation, who considered that He was cut off out of the land of the living, for the transgression of my people to whom the stroke was due? His grave was assigned with wicked men, yet He was with a rich man in His death, because He had done no violence, nor was there any deceit in His mouth. But the LORD was pleased to crush Him, putting Him to grief; if He would render Himself as a guilt offering, He will see His offspring, He will prolong His days, and the good pleasure of the LORD will prosper in His hand. As a result of the anguish of His soul, He will see it and be satisfied; by His knowledge the Righteous One, My Servant, will justify the many, as He will bear their iniquities. Therefore, I will allot Him a portion with the great, and He will divide the spoils with the strong; because He poured out Himself to death, and was numbered with the transgressors; yet He Himself bore the sin of many, and interceded for the transgressors.

Isaiah 53

Acknowledgements

There are many people I would like to thank, for without them, this dissertation would not be completed:

- Nathaniel Davis for being my advisor and guiding me through this dissertation process.
- The Bradley Department of Electrical and Computer Engineering for fully supporting my Ph.D. work. This work was supported by a Bradley Fellowship from the Bradley Department of Electrical and Computer Engineering and a grant from the Hekimian Laboratories.
- Randy Marchany for agreeing to serve on my committee, introducing me to people he knew, spending *many* hours talking with me about this work, and pushing me to complete the dissertation work.
- All my Ph.D. Committee — Nathaniel Davis, Ezra Brown, Mark Jones, Randy Marchany, Scott Midkiff, and Charles Nunnally — for giving me comments about how to make the work better and encouraging me.
- The staff in the department for helping me though all aspects of school.
- The staff at ILLIAD (Virginia Tech InterLibrary Loan) for supporting me off campus by mailing *numerous* copies of articles and books.
- Helmut Kopka and Patrick Daly's guide to LaTeX book that *greatly* aided me in writing this dissertation [Kopk1999].
- Aleksander Simonic for his WinEdt 2K program that is an editor for LaTeXentry.

- Richard B. Ertel who created *Bibmaker*, a Windows based program to manage the bibliographic entries for LaTeX.
- Tomas G. Rokicki, creator of *dvips* who helped me solve the problem of converting my files to Postscript for Acrobat pdf conversion.
- Biswaroop Guha et al's paper [Guha1997] for starting me to think about errors in protocols.
- Carl Landwehr for comments on his work [Land1994].
- Matt Bishop for his hospitality when I visited UC Davis, his tutorial at USENIX, and his comments on his work [Bish1996a, Bish1999].
- Clark Weissman for a copy of the original report on FHM [Weis1973].
- All the hackers I've met over the years and the knowledge they have given me.
- Alex Lostetter for many great memories, especially that of the 1996 "Sojorn."
- My friends Dan and Stephanie Ferrante; Ryan Smith; Fred and Kim Gray; Colin and Brenda Kerr; Brandon Ellison; Brett Kite; Victor Rosas; Dan Woodie; and Jeff Berner.
- My family for their caring support: Susan and Bruce Tyler; Wilfrido, Julie, Willie, and Joey Pérez; Denys and Anna Andriyenko; and Andrew, Emily, Marianna, and Edwin Cutright.
- "Mom" and "Dad" in Reva (Ed and Wanda Cutright) for the most wonderful and Godly set of in-laws a man can have.
- Dad who (not knowingly) got me started on my career by trading an old stove for a new Commodore VIC-20 computer over seventeen years ago.
- Mom who is *always* there for me and who loves me more than I can ever love back.

But most of all, to my wife and my best friend Rebecca. You are truly a Beautiful, Intelligent, and Godly Proverbs 31 woman. Words can not adequately describe how *much* I love you, and I look forward to spending each and every day of the rest of my life with you.

Contents

Abstract iii

Dedication v

Acknowledgements vii

1 Introduction 1

2 Background and Literature Survey 7

2.1 Computer Security 7

2.1.1 Traditional Areas of Computer Security 9

2.1.1.1 Confidentiality 10

2.1.1.2 Integrity 10

2.1.1.3 Availability 11

2.1.2 Information Warfare 12

2.2 Assumptions Made 16

2.2.1 Trusting the Environment 16

2.2.2 Trusting Source Code 19

2.2.3 Trusting the Application and the Trusted Computing Base 20

2.2.4 Trusting the Paradigms 24

2.2.5 Trusting the Assumptions 25

2.3 Computer Security Problems 27

2.3.1 The ARPANET Crashes in 1980, 1988, and Today 27

2.3.1.1 Modulo Time Stamp Error 27

2.3.1.2 The Internet Worm . . . 28
2.3.1.3 Crashes Today . . . 28
2.3.2 Déjà Vu All Over Again . . . 28
2.3.3 Why Computers Are Not Secure . . . 29
2.4 Penetration Testing . . . 30
2.4.1 Flaw Hypothesis Methodology . . . 30
2.4.1.1 Flaw Hypothesis Methodology Theory . . . 30
2.4.1.1.1 Knowledge of System Control Structure . . . 31
2.4.1.1.2 Flaw Hypothesis Generation . . . 31
2.4.1.1.3 Flaw Hypothesis Confirmation . . . 33
2.4.1.1.4 Flaw Generalization . . . 33
2.4.1.1.5 Flaw Elimination . . . 33
2.4.1.2 Flaw Hypothesis Methodology Application . . . 34
2.4.2 Other Penetration Resistant Methods . . . 34
2.4.2.1 Security Test and Evaluation Tools . . . 34
2.4.2.2 Hypothesis of Penetration Patterns . . . 35
2.4.2.2.1 System Isolation (or Tamperproofness) . . . 36
2.4.2.2.2 System Noncircumventability . . . 36
2.4.2.2.3 Consistency of System Global Variables and Objects 36
2.4.2.2.4 Timing Consistency of Condition Checks . . . 36
2.4.2.2.5 Elimination of Undesirable System User Dependencies 37
2.4.3 Summary of Penetration Testing . . . 37
2.5 Computer Attack Taxonomies . . . 37
2.5.1 Definition and Requirements of a Taxonomy . . . 37
2.5.2 Characteristics, Features, and Attributes of a Taxonomy . . . 40
2.6 Wireless Networking . . . 41
2.6.1 Multiple Access Methodologies . . . 41
2.6.1.1 Frequency Division Multiple Access (FDMA) . . . 41
2.6.1.2 Time Division Multiple Access (TDMA) . . . 42
2.6.1.3 Code Division Multiple Access (CDMA) . . . 42

2.6.2 Wireless Communication Standards 43
2.6.2.1 AMPS 43
2.6.2.2 PCS 43
2.6.2.3 802.11 44
2.6.2.4 Bluetooth 44
2.6.2.5 Summary of Wireless Communication Standards 45
2.6.3 Systems and Protocols 45
2.6.3.1 Mobile IP 45
2.6.3.2 Mobile Ad Hoc Networks (MANET) 45
2.7 Summary 46

3 Computer Attack Taxonomies 47
3.1 Types of Computer Misuse and Its Perpetrators 48
3.1.1 Anderson's Penetration Matrix 48
3.1.2 SRI Computer Abuse Methods Model 49
3.1.2.1 Evolution of the SRI Computer Abuse Methods Model . . . 50
3.1.2.2 Nine Categories and Twenty-Six Attacks 53
3.1.3 Lindqvist and Jonsson's Extension of Neumann and Parker 53
3.1.4 Jayaram and Morse's Network Security Taxonomy 55
3.1.5 Summary and Comparison of Misuse Taxonomies 57
3.1.5.1 Comparison of Neumann and Parker to Anderson's Penetration Matrix 58
3.1.5.2 Comparison of Lindqvist and Jonsson to Neumann and Parker 58
3.1.5.3 Comparison of Jayaram and Morse's Taxonomy to Neumann and Parker 58
3.2 From the Source to the Objective 59
3.2.1 Howard's CERT Taxonomy 59
3.2.2 Sandia Laboratory Taxonomy 60
3.3 Purdue's Taxonomy Work and Extensions 61
3.3.1 Kumar's Classification and Detection of Computer Intrusions 61

3.3.2 Aslam's UNIX Security Taxonomy 63
3.3.2.1 Aslam's Taxonomy 63
3.3.2.2 Bishop's Critical Analysis of Taxonomies 63
3.3.2.3 Krsul's Analysis of Aslam's Taxonomy 64
3.3.3 Krsul's Taxonomy 64
3.3.3.1 Krsul's Extension of Aslam 65
3.3.3.2 Bishop's Vulnerabilities Classification Scheme 65
3.3.4 Iowa State's Extension to Krsul 65
3.4 Flaw Hypothesis Methodology Penetration Lists 69
3.4.1 Linde's Generic System Functional Flaws 70
3.4.2 Attanasio's FHM Penetration Characteristics 72
3.5 Other Taxonomies and Attacks 72
3.5.1 Brinkley's Computer Misuse Techniques 73
3.5.2 Knight's Vulnerability Taxonomy 75
3.5.3 Beizer's Bug Taxonomy 77
3.5.4 SRI Security Breaching Incidents 77
3.5.5 Perry and Wallich: An Attack Matrix 79
3.5.6 Dunnigan and Cohen: Deception and Security 80
3.5.7 Parker's Taxonomies 83
3.5.7.1 Extensions to the Three Basic Categories of Security 83
3.5.7.2 Parker's Classes of Attack 83
3.5.8 Straub and Widom's Motivation-Security Response Taxonomy 86
3.5.9 Ristenbatt's Methodology for Network Communication Vulnerability 87
3.6 Summary 88

4 Operating System Integrity Flaws 89

4.1 RISOS: Research In Secured Operating Systems 89
4.1.1 Class of User 90
4.1.2 Class of Integrity Flaw 91
4.1.3 Class of Resource 92

4.1.4 Category of Method 92
4.1.5 Category of Exploitation 92
4.1.6 RISOS Operating System Security Flaws 93
4.1.6.1 R1: Incomplete Parameter Validation 93
4.1.6.2 R2: Inconsistent Parameter Validation 94
4.1.6.3 R3: Implicit Sharing of Privileged/Confidential Data 95
4.1.6.4 R4: Asynchronous Validation / Inadequate Sharing 96
4.1.6.5 R5: Inadequate Identification/Authentication/Authorization 97
4.1.6.6 R6: Violable Prohibition/Limit 99
4.1.6.7 R7: Exploitable Logic Error 100
4.1.6.8 Summary 100
4.2 PA: Protection Analysis 101
4.2.1 Protection Analysis Original Work 101
4.2.2 Protection Analysis Taxonomy Ten Error Types 103
4.2.2.1 P1: Consistency of Data Over Time 104
4.2.2.2 P2: Validation of Operands 105
4.2.2.3 P3: Residuals 105
4.2.2.4 P4: Naming 106
4.2.2.5 P5: Domain 106
4.2.2.6 P6: Serialization 107
4.2.2.7 P7: Interrupted Atomic Operations 107
4.2.2.8 P8: Exposed Representations 107
4.2.2.9 P9: Queue Management Dependencies 108
4.2.2.10 P10: Critical Operator Selection Errors 109
4.2.3 Reformulation and Reduction of Original PA Categories 109
4.2.4 PA: Four Global Error Categories with Subcategories 110
4.3 Neumann and Parker's Trapdoor Attacks 111
4.3.1 Neumann and Parker Trapdoor Types 112
4.3.1.1 TD1: Improper Identification and Authentication 112
4.3.1.2 TD2: Improper Initialization and Allocation 113

4.3.1.3 TD3: Improper Finalization (Termination and Deallocation) 113
4.3.1.4 TD4: Improper Authentication and Validation 113
4.3.1.5 TD5: Naming Flaws, Confusions, and Aliases 114
4.3.1.6 TD6: Improper Encapsulation, Such as Accessible Internals 114
4.3.1.7 TD7: Asynchronous Flaws, Such as Atomicity Anomalies . . 115
4.3.1.8 TD8: Other Logic Errors 115
4.3.2 Conclusions 115
4.4 McPhee's Seven Classes of Integrity Flaws 115
4.4.1 Time-Of-Check-To-Time-Of-Use (TOCTTOU) 115
4.4.2 McPhee's Seven Classes 116
4.4.2.1 M1: System Data in the User Area 117
4.4.2.2 M2: Nonunique Identification of System Resources 117
4.4.2.3 M3: System Violation of Storage Protection 118
4.4.2.4 M4: User Data Passed as System Data 119
4.4.2.5 M5: User-supplied Address of Protected Control Blocks . . 119
4.4.2.6 M6: Concurrent Use of Serial Resources 119
4.4.2.7 M7: Uncontrolled Sensitive System Resources 120
4.4.3 Conclusions 121
4.5 Landwehr's Taxonomy Survey Paper 121
4.5.1 By Genesis 122
4.5.2 By Time of Introduction 123
4.5.3 By Location 125
4.5.4 Conclusions 126
4.6 Summary and Comparison of Operating System Integrity Flaws 128
4.6.1 Neumann's Nine Levels of PA 128
4.6.2 Comparison of RISOS and PA (Bishop and Lough) 132
4.6.2.1 R1/R2: Incomplete/Inconsistent Parameter Validation . . . 133
4.6.2.2 R3: Implicit Sharing of Privileged/Confidential Data 133
4.6.2.3 R4: Asynchronous Validation / Inadequate Serialization . . 135
4.6.2.4 R5: Inadequate Identification/Authorization/Authentication 135

4.6.2.5 R6: Violable Prohibition Limit . . . 136
4.6.2.6 R7: Exploitable Logic Error . . . 138
4.6.2.7 Missing Categories . . . 138
4.6.3 Neumann Trapdoor Attacks and Protection Analysis (PA) . . . 139
4.6.4 McPhee Comparison with Other Taxonomies . . . 141
4.6.4.1 M1: System Data in the User Area . . . 141
4.6.4.2 M2: Non-unique Identification of System Resources . . . 143
4.6.4.3 M3: System Violation of Storage Protection . . . 143
4.6.4.4 M4: User Data Passed as System Data . . . 143
4.6.4.5 M5: User-supplied Address of Protected Control Blocks . . . 143
4.6.4.6 M6: Concurrent Use of Serial Resources . . . 143
4.6.4.7 M7: Uncontrolled Sensitive System Resources . . . 144
4.6.5 Landwehr's Comparison to Other Taxonomies . . . 144
4.6.6 Lindqvist and Jonsson's Operating System Flaws . . . 144
4.6.7 Lough's Comparison of OS Flaws . . . 146
4.7 Summary . . . 148

5 VERDICT 150

5.1 VERDICT Characteristics . . . 150
5.1.1 Fundamentum Divisionis . . . 151
5.1.1.1 Tree Structure . . . 151
5.1.1.2 Characteristics Structure . . . 152
5.1.2 Abstraction Levels . . . 152
5.1.3 Cause and Effect . . . 152
5.2 Evolution of VERDICT . . . 153
5.2.1 Summary of Operating System Attacks . . . 153
5.2.2 Addition of Randomness Category . . . 155
5.2.3 Combination of Categories . . . 155
5.2.4 The SERVR Taxonomy . . . 156
5.3 VERDICT Categories . . . 157

5.3.1 Validation . . . 158
5.3.2 Exposure . . . 158
5.3.3 Randomness . . . 158
5.3.4 Deallocation . . . 159
5.4 Summary . . . 161

6 Verification of VERDICT 162
6.1 VERDICT Applied to PA . . . 162
6.1.1 P1: Consistency of Data Over Time . . . 162
6.1.2 P2: Validation of Operands . . . 163
6.1.3 P3: Residuals . . . 163
6.1.4 P4: Naming . . . 164
6.1.5 P5: Domain . . . 164
6.1.6 P6: Serialization . . . 165
6.1.7 P7: Interrupted Atomic Operations . . . 165
6.1.8 P8: Exposed Representations . . . 166
6.1.9 P9: Queue Management Dependencies . . . 166
6.1.10 P10: Critical Operator Selection Errors . . . 166
6.1.11 Summary . . . 167
6.2 VERDICT Applied to RISOS . . . 167
6.2.1 R1: Incomplete Parameter Validation . . . 168
6.2.2 R2: Inconsistent Parameter Validation . . . 168
6.2.3 R3: Implicit Sharing of Privileged/Confidential Data . . . 168
6.2.4 R4: Asynchronous Validation / Inadequate Sharing . . . 168
6.2.5 R5: Inadequate Identification / Authentication / Authorization . . . 169
6.2.6 R6: Violable Prohibition/Limit . . . 169
6.2.7 R7: Exploitable Logic Error . . . 169
6.2.8 Summary . . . 170
6.3 VERDICT Applied to Neumann and Parker's
Computer Misuse Categories . . . 170

6.3.1 CM1: Visual Spying . . . 172
6.3.2 CM2: Misrepresentation . . . 172
6.3.3 CM3: Physical Scavenging . . . 172
6.3.4 CM4: Logical Scavenging . . . 172
6.3.5 CM5: Eavesdropping . . . 172
6.3.6 CM6: Interference . . . 173
6.3.7 CM7: Physical Attack . . . 173
6.3.8 CM8: Physical Removal . . . 173
6.3.9 CM9: Impersonation . . . 173
6.3.10 CM10: Piggybacking Attacks . . . 173
6.3.11 CM11: Spoofing Attacks . . . 174
6.3.12 CM12: Networking Weaving . . . 174
6.3.13 CM13: Trojan Horse Attacks . . . 174
6.3.14 CM14: Logic Bombs . . . 174
6.3.15 CM15: Malevolent Worms . . . 175
6.3.16 CM16: Virus Attacks . . . 175
6.3.17 CM17: Trapdoor Attacks . . . 176
6.3.18 CM18: Authorization Attacks . . . 176
6.3.19 CM19: Basic Active Misuse . . . 176
6.3.20 CM20: Incremental Attacks . . . 177
6.3.21 CM21: Denials of Service . . . 177
6.3.22 CM22: Browsing . . . 177
6.3.23 CM23: Inference, Aggregation . . . 177
6.3.24 CM24: Covert Channels . . . 177
6.3.25 CM25: Inactive Misuse . . . 178
6.3.26 CM26: Indirect Misuse . . . 178
6.3.27 Summary . . . 179
6.4 VERDICT Applied to Neumann and Parker's Trapdoor Attacks . . . 179
6.4.1 TD1: Improper Identification/Authentication . . . 179

6.4.2 TD2: Improper Initialization/Allocation 180
6.4.3 TD3: Improper Finalization (Termination and Deallocation) 180
6.4.4 TD4: Improper Authentication/Validation 180
6.4.5 TD5: Naming Flaws, Confusions, Aliases 180
6.4.6 TD6: Improper Encapsulation, such as Accessible Internals 181
6.4.7 TD7: Asynchronous Flaws, such as Atomicity Anomalies 181
6.4.8 TD8: Other Logic Errors 181
6.4.9 Summary 181
6.5 VERDICT Applied to McPhee's Integrity Flaws 182
6.5.1 M1: System Data in the User Area 182
6.5.2 M2: Nonunique Identification of System Resources 183
6.5.3 M3: System Violation of Storage Protection 183
6.5.4 M4: User Data Passed as System Data 183
6.5.5 M5: User-Supplied Address of Protected Control Blocks 184
6.5.6 M6: Concurrent Use of Serial Resources 184
6.5.7 M7: Uncontrolled Sensitive System Resources 185
6.5.8 Summary 185
6.6 VERDICT Applied to SANS Top 10 Attacks 185
6.6.1 BIND Weaknesses Allow Immediate Root Compromise 187
6.6.2 Vulnerable CGI Programs Installed on Web Servers 189
6.6.3 Remote Procedure Call (RPC) Weaknesses Yields Root 190
6.6.4 Remote Data Services Hole in MS Internet Info Server 191
6.6.5 Sendmail and MIME Buffer Overflows and Pipe Attacks 192
6.6.6 sadmind and mountd 192
6.6.7 Global File Sharing in NetBIOS, NFS, Appleshare 193
6.6.8 User IDs, Esp. Root/Admin with No or Weak Passwords 194
6.6.9 IMAP and POP Buffer Overflow or Incorrect Configuration 195
6.6.10 Default SNMP Community Strings "Public" and "Private" 195
6.6.11 Summary 196
6.7 Summary 196

7 Methodologies of VERDICT **197**

7.1 Validation . . . 197

7.1.1 Validation of Code . . . 198

7.1.1.1 Critical Conditions . . . 198

7.1.1.1.1 Outside-to-Inside Validation Methodology . . . 198

7.1.1.1.2 Inside-to-Outside Validation Methodology . . . 199

7.1.1.2 Buffer Overflows . . . 200

7.1.1.3 Summary . . . 201

7.1.2 Validation of Protocols . . . 201

7.1.2.1 Inputs . . . 202

7.1.2.2 Protocol Verifiers . . . 202

7.1.2.3 State Machines . . . 203

7.1.2.4 Past Attacks . . . 203

7.1.3 Summary . . . 204

7.2 Exposure . . . 204

7.2.1 Exposure Methodology Introduction and Observations . . . 204

7.2.2 Finding Vulnerabilities vs. Exposures . . . 205

7.2.2.1 Vulnerabilities . . . 206

7.2.2.2 Exposures . . . 206

7.2.3 Exposure Methodologies . . . 207

7.2.3.1 In-to-Out . . . 207

7.2.3.2 Exposure Matrix . . . 207

7.2.4 Summary . . . 209

7.3 Randomness . . . 209

7.3.1 Types of Random Numbers . . . 210

7.3.1.1 Perfect Random Bits . . . 210

7.3.1.2 Natural Random Bits . . . 210

7.3.1.3 Pseudo-random Bits . . . 210

7.3.2 Randomness Methodology . . . 210

7.3.3 Summary . . . 211

7.4 Deallocation . . . 211
7.4.1 Data Residuals Methodology . . . 211
7.4.2 Composition Residuals Methodology . . . 212
7.4.3 Access Residuals Methodology . . . 213
7.4.4 Summary . . . 213
7.5 Summary . . . 214

8 Application of VERDICT to IEEE 802.11 215
8.1 Summary of IEEE 802.11 . . . 215
8.1.1 IEEE 802.11 — PHY and MAC . . . 216
8.1.1.1 IEEE 802.11 Physical Layer (PHY) . . . 217
8.1.1.2 IEEE 802.11 Medium Access Control Layer (MAC) . . . 219
8.1.1.2.1 Backoff Factor . . . 219
8.1.1.2.2 RTS, CTS, and the Hidden Node Problem . . . 219
8.1.1.2.3 DCF and PCF . . . 220
8.1.1.2.4 Beacon and Probe Frames . . . 220
8.1.1.2.5 Association and Authentication . . . 221
8.1.2 Wired Equivalent Privacy (WEP) . . . 221
8.2 Security Vulnerabilities in IEEE 802.11 . . . 224
8.2.1 Application of VERDICT: Improper Validation . . . 224
8.2.1.1 802.11 Validation: MAC Address Validation? . . . 224
8.2.1.2 802.11 Validation: Invalid State? . . . 225
8.2.1.3 802.11 Validation: Forced Deauthentication/Disassociation? 225
8.2.1.4 802.11 Validation: RTS Flood? . . . 226
8.2.1.5 802.11 Validation: Fragmentation Attacks? . . . 227
8.2.1.6 802.11 Validation: Retry Overwrites (Hijacked Sessions)? . . 227
8.2.1.7 802.11 Validation: Authentication in 802.11 . . . 228
8.2.1.8 Summary . . . 229
8.2.2 Application of VERDICT: Improper Exposure . . . 229
8.2.3 Application of VERDICT: Improper Randomness . . . 231

8.2.3.1 Binary Exponential Backoff Algorithm . . . 231
8.2.3.2 Orthogonal Frequency Division Multiplexing (OFDM) . . . 231
8.2.3.3 Summary . . . 232
8.2.4 Application of VERDICT: Improper Deallocation (Residuals) . . . 232
8.2.5 Summary of Application of VERDICT to IEEE 802.11 . . . 232
8.3 Summary . . . 232

9 Conclusions **233**
9.1 Dissertation Hypothesis . . . 233
9.2 Discussion of Dissertation Research . . . 234
9.3 Contributions to the Field of Computer Engineering . . . 236
9.4 Future Work . . . 236
9.5 Summary . . . 237

A Computer Attacks **238**
A.1 Types of Computer Attacks . . . 238
A.1.1 Input/Output . . . 239
A.1.2 Design Oversights . . . 239
A.1.3 File Security . . . 240
A.1.4 Resource Limits . . . 240
A.1.5 Accounting Methods . . . 240
A.1.6 Disruption/Denial-of-Service (DoS) Attacks . . . 240
A.1.7 Object Reuse / Residuals . . . 241
A.1.8 Noncaptive Environments . . . 241
A.1.9 Privilege Escalation . . . 242
A.1.10 Parameter Checking . . . 243
A.1.11 Weak Passwords . . . 243
A.1.12 Version Rollback Attack . . . 244
A.1.13 Lack of Randomness . . . 244
A.1.14 Nonunique Identification of System Resources . . . 244
A.1.15 Overflowing Bounds . . . 245

A.1.16 Race Conditions 246
A.1.17 Salami Attacks 246
A.1.18 DoS SYN Flood Attacks 246
A.1.19 Electromagnetic Eavesdropping and TEMPEST 247
A.2 Further Information 247

B Art of Security **250**
B.1 The "Eggs"hortations of Neumann 251
B.2 Tenets of Hoffman 252
B.3 Law and Order for the Personal Computer 253
B.4 Tao of Security Tenets 254
B.5 Summary 257

Annotated Bibliography **258**

Vita **345**

List of Tables

2.1 Assumptions in the 1970s versus today 17
2.2 Linde vs. Weissman Flaw Hypothesis Methodology 31
2.3 Linde, Weissman, and Hollingworth Flaw Hypothesis Methodology 35
3.1 Anderson's Penetrator Matrix [Ande1980] 49
3.2 Variations of the SRI Computer Abuse Methods Model 52
3.3 Neumann and Parker's Categories of Computer Misuse (NP1 – NP9) 53
3.4 Neumann and Parker's Types of Computer Misuse (CM1 – CM26) [Neum1995] 54
3.5 Lindqvist and Jonsson's Intrusion Techniques 55
3.6 Lindqvist and Jonsson's Intrusion Results 56
3.7 Jayaram and Morse's Network Security Taxonomy (JM1 – JM5) 56
3.8 Comparison of Anderson and Neumann/Parker [Neum1989] 58
3.9 Comparison of Jayaram & Morse [Jaya1997] and Neumann & Parker [Neum1995] 59
3.10 Comparison of Brinkley and Schell with Other Taxonomies 75
3.11 Knight's Taxonomy and Comparison [Knig2000] 76
3.12 Beizer's Bug Taxonomy 78
3.13 Perry and Wallich Attack Matrix [Perr1984] 81
3.14 Motivation-Security Response Taxonomy [Stra1984] 87
4.1 Research in Secured Operating Systems (RISOS) categories (R1 – R7) 93
4.2 Protection Analysis (PA) categories (P1 – P10) 103
4.3 Neumann and Parker's Types of Trapdoor Attacks 112
4.4 McPhee's Classes of Integrity Problems 116
4.5 Landwehr et al.'s Flaws by Genesis [Land1994] 123
4.6 Landwehr et al.'s Flaws by Time of Introduction [Land1994] 124

4.7 Landwehr et al.'s Flaws by Location [Land1994] 125
4.8 Comparison of Neumann [Neum1978], Bishop [Bish1995], and PA [Bisb1978] Categories 131
4.9 Comparison of Bishop [Bish1995], PA, and RISOS Categories 132
4.10 Comparison of Neumann and Parker's Trapdoor Attacks and PA [Bisb1978] 140
4.11 Comparison of McPhee, RISOS, PA, and Neumann 142
4.12 Comparison of Landwehr's Flaw by Genesis to RISOS; PA; and Neumann and Parker [Land1994] 145
4.13 Comparison of Lindqvist and Jonsson [Lind1997] with Other Taxonomies . . 147
4.14 Summary of Operating System Integrity Flaws 149
5.1 Summary of Operating System Integrity Flaws 154
6.1 VERDICT Applied to PA 163
6.2 VERDICT Applied to RISOS 167
6.3 VERDICT Applied to Computer Misuse Categories 171
6.4 VERDICT Applied to Trapdoor Attacks 179
6.5 VERDICT Applied to McPhee 182
6.6 VERDICT Applied to SANS Top 10 Attacks 186
7.1 Buffer Overflow Locations 201
7.2 Vulnerability Logic 206
7.3 Exposure Logic 206
7.4 Object Domain Exposure Matrix 208
7.5 Domain Exposure Matrix 208

List of Figures

2.1 Message Digest 11
2.2 Man-in-the-Middle 12
3.1 Neumann and Parker's SRI Computer Abuse Methods Model [Neum1995] . . 51
3.2 Howard's CERT Taxonomy 60
3.3 Sandia Laboratory Taxonomy 62
3.4 Aslam's Taxonomy 64
3.5 Krsul's Taxonomy, Part I 66
3.6 Krsul's Taxonomy, Part II 67
8.1 IEEE 802.11 Independent Basic Service Set (IBSS) 217
8.2 IEEE 802.11 Extended Service Set (ESS) 218
8.3 IEEE 802.11 Hidden Node Problem 220
8.4 IEEE 802.11 State Diagram 222

Chapter 1

Introduction

The Internet as we know it today evolved from the ARPANET,[1] which was created in 1969. Dr. Leonard Kleinrock, a professor at University of California, Los Angeles (UCLA), responded in an e-mail message to my inquiries about the birth of the Internet in the following way:

> I consider the birth to have occurred on Sept 2, 1969 when the first piece of network equipment (the IMP)[2] first connected to a computer in the "outside world" running in a real environment of users. I like to say that was the point at which the Internet took its first breath of life. On Oct 29,[3] when we first sent a message to the second host on the network at SRI,[4] I like to say that the infant net uttered its first cry.

Scientists and engineers wanted to share information with each other over the newly linked computers. Lynch and Rose [Lync1993] assert that the original goal was a distributed communication network capable of withstanding a nuclear "event"; however, others disagree. Regardless of the motive, security was not an issue, as the ARPANET was designed with openness in mind. When multiuser computers were attached to the ever expanding network, many had "guest" accounts with no password for anyone to use. Users could log into different

[1] Advanced Research Project Agency Network. ARPA is presently called DARPA, the Defense Advanced Research Project Agency.

[2] Interface Message Processor

[3] 1969

[4] Stanford Research Institute

computers as guests of the host system and use or experiment with the services available. This network was limited to a few computers, and usually only those with knowledge of computers had access. However, today people think of the Internet as an "Information Superhighway," a metaphor not necessarily true.[5] More and more people and information are on the Internet, but the technology has not kept up with the appetite of the masses.

As some information on the network became more restricted, some users wanted to keep the information and the knowledge free. They wanted to keep experimenting with the new computers on the network. They were able to do more with the computers than the makers of the computers who had designed them. They developed ingenious ideas and expanded the knowledge of computers. They formed their own culture — elements that still remain today. These "hackers" of all backgrounds were truly the heroes of the information age. Denning discusses what hacking is with a hacker "Frank Drake" and summarizes her findings in [Denn1990a]; her interview with Drake is found in [Denn1990b]. For one of the best and truest history of hackers, refer to Steven Levy's book, *Hackers* [Levy1984].[6] To understand the culture of hackers, refer to the *The New Hacker's Dictionary* that includes definitions of the terms of the culture [Raym1996] or *Zen and the Art of Hacking* by Thieme [Thie1997]. In addition, refer to the etymology of the word "hack" by Grosser [Gros1988]. Throughout this dissertation, the word "hacker" will be used in the sense of [Raym1996]; that is, one who loves computers, has an intense desire to explore the computer, and can possibly make the computer do more than they were designed to do.

Some hackers have gone beyond investigating computers to see what the computers themselves could do. It is in these cases that the original meaning of hacker was defiled, and the word "cracker" is a more apt definition. Crackers, or intruders, are confused with the original meaning of hacker in popular culture. Parker classifies ten characteristics of perpetrators [Park1975b]; however, in his case studies, he only studies 17 cases. The characteristics he lists are as follows: age (young), skill level (high), relation between occupation and abuse (engaged perpetrations while on their job), abuse modi operandi (unauthorized computer use and unauthorized data manipulation), collusion (some), personal gain (half got money out

[5] A humorous essay on that term is found in this dissertation's annotated bibliography in [Vand1999].

[6] However, there have been reports of some factual errors in Levy's book.

of their exploits), differential association ("deviates from accepted practices of his associates only in small ways"), the Robin Hood syndrome (differentiated between not harming people but harming organizations), game playing (them against the computer), and dispositions (felony convictions). Since this study was done over twenty-five years ago, other studies should be done in order to better characterize the attackers. Understanding the attacker is important, for even today, computer web sites of Palestine, Israel, China, Taiwan, and the United States are presently being broken into and defaced to express political statements.

Computer security is starting to become one of the more active areas in computer science and engineering. Almost everyday some flaw is found in a protocol, a program, or system. These flaws sometimes lead to security breaches that affect many companies and nations worldwide. From a reading of current security advisories, one sees the same few types of attacks repeatedly occurring. Specifically, an advisory that is most seen is one that: affects "most every system"; is caused by a buffer overflow that results in the attacker being able to run any arbitrary program; results in the gaining of root or administrator privileges; and ascribes the solution to install the latest patches from the vendor.

As more and more computers are connected to the Internet, the world is tied closer together. The United States relies on many forms of computer communication. It seems everything these days is done by a computer, transactional data is stored on a computer, or the computer has the final word.[7] The President's Commission on Critical Infrastructure Protection (PCCIP) report detailed the extent of our dependence upon computers [Mars1997].[8] Information warfare, the art of conducting warfare operations using computers as weapons of offense and defense, is becoming a bigger topic than ever because so many things can be potentially damaged by a computer [Schw1996]. Because of these aspects of computers in our lives, security of critical infrastructures and computers themselves is vital.

Mobility of computers is also greatly increasing. When computers and the ARPANET were first being developed, computers were huge pieces of electronics that were kept behind glass walls or locked in laboratories with many technicians to oversee each action on the

[7]This was personally apparent when a grocery store chain would not sell groceries that I wanted to buy (even with cash) because the "computer" was down.

[8]PCCIP has since been encompassed into CIAO, Center for Infrastructure Assurance Office.

computer. Today, people have handheld organizers and the desire to connect almost everything to the Internet. Cellular phones are becoming more ubiquitous, and companies are extending that usage from voice-only communications to all forms of data communications.

Protocols designed in the 1970s like TCP and IP did not have a vision of mobility. Dr. Steven M. Bellovin, in a recent speech at the 8th USENIX Security Symposium, spoke of this dilemma: with standard TCP over a wired link, a decrease in the throughput was probably due to congestion; this would be solved by decreasing the rate of transfer from the source. However, with TCP over a wireless connection, the slowdown is probably due to interference (the Bit Error Rate (BER) of wireless is many orders of magnitude higher than a wired link). Hence, the correct action would be to *increase* the transfer rate from the source by repeating the last bits of information. Wireless transfers are causing many researchers to reexamine the protocols.

With wireless technologies come security problems. One can easily eavesdrop on an Advanced Mobile Phone System (AMPS) cellular phone (the original widespread first generation analog cell phone) because the system only uses standard FM signals. Everything sent via a radio transmission has a great potential to be intercepted. System administrators knew the same problem existed on Ethernet segments. There, a computer could be put in promiscuous mode and told to capture every piece of information on the Ethernet segment,[9] regardless whether the packet was supposed to be destined to it. Every packet could be searched for a user name/password combination. The solution was to develop switched Ethernet where the each system was connected only to a router, and only the two computers that were communicating were directly connected through the router itself. No other system can read the packets between the intended communicators. With wireless communications, the same problem of "sniffing the wire" has reappeared.

Both wired and wireless systems have security problems. Attacks are laid upon each, but certainly more to wired systems due to the wireless systems' lesser permeation in the marketplace. However, the majority of attacks made upon modern computers have been successful due to the exploitation of the same errors and weaknesses that have plagued computer systems for the last thirty years. Because the industry has not learned from

[9]This is called "sniffing" or "sniffing the wire".

these mistakes, new protocols are not designed with the aspect of security in mind; and the security that is present is typically added as an afterthought. What makes these systems so vulnerable is the fact that in part this security design process is based upon assumptions that have been made in the past; assumptions which now have become obsolete or irrelevant.

What is needed in computer security research is a comprehensive analysis of the types of attacks that are being leveled upon computer systems and the construction of a universal taxonomy with methodologies to apply the taxonomy. The taxonomy and methodologies will facilitate design of secure protocols. Therefore, the central hypothesis of this dissertation is the following:

> ***A finite number of types of computer attacks and vulnerabilities can be classified into a taxonomy, and the taxonomy along with applicable methodologies can be used to predict future attacks.***

To that end, this research has contributed the following to the field of computer engineering and the field of computer security:

1. There exists a finite number of types of computer attacks and vulnerabilities;
2. Computer attack taxonomies presented in the past have a common set of categories;
3. Those categories can be classified into a common unified taxonomy called **VERDICT**: **V**alidation **E**xposure **R**andomness **D**eallocation **I**mproper **C**onditions **T**axonomy;
4. Methodologies and algorithms are developed for each of the four VERDICT categories;
5. Developed methodologies and algorithms are applied to predict future attacks in IEEE 802.11.

The dissertation proceeds as follows: Chapter 2 presents a background of computer security, and examines the assumptions that have been made in the past. Chapter 3 examines existing lists, charts, and taxonomies of computer attacks; assumptions made about security; and the different types of host and network attacks. In order to develop a comprehensive taxonomy, computer attack taxonomies published over the last thirty years are combined,

revealing common denominators among them in Chapter 4. These common denominators, as well as new information, are assimilated to produce a broadly applicable taxonomy called VERDICT presented in Chapter 5. The new taxonomy is verified by comparison with the seminal taxonomies of the past in Chapter 6, and the newly developed methodologies of VERDICT are covered in Chapter 7. Because interfacing with wireless systems involves a third spatial dimension not seen in traditional networks, this dissertation addresses the unique challenges of wireless network security applicable in the combined taxonomy. This taxonomy is applied to the IEEE 802.11 protocol to ascertain if there are any security weaknesses in Chapter 8. This dissertation will predict some of the challenges the industry may face in the years to come. Conclusions of such are presented in the final chapter, Chapter 9. Appendix A lists types of common computer attacks and Appendix B gives a listing of computer security wisdom.

Chapter 2

Background and Literature Survey

An overview of computer security, assumptions made when designing protocols, problems with current computer security, penetration testing, and wireless technology will be examined in this chapter. Section 2.1 discusses what computer security is and gives a broad overview of the field. Section 2.2 discusses assumptions that are made in designing computer security protocols, while Section 2.3 describes what the main problems are in computer security. Section 2.4 summarizes the theory of finding flaws in systems, and Section 2.5 describes the definitions and requirements of a taxonomy as well as the characteristics, features, and attributes of one. Finally, a background in wireless networking is given in Section 2.6, and the chapter is summarized in Section 2.7.

2.1 Computer Security

Things go wrong, but why do they go wrong? Dörner discusses the logic of failure of systems with or without computers in [Dörn1996]; it is a fascinating read for those dealing with security and the failure thereof. In addition, see Peterson's book [Pete1995] on eradicating computer bugs. More specifically, computer security is a broad area of computer science and engineering. Many books have been written on computer security including: general topics [Ahuj1996, Amor1994, Anon1997, Pfle1997, Whit1996], UNIX security [Curr1992, Garf1996], World Wide Web (WWW) security [Garf1997], Java Security [McGr1997], and network security [Kauf1995, Pipk1997, Stal1995]. Ross Anderson has written a recent comprehensive book

on the *engineering* of security solutions that is highly recommended [Ande2001]. Schneier has also written a general book about the threats and the limitations of technology in [Schn2000]. A general article on LAN[1] security [Abra1995b] can be found with other tutorial papers in in [Abra1995a]. The proceedings of the first twenty years (1980–1999) of the present premier conference in computer security (IEEE Symposium on Security and Privacy) can be found on [IEEE1999]. Before discussing computer security, it is useful to discuss security in general.

The need for security has existed since the dawn of time. There has always been a need to protect physical assets from others. When something was needed or wanted, force was often used when the person holding or owning that item did not want to give it up or share it. Hence, mankind has fought for dominance of the battlefield and the society in which he lives. Protection and safety of goods or people has driven the need for fortifications and armies to ward off the invaders [Burs1994].

Information has also often needed protection. From battleplans of the Romans to the latest quarterly numbers in a corporation, information is often as vital as the physical assets that it represents. The solution has often been to protect information with cryptography. Cryptography, or hidden writing,[2] is the art of making messages secret and the ability to making them readable again. Julius Caesar is credited with using a substitution of alphabetic letters to encipher messages sent via courier. The Germans used character substitution and transposition in their three rotor Enigma machine. These are just two of the examples of the many uses of cryptography. Schneier has written one of most thorough and understandable studies of cryptography in [Schn1996]. For a truly "comprehensive history of secret communication from ancient times to the Internet," refer to Kahn's magnum opus *The Codebreakers* [Kahn1996].

Cryptography and the policy and role of the United States Government is a debate at the present time. Should "unbreakable" cryptography be allowed to be in the hands of organized crime, terrorists, pornographers, and drug traffickers? Should the government and law enforcement have the right in the public safety to be able to read anything relating to a crime, perhaps even to help save a kidnapped victim? Dorothy Denning, an advocate of

[1]Local Area Network.

[2]Cryptography is an English compound word (*crypto* and *graph*) derived from two Greek words: *kryptos* meaning hidden and *graphon* meaning to write [Morr1985b].

the U.S. Government's "clipper chip" and others agree that the government should have the right to keep keys ("key escrow") to break any message under the proper law enforcement procedures. See Schneier [Schn1996], pp. 591–593 for information on "clipper" including how to defeat the key escrow system.

Others disagree. They think that cryptography, no matter how strong, should be able to be used by anyone for privacy, including for example dissidents under a hostile regime. See Hoffman [Hoff1995] and Schneier [Schn1997] for many papers on cryptographic policy. Caloyannides writes a two-paper series on "encryption wars" in [Calo2000a, Calo2000b].

The National Research Council, under Dam et al. produced a report, *Cryptography's Role In Securing the Information Society* (CRISIS) that outlines what is the United States government's role in cryptography should be. They argue that the present policy of restricting certain cryptography should be reversed, along with other recommendations, seen in the annotation of [Dam1996]. In another National Research Council report, Clark et al. examine directions of research in relationship to infrastructure and how the government should involve itself [Clar1990].

The U.S. Government, like most other sectors of this country, has relied on computers to store and classify information; weak security has existed for many years (see Section 2.3), and the Committee on Governmental Affairs of the United States Senate held hearings in 1998 that discussed the risk [Sena1998]. Section 2.1.1 outlines the three classic areas of computer security. (Further extensions of the three classic areas are given in Section 2.5.) Information warfare is outlined in Section 2.1.2.

2.1.1 Traditional Areas of Computer Security

Security has traditionally consisted of ensuring correct disclosure (or confidentiality) of data, the complete integrity of the data, and the availability of the data when needed (that is, service is not denied) [Amor1994, Gass1988]. Section 2.1.1.1 discusses disclosure, Section 2.1.1.2 covers integrity of data, and Denial of Service (DoS) or availability, is outlined in Section 2.1.1.3.

2.1.1.1 Confidentiality

Confidentiality, or sometimes referred to as disclosure, is the first aspect of the three traditional areas of computer security. Today, printed information that is sensitive to the United States government is labeled "classified." There are three basic levels to classified documents: Confidential, Secret, and Top Secret. Within each level, there may be compartmental classifications detailing who may see the document (No Foreign Nationals, NATO, etc.). In the early 1970s, government and researchers wanted to expand the idea of control over information disclosure into the realm of information stored on a computer. Levels of security were invented and outlined in the Trusted Computer System Evaluation Criteria (TCSEC) [DoD1985] to determine in part, what the security of a system is and the requirements to keep it secure. To protect information from unauthorized disclosure, encryption is usually used. In addition, data structures in the operating system and even hardware itself may also aid in this protection.

2.1.1.2 Integrity

A second aspect of computer security is the integrity, or soundness,[3] of the information. Information that needs to be constant, or information that must only be modified by a certain authorized set of users must have the guarantee that it will not be modified by an unauthorized user. This can be accomplished through the use of cryptographic hashes, or message digests (See Figure 2.1 on page 11). Message digests take a arbitrary length message and, using a one-way function, transform it into a fixed length message. Any modification of the original message would yield a different hash, thus showing to a high probability that the original message (or file) was modified [Schn1996].

Can an operating system be provably secure? Neumann et al. discuss such issues in [Neum1975]. In addition, see [Will1995] for information on "assurance" and [Lapr1995] for concepts on reliability, availability, safety, security, and maintainability. In addition, see papers on reliability in [Litt1995] and the followup paper [Olov1995].

[3]Integrity is derived from *integritas* in Latin meaning soundness [Morr1985b].

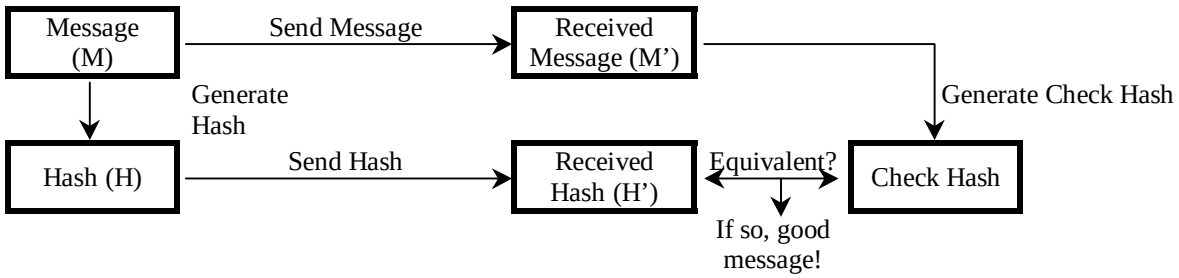

Figure 2.1: Message Digest

2.1.1.3 Availability

Availability, the opposite of the ability to deny information is the third aspect of computer security. This is the protection against a Denial-of-Service (DoS) attack; see for example, [Garb2000]. A DoS attack is exactly what it sounds like: the inability of a user, process, or system to get the service that it needs or wants. By preventing the service from happening, information whose integrity may or may not be intact can not be disclosed, even to an authorized user. A formal description and a key paper on denial of service and the concept of Maximum Waiting Time (MWT) is found in [Glig1983].

In the Senate hearing mentioned above [Sena1998] in Section 2.1, hackers of the L0pht[4] testified that with a few packets they could "bring the Internet down" within 30 minutes. They asserted that they could launch a DoS attack to the connecting points of the long haul providers, effectively cutting the links between the providers. No one using one provider could talk with a computer using another provider.

The L0pht continued by saying that DoS attacks could be leveled upon one long haul network forcing the routing protocols to use another functioning long haul network. If one is able to reroute packets to another long haul network that ones has access to, it may be possible to perform a man-in-the-middle attack. A man-in-the-middle attack consists of the attacker being between two parties trying to communicate. The attacker sees everything and could pass information as he wants (perhaps even modified) to the second party claiming to be the first party and visa-versa, as seen in Figure 2.2.

[4]L-zero-p-h-t pronounced "loft" is a hacker think tank in the Boston, Massachusetts area that was bought out January 2000 by a venture capital firm @Stake. The L0pht is presently the research division of the newly formed company @Stake. See http://www.l0pht.com and http://www.atstake.com for the home pages of the L0pht and @Stake.

Figure 2.2: Man-in-the-Middle

Denial of Service does not always involve just the lack of availability. It may also include elements of loss of integrity. Needham discusses attacks where substitute messages are sent back to a client so that the client thinks all is well [Need1994]. There are many defenses; see Richardson [Rich2001] for an overview of different defenses.

2.1.2 Information Warfare

Information Warfare has become a buzz word in the field of security, but it is becoming more mainstream in the defense and intelligence communities. If it is to be considered a part of mainstream warfare theory, it is just in its infancy; however, as it is an infant, it could prove to be a most devastating child. The U.S. is trying to figure out how to use the new technology of cyber warfare and cyber defenses [Grah1999, Drog1999]. Even the legal issues of information warfare are discussed in [DoD1999].

But what is information warfare? A 1994 Defense Science Board report [DoD1994] quotes a draft Department of Defense (DoD) unclassified definition as, "Actions taken to achieve information superiority in support of national military strategy by affecting adversary information and information systems." The same report muses that information warfare is considered to be the next revolutionary technology.[5] The report continues by quoting Russian general officers who commented on the current DoD policy of mitigating attacks on information and information infrastructure and saying:

> This view of warfare is made clear in the October 1991 observation of Lieutenant General Bogdanov, Chief of the General Staff Center for Operational and Strategic Studies, that "Iraq lost the war before it even began. This was a war of

[5]...behind the long bow, gunpowder, repeating rifles, armored vehicles, military aircraft, code breaking, radar, the transistor, nuclear weapons, guided missiles, and stealth.

> intelligence, electronic warfare (EW), command and control and counter intelligence. Iraqi troops were blinded and deafened.... Modern war can be won by informatika (sic) and that is now vital for both the U.S. and USSR." In a similar vein, Major General G. Kirilenko wrote in the June 4, 1991 issue of Komsomolskaia Pravada, "...the number of barrels and ammunition, aircraft and bombs is no longer the important factor. It is the computers that control them, the communications that makes it possible to manage force on the battlefield, land the reconnaissance and concealment assets that highlight the enemy's dispositions and cloak one's own. (sic)"

Deception has been used in warfare since the dawn of time. Dunnigan and Nofi overview numerous cases of deception from the ancients to modern times [Dunn1995]. Refer to Section 3.5.6 for examples of how computers in information warfare can use the nine techniques of deception to gain advantage in warfare.

In 1998, John Arquilla, who worked for RAND, wrote a fictional article in Wired magazine that describes a cyberwar of information warfare in 2002 [Arqu1998]. It was taken so seriously that the National Security Agency (NSA) hired Dr. Robert Anderson, another RAND consultant to write a Indications and Warnings (I&W) brief about what the United States could do now to prepare for such a situation.[6] His journal article on the subject of "cyberwar" and "netwar" is [Arqu1993].

Many people wonder if the United States will face an "electronic Pearl Harbor" or "Global Chernobyl."[7] The former Director of Central Intelligence (DCI) thinks that the cyber threats we are facing are extremely serious. In his testimony before the 1996 Senate subcommittee hearing, "Security in Cyberspace," [Sena1996] the DCI had the following exchange with ranking minority member Senator Sam Nunn:

> Senator NUNN. If you gave some sense of priority in terms of the threats we face in the future, where would you rate this overall threat we are discussing

[6]Information of this hiring was given in a presentation given at Shadowcon 1999 in Dahlgren, Virginia by [Pall1999].

[7]The author does not know the original reference for these coined phrases; they are mentioned by various participants in [Sena1996].

> this morning — the whole threat of cyberspace attack, both in terms of defense resources as well as infrastructure, economy and so forth — fit in the scale of potential threats?
>
> Mr. DEUTCH. I would say it is very, very close to the top, especially if you ask me to look 10 years down the road. I would say that after the threats from weapons of mass destruction, from rogue states and the proliferation of nuclear, chemical and biological weapons, this would fall right under it; it is right next in priority, and it is a subject that is going to be with us for a long time. It is not going to be handled in the next 6 months or 18 months. The threat is going to evolve, and our ability to deal with that threat is going to take time. The scale of time here, I think, is more like decades than it is months.[8]

In written testimony before the same Senate subcommittee, the United States General Accounting Office [GAO1996b] (See also [GAO1996a]) discusses the national security concerns and states that:

> Several studies document this looming problem. An October 1994 report entitled Information Architecture for the Battlefield [DoD1994] prepared by the Defense Science Board underscores that a structured information systems attack could be prepared and exercised by a foreign country or terrorist group under the guise of unstructured hacker-like activity and, thus, could "cripple U.S. operational readiness and military effectiveness." The Board added that "the threat... goes well beyond the Department. Every aspect of modern life is tied to a computer system at some point, and most of these systems are relatively unprotected."

Indeed. The Critical Infrastructure Working Group (CIWG) identified the critical infrastructures as: "Telecommunications, Electrical Power Systems; Gas and Oil; Banking and Finance; Transportation; Water Supply Systems; Emergency services (including medical,

[8] It is interesting to note that just three days after leaving the CIA in 1996, he had "enormously sensitive material"[Loeb2000b] on his computers at home [Loeb2000a]. He did not seem to take his own advice to heart.

police, and fire and rescue services); and Continuity of Government and Government Operations" [Sena1996].[9] Consider the Staff Statement of the Hearings on Security in Cyberspace. They report society's dependence on the National Information Infrastructure (NII)[10] and Global Information Infrastructure (GII)[11] by giving the dependence by communications, money, economy, health care, aeronautics, railway, and government operations. giving many facts. Some of the more incredible facts include the following:

> ...one major bank transfers approximately $600 billion electronically per day to the Federal Reserve. Over $2 trillion is sent in international wire transfers every day.... Within our national defense structure, over 95% of the military's communications utilize the public switched network.

Information warfare is waged at different levels of society. Winn Schwartau, one of the first authors to publish a survey book in this field [Schw1996] (first edition 1994), discusses three classes of Information Warfare: Class 1 (Personal Information Warfare), Class 2 (Corporate Information Warfare), and Class 3 (Global Information Warfare). Personal Information Warfare involves the use of computers to attack or get desired information about another person through the use of public or private databases, eavesdropping, and other nefarious means. Corporate Information Warfare uses similar means (and perhaps more costly ones, depending on the funds available) to get information from business competitors. Finally, Schwartau describes Global Information Warfare to be, "...waged against industries, political spheres of influence, global economic forces, or even against entire countries" [Schw1996]. These are the types of cyberspace attacks that former Director Deutch talked about.

Information warfare is a limited tool of the present that may become a more comprehensive tool of the future. [Beha1997] describes how computers in corporations can be attacked,

[9] The Critical Infrastructure Working Group (CIWG) mentioned in [Sena1996] is the group that was established by the Attorney General and commissioned by Presidential Decision Directive (PDD) 39. This group later wrote the Report of the President's Commission on Critical Infrastructure Protection (PCCIP) [Mars1997].

[10] "...that system of advanced computer systems, databases, and telecommunications networks throughout the United States that make electronic information widely available and accessible." [This is the definition used by the National Information Infrastructure Security Issues Forum.] This includes the Internet, the public switched network, and cable, wireless, and satellite communications [Sena1996].

[11] "The National Information Infrastructure is merely a subset of what has become known as the Global Information Infrastructure..." [Sena1996].

and [Levy1996] writes how hackers crack cryptography. Many volumes could be written on information warfare ([Denn1999] and [Walt1998] are two recent books), but many tools of it are derived from the use of computers and are only possible by the holes and vulnerabilities therein. A lot of holes and vulnerabilities come from the misjudged assumptions in the design process. Section 2.2 discusses what these assumptions are and how they have changed over the years.

2.2 Assumptions Made

When anything is created, assumptions are made. It is the same in the creation of computers and programs that use them and run on them. This section will discuss the assumptions that are made in computer security, and how some assumptions lead to bad designs or implementations. Krsul [Krsu1998] cites that 63% problems occur because of assumptions and not from design errors. Section 2.2.1 looks at how the computing environment has changed over the last 20 years and what effect that has on computer security. Section 2.2.2 describes Ken Thompson's seminal paper on trusting source code itself, and Section 2.2.3 shows that applications and the trusted computing bases that are supposed to be secure may not be the solid base that designers assume. In the fourth section, Section 2.2.4, the subject/object model of traditional security may need to be changed. Finally, Section 2.2.5 ties the thoughts brought forth in the referenced papers together with some final thoughts.

2.2.1 Trusting the Environment

Before looking at other assumptions made in computer security, let us look at the basic computing environment. What implicit assumptions were made in the late 1970s, and what implicit assumptions are still around that may not need to be? Roger Needham discusses these issues in his 1997 paper *The Changing Environment for Security Protocols* [Need1997] where he states that the assumptions we make with security protocols has changed dramatically in twenty years. Table 2.1 on page 17 cites the difference in the assumptions. Needham shows memory on computers and in storage capacities has greatly increased, clocks have become more reliable, and processing power has increased so that encryption and decryption

Table 2.1: Assumptions in the 1970s versus today

State	1970s Assumptions	Today's Reality
Computers	Shared resource	Individuals have personal computers
Memory Storage	Large physical size Small capacity memory Small capacity disks	Small physical size Large capacity memory Large capacity disks
Clocks	Unreliable (looked at watch)	Reliable
Encryption and Decryption	Slow Operations	Fast Operations
Public Key Cryptography	New; untrusted	Acceptable
Practicality of Multiple Principals	Pairwise communication impractical; need authentication server (trusted third party)	"Secure enclaves" behind firewalls to lessen pairwise authentication
Transactions	Future will be all electronic	People do meet face-to-face and over phone; can exchange authentication data

algorithms are much faster. When multiple people had to share secrets, pairwise sharing of keys was impractical thus mandating the need for a trusted third party. One-time pads used to be very bulky and the use was less practical. Today, Needham states that they are practical. In another instance he shows that with a fairly good sized lookup table, security protocols can be designed with improved characteristics.

Needham notes that because of these changes, protocols can be designed differently. Specifically, we should use the abundance of memory to our benefit; he gives two examples. First, he notes that in the past, one-time pads were unusable due to the size of the non-repeating pad to which the plaintext is exclusive-or'd (XOR). However, one-time-pads can now be exchanged in person because an, "...Exabyte tape contains about 8 Gb, which will furnish a megabyte of one-time pad a day for 20 years."[12] Or one could use many one-time

[12]The author must mean gigabytes (usually noted GB) and not gigabits (usually noted Gb), since 20 years is 7,305 days (365.25 days/year) and one megabyte times 7,305 is about 7.66 gigabytes. Eight gigabits is one gigabyte, which storage capacity would not cover the required 20 years.

keys. "To have available 128-bit keys sufficient to use one-a-day for three years is trivial. A gigabyte of disk will hold that many for 64,000 confidential correspondents, which is as many as a lot of organizations have."[13]

The other use of memory cited in the paper is the use of lookup tables. Before computers, lookup tables were used for mathematical functions. The primary function of the ENIAC (Electronic Numerical Integrator and Computer, 1946) was to calculate firing tables for artillery [Gold1996]. But, the first computers did not have a lot of memory, so table lookups were not used often.[14] Presently, computers do have enough spare memory to use lookup tables, and Needham suggests through an example using an ATM[15] system that they can be used in some cryptographic situations to ensure confidentiality.

Finally, Needham notes that once a one-time pad connection is setup between two parties, third party arbiters are not needed except to resolve disputes on non-repudiation matters. In conclusion, he notes, "It is very easy for a particular set of assumptions to become institutionalized among the members of a research community, and remain in place for much longer than is justifiable." All the researchers in the computer security field need to reevaluate protocols and see what kind of implicit assumptions they are making. In the future, assumptions may change.

[13]Technically, 128 bit keys are 16 bytes each. Three years (not even counting a leap year) is 1,095 days. With 17,520 bytes needed for three years with each of the 64,000 clients, the total number of bytes needed is approximately 1.12 gigabytes. Slightly more than the noted one gigabyte, but close enough.

[14]An interesting historical side note: In the mid-1980s, various groups "released" and spread around programs that demonstrated their group's programming skills. These programs were called "demos" (a shortened version of demonstrations) or more accurately "demoz." (The word "demoz" was noted with a "z" as the last letter because of the culture of a hacker to change letters into numbers or other letters (much like vanity license plates today) (e='3', l='1', o='0', s='z', etc.) [Raym1996].) These "demoz" often contained as much simultaneous on-screen graphical manipulations as was possible; the more one was able to put on the screen at a time, the "better" programmer one was, or the better "group" one was a member of.

The "demoz" produced on the Commodore line of computers (Commodore 64, Commodore 128, Commodore Amiga, etc.) were much faster than the similarly produced "demoz" on the current IBM computers (IBM XT, IBM AT, etc.) of the same era because of the table lookups utilized. Commodore computers used table lookups for sine and cosine calculations when running these "demoz" for speed and computational efficiency. The Commodore 64 ran at only *one* megahertz and did not have the computing power to calculate on-the-fly Taylor series polynomials. Neither did the microprocessor of the Commodore 64 (Motorola 6502) have the trigonometric instructions built into its microcode. Actually, the Amiga had specialized graphic coprocessors, but to calculate where the spheres needed to be on the screen did require main CPU calculations. For a recent article on the history of the Amiga and the attempt to resurrect the technology, see [Wall2001].

[15]Automatic Teller Machine, as opposed to Asynchronous Transfer Mode.

2.2.2 Trusting Source Code

Ken Thompson, along with Dennis Ritchie, created UNIX in the early 1970s. Thompson gave a speech at his receipt of the Turing Award from the ACM. That speech was reprinted in the Communications of the ACM, and it became a critically seminal paper in the field of computer security. In *Reflections on Trusting Trust* [Thom1984], Thompson outlines the "cutest program... [he] ...ever wrote." This "cute" program was designed to print out a copy of its own source code. The functionality sounded innocuous. He describes this program and goes on to describe how the C compiler "learns" what escape characters (like the newline escape character) mean. The key is to see what happens if one extends this concept into system commands, such as *login*. The compiler is modified to recognize the source code of the UNIX login command and compile it with a built-in back door. Since one would be able to see the modified compiler source code, rendering the ruse moot, he modifies the compiler again to recognize the standard C compiler source code, allowing the login Trojan horse and the "evil" C compiler source to be built into the completed compiled C compiler. The original benign C compiler source code and the original benign login C source code is left on the system. Thus, when one recompiles the C compiler from the "benign" source, the "evil" C compiler is produced. Thus, the Trojan horses are replicated, with no evidence in the source code. What was the moral of his speech?

> The moral is obvious. You can't trust code that you did not totally create yourself. (Especially code from companies that employ people like me.) No amount of source-level verification or scrutiny will protect you from using untrusted code. In demonstrating the possibility of this kind of attack, I picked on the C compiler. I could have picked on any program-handling program such as an assembler, a loader, or even hardware microcode. As the level of program gets lower, these bugs will be harder and harder to detect. A well-installed microcode bug will be almost impossible to detect.

That is a hugely powerful moral. The assumption that basic source code can be trusted has had the rug pulled out from underneath it. With companies such as Microsoft having 85% of the operating system market and computers with critical data using those operating

systems, is not Microsoft itself a great security concern?[16] Although I refer to Microsoft, any software manufacturer (or most hardware manufacturers) could have the potential opportunity to do just as Thompson suggested many years ago.

Even if the source code is trusted, Boyle et al. discuss problems with compilers themselves and "correctness-preserving transformations" to ensure reliability of programs [Boyl1999]. Trusted compilers are needed in addition to the trusted code.

2.2.3 Trusting the Application and the Trusted Computing Base

Bob Blakley's abstract in *The Emperor's Old Armor* states in part [Blak1996]:

> The traditional model of computer security... rests on three fundamental foundations: management of security *policy* describing the set of actions each user is entitled to perform, *integrity* of the physical system, its software, and especially its security-enforcing mechanisms, and *secrecy* of cryptographic keys and sensitive data. ...the traditional model of computer security is no longer viable, and that new definitions of the security problem are needed before the industry can begin to work toward effective security in the new environment.

The reference monitor[17] that monitors everything in the system is hard to build correctly.[18] Blakley quotes a small study done by Kate Finney [Finn1996] to determine how difficult it is for programmers could read formal specifications (in this case, they were written in system Z). The results were that nearly a third of the group could not answer any of the three questions given. If that is the case, Blakley argues, how hard is it to make formal specifications, and furthermore the integrity of the systems secure? Secrecy is hard to

[16] Recent government antitrust suits against Microsoft leave the prospect open of splitting Microsoft. But, Microsoft may be split along product lines (Windows and Application Software) so the security threat will probably still be there. Even in military systems, Microsoft Windows is being used as an underlying base. Questions about the reliability of Windows is seen, for example, in [Nanc2000].

[17] An access control/mediation concept that refers to an abstract machine that mediates all accesses to objects by subjects [Abra1995a].

[18] See [Ande1972a, Ande1972b]; Amoroso describes these papers in his annotated bibliography in [Amor1994] as the following: "James Anderson is often credited with having introduced the reference monitor concept in this report based on an earlier work done by Butler Lampson. Anderson made some of the earliest contributions to computer security, including this work, which was written years before most people began to recognize security as an issue.

maintain because people cannot keep secrets. Social engineering[19] is used to get information out of people because people, not computers, tend to be the weakest link in the security chain. Industrial espionage, which can be carried out in part by social engineering is a huge problem [Wink1996]. Blakley's manifesto is pretty clear:

> No viable secure system design can be based on the principles of Policy, Integrity, and Secrecy, because in the modern world Integrity and Secrecy are not achievable and Policy is not manageable. That is why computer security is starting to fail and why it will continue to fail until it is re-built on new foundations.

Building on his manifesto, Blakley gives examples of "new fundamentals:" that which looks simple is not necessarily so, inherent vs. imposed properties, and economic models. Firstly, he quotes Clausewitz's [Clau1993] statement on how war looks simple, but so many minor incidents add up to make the overall goal difficult to obtain; Clausewitz calls this "friction." Secondly, Blakley theorizes how programmers try to make the world better sometimes without thinking why things are the way they are. He gives an example by stating:

> $1 Billion US, in $100 bills, occupies perhaps 15 cubic yards. At prices as this is written, $1 Billion US, in gold, weighs about 80 tons. $1 Billion US, in electronic cash, on the other hand, is 32 bits plus some application-dependent headers. This is madness — surely a prescription for fraud on a breathtaking scale.... The size and weight of cash is inconvenient. It was *designed* to be inconvenient — precisely so that there would be inherent limits to the scale on which fraud, smuggling, and theft are feasible.... The temptation to make electronic cash better (than physical cash) by removing the inconvenient relationship between value and size is natural — and it should be resisted.

He suggests that electronic cash (e-cash) should be given "physicality" by tying its value to size. For example, one could take the value of e-cash and squaring it to yield the storage

[19] "...cracking techniques that rely on weaknesses in wetware [(probably from the novels of Rudy Rucker)... the human nervous system, as opposed to computer hardware or software.[Raym1996]] rather than software; the aim is to trick people into revealing passwords or other information that compromises a target system's security. Classic scams include phoning up a mark who has the required information and posing as a field service tech or a fellow employee with an urgent access problem" [Raym1996].

capacity in bits. His $1 Billion US would be equivalent to 1 Billion terabits, something that most common people do not have, only large institutions and governments. Another example includes transforming the length of copyrighted data to be proportional to its price. Time would be needed to download the desired document, and perhaps copyright fees could be paid from the connect time. His final example is to not tie privacy with secrecy. By charging a high fee, say $100,000 per access, no one would reasonably look at another person's medical records (thus insuring privacy). However, a doctor could pay $100,000 to get to the records, and the patient would pay back the access money, because it would appear on the bill!

His thoughts of software approach is still seen in use today. Software (and the design of protocols) is done to make the protocol work, and then security is added in as an afterthought.

> This is particularly true in the case of security; we build systems under the assumption that everyone is authorized to do everything, and then we build in *authentication* and *access control* mechanisms to limit the actions of particular users. This means that in most cases, security is a property which is imposed on the system rather than a property which is inherent in the system.

It is along this thought that Dixie Baker's 1996 paper *Fortresses Built Upon Sand* [Bake1996] states that "philosophy of protection... expect, assume, and depend upon systems to:

- Behave predictably; they should do what we think they will do.
- Be available when we need them.
- Be safe; they should not do what we don't want them to do.
- Be capable of protecting our data from unwanted disclosure, modification, and destruction.
- Respond quickly.

In other words, systems should be *trustworthy*." Baker says in reference to users accepting the fact that computer programs crash, "Instead of moving computer science forward

in developing systems, we appear to be going backwards in our acceptance of mediocrity. 'Correctness' is not the issue; 'dependability' is." Baker argues that the system itself is not trustworthy.[20] Loscocco builds upon that assertion.

Loscocco begins his paper, *The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments* by saying, "Current security efforts suffer from the flawed assumption that adequate security can be provided in applications with the existing security mechanisms of mainstream operating systems" [Losc1998]. He asserts that without a secure operating system as the base of the application space, applications cannot be secure on their own. Mandatory security[21] is needed to restrict the damage caused by malicious applications. A trusted path[22] or in the case of networked computers, a "...mechanism that guarantees a mutually authenticated channel, or *protected path*, is necessary to ensure that critical system functions are not being spoofed" [Losc1998]. Loscocco shows that firewalls need to have mandatory security to prevent leakage, and security layers such as IPSEC,[23] SSL,[24] and TLS[25] are useless without secure endpoints. For information on firewalls and building firewalls, see [Chap1995], the classic book [Ches1994], and [Bell1994]. For a description of distributed firewalls, see [Bell1999].

Dr. Gene Spafford offers a quote to that end in [Garf1997] by stating:

> Secure web servers are the equivalent of heavy armored cars. The problem is, they are being used to transfer rolls of coins and checks written in crayon by people on park benches to merchants doing business in cardboard boxes from beneath highway bridges. Further, the roads are subject to random detours, anyone with a screwdriver can control the traffic lights, and there are no police.[26]

[20] If a system does crash, an analysis is sometimes done to determine why a system crashed. This analysis techniques are discussed in a high amount detail for UNIX in [Drak1995].

[21] "...a mandatory security policy is considered to be any security policy where the definition of the policy logic and the assignment of security attributes is tightly controlled by a system security policy administrator" [Losc1998].

[22] "...a mechanism by which a user may directly interact with trusted software, which can only be activated by either the user or the trusted software and may not be imitated by other software [DoD1985].

[23] Internet Protocol Security [Kent1998].

[24] Secure Socket Layer was created at Netscape Corporation. SSL version 2 was created Feb 9, 1995 (Kipp Hickman, "The SSL Protocol," Netscape Communications Corp.) and SSL version 3 was created November 18, 1996 (A. Frier, P. Karlton, and P. Kocher, "The SSL 3.0 Protocol," Netscape Communications Corp.). These two references were gotten out of RFC 2246 [Dier1999]. SSL was patented [Elga1997] in 1997.

[25] Transport Layer Security [Dier1999].

[26] In a private e-mail message to me from Dr. Spafford regarding the reference of the origination of this

There are design flaws in cryptography systems as well. See Schneier's article [Schn1998b] for further information in general, and Schneier and Mudge's articles on PPTP specifically in [Schn1998a, Schn1999].

2.2.4 Trusting the Paradigms

The traditional security paradigm is to see the world through subjects and objects. Subjects are users that use (or need access to) objects such as files. Ruth Nelson comments that this paradigm, "...captures the concept of access control for the data resources of the system, but does not consider access to specific processing functionality. The assumption is that security can be modeled in terms of access to data. Behavior of 'untrusted' code is assumed to be non-security-relevant, possibly hostile and subject to contamination by Trojan Horses deliberately trying to leak data" [Nels1994].

Does the paradigm need to change? She notes that no matter how secure a system is (even A1 by [DoD1985] standards), leaks can occur when data flow must go to a "lower" environment. Her example is an A1 system that has a Trojaned untrusted application running that can control which of the other two A1 systems on the networks it can communicate with. The very network addresses compose a two character alphabet in which to leak data.

One could prevent this by requiring the communicating entities always send data so that the data or the destination of the data itself would not give away a secret.[27] However, sending meaningless data on a multinode network would cause traffic to always be at 100%. Some sort of synchronous agreement would have to be worked out so that collisions would not send the effective throughput plummeting.

Nelson proposes changing the paradigm from subjects and objects to the three-tuple human users, programs that access and produce data, and data itself. Is the very paradigm of security and trusted models a good assumption?

quote said, "I originally came up with an abbreviated version of this quote during an invited presentation at SuperComputing 95 (December of 1995) in San Diego. The quote at that time was everything up to the 'Further....'."

[27] This is similar to the problem that the Pentagon had during the Gulf War in 1991. Whenever some major operation was happening at the Pentagon or the White House, many more than usual pizza delivery trucks would deliver pizza for those working late inside. The media had to just look at the frequency and number of the pizza trucks to have a fairly good idea of when some military change was about to come about.

2.2.5 Trusting the Assumptions

Are all systems created equal? Designers of Intrusion Detection Systems (IDS) think so. Passively, they watch packets flow by and try to find signs of occurring attacks. However, as Ptacek and Newsham present in [Ptac1998], some IDS systems will accept packets that the "target" (watched) system will reject and visa-versa. This causes problems in that packets can be made to insert into the IDS and not the "target," and packets can be made to evade IDS systems and get to the "target" system.

This problem occurs because one system does not know what the other system will or will not accept as valid data. Paradoxically however, if the entire enterprise's equipment is homogeneous, one vulnerability can destroy all of the systems in a single attack. The solution seems to be heterogeneous systems that know what the other systems view as valid data. Even aircraft autopilot systems use different processors to prevent a single "power interrupt or power surge" to destroy both systems. And if the two systems disagree on the "answer" or "solution," a human pilot (or a third computer) can decide the solution. Much can be learned from the avionics industry [Rich2001].[28]

Another assumption that one should question is that general assumptions stay the same. As stated in Section 2.2.1, assumptions may change in the future. For example, designers make assumptions based on wired networks; does the advent of wireless networks affect how protocols are to be designed? (See Chapter 1 on page 4 for Bellovin's TCP example.) And in general, data is vulnerable in a multitude of locations [Park1984].

There are assumptions that all designers make when protocols are crafted. However, they have changed since the basic protocols of the Internet (IP, TCP, etc.) were created [Post1981a, Post1981b]. Do the assumptions of the 1970s have to be revisited and rethought? In 1970s, the number and size of messages designed into protocols was minimized due to limited memory and limited processing power. However, today, as Needham [Need1997] reminds us, we have much faster computers with much more memory. As far as computer technology is concerned, yes, protocols can change now that more modern machines are here.

However, what about wireless? Because of the power usage and bandwidth limitations with wireless communications, we are still held by the restrictions of the 1970s when dealing

[28] [Rich2001], pp. 33–34.

with wireless handheld computers. Perhaps we do need to continue to restrict the number of operations needed for wireless due to bandwidth and price constriction. Wireless is going to change our paradigm in looking at things not only in protocol development but also in security.

Dr. Benjamin Franklin commented in the eighteenth century on the idea of troops descending from the clouds (perhaps in balloons) a century and a half before the advent of modern airborne troops [Clan1997]:

> And where is the prince who can so afford to cover his country with troops for its defense, as that ten thousand men descending from the clouds might not in many places do an infinite deal of mischief before a force could be brought together to repel them?

With the coming of the wireless age, all packets will no longer have to go through a firewall to get into the corporate network. Wireless systems involve a third spatial dimension not seen in traditional networks. Attacks will not be limited to two dimensions only through wires, but will encompass three dimensions. Attackers can physically go over and around the firewalls to make the attack. For example, a small handheld computer "accidentally" left in a competitor's workplace may be able to breech more easily computers inside the "secure" firewall. It is indeed similar to adding airborne operations into cyberspace warfare. As in traditional warfare, one can come down from the sky (in three dimensional vertical envelopment) as opposed to only a two dimensional horizontal envelopment.

Wireless technology will change the way computer attacks are levied against persons, businesses, and sovereign countries. Security must become part of our lives, and it must be integral in programs that are written. It must be built on a foundation of rock, else it will be as Jesus said, "...like a foolish man who built his house on sand. The rain came down, the streams rose, and the winds blew and beat against that house, and it fell with a great crash." [Matthew 7:26b – 27 NIV]. See Section 2.6 for an overview of wireless networking and how wireless networking changes the paradigm of computing.

2.3 Computer Security Problems

So what's the problem with the current security technology? Section 2.2 talks about the assumptions designers have made in regard to the environment, source code, the application and the trusted computing base, the paradigms, and the assumptions themselves. Obviously, some of the assumptions need to be revamped.

2.3.1 The ARPANET Crashes in 1980, 1988, and Today

There have been a couple of ARPANET crashes in the past that have been nearly all-encompassing. The 1980 "modulo time stamp error" (Section 2.3.1.1) and the Internet Worm written by Robert Morris (Section 2.3.1.2) are covered in brief.

2.3.1.1 Modulo Time Stamp Error

On October 27, 1980, the ARPANET lost all connectivity for four hours. As described by Rosen [Rose1981a, Rose1981b][29] and summarized in [Neum1995], the entire ARPANET went down in the following way: the status messages sent back and forth between nodes were deleted if the time stamps were older (smaller) than a previous message. Due to a bit corruption in a memory node, three six-bit time stamps were each kept (none being deleted) because they each happened to be larger than their predecessor message (modulo 64).[30] Each of these three messages caused other messages to be sent over and over to other neighboring nodes who kept these sent messages. In summary, each node had to be manually shut down.

[29] These two articles appear to be identical.

[30] "For simplicity, 32 was considered a permissible difference, with the numerically larger time stamp being arbitrarily deemed the more recent in that case. In the situation that caused the collapse, the correct version of the time stamp was 44 [101100 in binary], whereas the bit-dropped versions had time stamps 40 [101000] and 8 [001000]. The garbage-collection algorithm noted that 44 was more recent than 30, which in turn was more recent than 8, which in turn was more recent than 44 (modulo 64). Thus all three versions of that status message had to be kept." [Neum1995].

2.3.1.2 The Internet Worm

Eight years later (November 2, 1988) when the Internet worm written by Robert T. Morris Jr.[31] was unleashed, some sites were disconnected from the Internet so that the extent of the damage could be assessed. There are many papers in the literature on the Internet worm. Those listed below are all printed in [Denn1990b]. They include an American Scientist article by Denning [Denn1989]; accounts from Spafford at Purdue [Spaf1989a, Spaf1989c]; Seely at Utah [Seel1989a, Seel1989b]; and Eichin and Rochlis of MIT [Roch1989]. The classic early paper on worms for beneficial use is in Shoch and Hupp's paper [Shoc1982]. Spafford comments on the break-ins and whether they are ethical and whether they should be condoned [Spaf1989b].

2.3.1.3 Crashes Today

If an error in present day routing software were to occur similar to these, the entire Internet could not be shut down by hand. As stated in Section 2.1.1.3, it has been stated that the Internet can be brought down by selected Denial of Service (DoS) attacks on key Internet backbone intersections. By splitting apart providers, connectivity can be cut. Even today, Distributed Denial of Service (DDoS) attacks have been leveled against specific sites; these DDoS attacks could be turned to the backbone routers as well.

2.3.2 Déjà Vu All Over Again

The primary problem in today's computer security technology is that we are seeing the same types of attacks that we have seen in the past. Bace and Shaefer in [Bace1995] review attacks that have happened in the past and have been "repackaged" in today's environments. Mudge and Benjamin in *Déjà Vu All Over Again* [Mudg1997] discuss specific attacks in Windows NT that were seen and solved in UNIX many years ago. What kinds of attacks are we seeing that we have seen before? Section A.1 describes specific examples of these and other computer attacks.

[31] Robert T[appan] Morris Jr. [Denn1990b] is the son of Robert T[appan] Morris. The father worked at AT&T Bell Laboratories and wrote [Morr1985a] along with other articles.

A way is needed to test the vulnerabilities of computer systems. Penetration testing, described in Section 2.4 uses a "penetrate and patch" technique such as the Flaw Hypothesis Methodology (FHM) (Section 2.4.1) and other methods to test a system by finding flaws.

2.3.3 Why Computers Are Not Secure

Gasser [Gass1988] discusses six reasons that computers are not secure. The first is that security is fundamentally difficult. From bugs in operating systems to the constant battle of the penetrators versus the penetrated (Section 2.4), there is no easy solution to the fundamental security problems. Secondly, security is an afterthought. After the functionality and cost of the system is designed, security is sometimes (but not too often) put in. Security cannot be put in as an afterthought. It must be designed fundamentally into the system because of the complex interactions between computer subsystems. Thirdly, most users see security measures as an impediment. Users try and get around any measures proposed by security administrators. An employee that wishes to get around the firewall may install a modem to their personal computer and dial directly out. However, with this modem installed, intruders can come back in through the modem. This is a very common penetration attack test. Complex firewall strategies can be compromised by just one user.

Fourthly, Gasser says that false solutions impede progress. Two examples he gives are call back modems and the continued reentry of passwords. Call back modems work by taking the call, hanging up the phone, and redialing the user back. This gives the users on the inside a false sense of security by thinking that only good users will call. Other security schemes such as password aging[32] are sometimes ignored. Fifthly, he states that people are the problems, not computers. People are the weakest link in computers, such as social engineering.[33] Finally, Gasser asserts that technology is oversold; he says that bad media coverage of past security research and development projects has hampered the security products out on the market. As examples, he cites research programs being touted as commercial products, vendors promising more than they deliver (case in point, Microsoft), and security products

[32]This is the procedure of making users change their passwords on a regular basis, such as every month.

[33]The art of pretending (lying) to be someone else in another position (technician, employee who has "lost his or her password," etc.).

not fully developed in the laboratory were mandated for some government projects in the field. When those projects in the laboratory did not pan out, the field product was discredited or not used.

2.4 Penetration Testing

How does one test for flaws that may lead to penetration and eventual control over the entire system? This was a question in the early 1970s that was studied by Clark Weissman [Weis1973] and Richard Linde at Systems Data Corporation (SDC) [Lind1975]. Years later after publication of his original SDC report in 1973 [Weis1973], Weissman wrote a summary article on penetration testing [Weis1995] which he reviewed his Flaw Hypothesis Methodology (FHM). This method has become a key means of testing a system, and it is described in Section 2.4.1. However, others propose similar but different methods. Gupta and Gligor [Gupt1991, Gupt1992], Hollingworth [Holl1974], and Carlstedt [Carl1975] are alternative methods to making penetration-resistant systems, which are described in Section 2.4.2.

2.4.1 Flaw Hypothesis Methodology

As stated before in Section 2.4, the Flaw Hypothesis Methodology (FHM) has been a key method for penetration testing. Section 2.4.1.1 discusses the theory of FHM with Weissman's and Linde's papers. Section 2.4.1.2 looks at the use of FHM in a real world operating system test.

2.4.1.1 Flaw Hypothesis Methodology Theory

Linde [Lind1975] outlines four steps of the Flaw Hypothesis Methodology (FHM) theory. Weissman [Weis1995] has a similar three-step method, and he leaves out the first and last step in Linde's method while adding a final step in the later work.[34] The three FHM theories are aligned together and outlined in Table 2.2.

[34]Both Linde and Weissman worked at System Data Corporation (SDC) In [Lind1975], the methodology is called the SDC Flaw Hypothesis Methodology.

Table 2.2: Linde vs. Weissman Flaw Hypothesis Methodology

Linde [Lind1975]	Weissman [Weis1973]	Weissman [Weis1995]
Knowledge of system control structure	N/A	N/A
Flaw hypothesis generation	Flaw hypothesis generation: The generic flaw	Flaw generation
Flaw confirmation	Flaw hypothesis confirmation	Flaw confirmation
Flaw generalization	Flaw generalization	Flaw generalization
N/A	N/A	Flaw elimination

2.4.1.1.1 Knowledge of System Control Structure Knowledge of system control structure involves studying all aspects of the operating system, including such items as the file system, the I/O systems, and the kernel. Although Weissman leaves this stage out of his listing in [Weis1995], he has background stages, the first of which lists what goals are to be accomplished (specific number of flaws to be found), a time limit of testing, etc. The second is to define the object to be tested (whether it is a stand alone system or on a network) and isolate the tested system from other production systems so that side effects of the testing are minimized. The third back-ground step is to "posture the penetrator" and figure out whether it is to be tested as an open box (white box) or a closed box (black box). A penetrator testing an open or white box has the ability to place random code on the system. A closed or black box system does not allow such benefits. The final background section is to "fix penetration analysis resources," meaning to determine what resources (people, tools, time, etc.) are needed and to make sure that the people who are doing the penetrations have the prerequisite knowledge of the system. This is analogous to Linde's first step in FHM.

2.4.1.1.2 Flaw Hypothesis Generation The first stage of Weissman corresponds to Linde's second stage and is known as flaw hypothesis generation, or flaw generation. This is where brainstorming and past experience come into play when making a list of potential flaws.

Linde lists the following as possible locations of flaws: historical generic system weaknesses; system prohibitions and warnings (timing dependencies); interfaces (man-man and man-system); seldom used or unusual functions or commands (read backward); control object dependency graph templates; historical attack strategies; systems listings, logic manuals,

and users' guides; and collection of user and system programmer experiences with the system under analysis. Linde's historical generic system weaknesses are the following: authentication; documentation; encryption; error detection; implementation; implicit trust; implied sharing; interprocess communication; legality checking; line disconnect; modularity; operator carelessness; parameter passing by reference vs. passing by value; passwords; penetrator entrapment; personnel inefficiency; privity; program confinement; prohibitions; residue; security design omissions; shielding; threshold values; use of test and set; and utilities. Linde's generic operating system attacks include: asynchronous, browsing, between lines, clandestine code, denial of access, error inducement, interacting synchronized processes, line disconnect, masquerade, "NAK"[35] attack, operator spoof, permutation programming, piecewise decomposition, piggy back, Trojan horse, unexpected operations, unexpected parameters, and wire tapping.

Weissman lists the "most productive 'top 10' generators" as the following:

1. past experience with flaws in other similar systems;
2. ambiguous, unclear architecture and design;
3. circumvention/bypass of "omniscient" security controls;
4. incomplete design of interfaces and implicit sharing;
5. deviations from the protection policy and model;
6. deviations from initial conditions and assumptions;
7. system anomalies and special precautions;
8. operational practices, prohibitions, and spoofs;
9. development environment, practices, and prohibitions; and
10. implementation errors.

[35] **N**ot **A**c**K**nowledged.

In these days, many automated tools have been written that may aid the process of finding flaws. Examples of such tools written in the past (listed in alphabetical order) include COPS, L0phtcrack, nmap, SATAN [Farm1995], and other scanners for both the host and the network.[36]

2.4.1.1.3 Flaw Hypothesis Confirmation This step involves confirming what potential flaws were found in the Flaw Generation step (Section 2.4.1.1.2). Weissman lists three substeps in this step: flaw prioritization and assignment; desk checking; and live testing. The first is to distinguish between the "high probability / high payoff (HH)" and the "low probability / low payoff (LL)." This will aid the penetrator in knowing which flaws to test first. Secondly, paperwork (code listings, maps, and models) is used to determine the probability of the flaw's occurrence. Thirdly, the system is put through a live test. The reason for doing desk checking at the expense of speed is that flaws may cause damage to the system under test.

2.4.1.1.4 Flaw Generalization The next step involves generalizing flaws into categories of flaws. This enables the administrators and the management of systems to know what classes of attacks are coming, and perhaps what attacks are more likely to come in the future; Weissman refers to this as "beading." Perhaps by knowing the nature of classes of attacks, one can search (as in Section 2.4.1.1.2) with even more detail to find flaws in vulnerable sections of the system under penetration testing. Section A.1 lists different types of attacks seen, and Section 2.5 reviews the literature that has classified attacks into taxonomies.

2.4.1.1.5 Flaw Elimination The final step in Weissman's FHM is flaw elimination. Once the flaws have been found, the flaws should be patched so that they cannot be exploited again. This step does not appear in Weissman's original report [Weis1973]. Other FHM penetration tests may need to be run [Holl1974], but this completes the FHM written by Weissman and Linde.

[36] Check the Internet for these and other more recent tools.

2.4.1.2 Flaw Hypothesis Methodology Application

FHM is a systematic approach to penetration testing. It is not a haphazard approach such as finding and patching each hole one at a time without a comprehensive plan. This is often referred to as "hack-and-patch" [Weis1995] or "penetrate-and-patch" [Land1993, Weis1995]. One must, "design security, quality, performance, and so on, into the system and not add it on [Sche1979]." [Weis1995].

Researchers at IBM used the FHM to study a VM/370 systems and found I/O to be the source of the most problems. Simplistic design was found to be the source of the most strength [Atta1976].

2.4.2 Other Penetration Resistant Methods

Hollingworth [Holl1974] wrote of a method similar to FHM and is outlined in Section 2.4.2.1. Gupta overviews a "theoretical foundation of penetration analysis" in [Gupt1991] and the use of their penetration analysis tool in [Gupt1992]. This is covered in Section 2.4.2.2.

2.4.2.1 Security Test and Evaluation Tools

Hollingworth [Holl1974] suggests that testing a system for penetration weaknesses involves, "(1) manual inspection of system documentation, (2) generation of penetration hypothesis based upon suspected security weaknesses, (3) hypothesis testing via the development of example penetration routines, (4) manual evaluation of the results, and (5) repetition of the preceding four steps as necessary to investigate unsuspected anomalies of system interaction and refine the penetration techniques." This is quite similar to Weissman's FHM technique. Hollingworth's steps are aligned to Linde's and Weissman's in Table 2.3; Weissman's 1973 FHM work is not presented here, but matched in Table 2.2 on page 31.

The report told of the desire for more tools that would aid in the investigators. Examples of aid in such tools would be in the help with the following: tracking parameters through and between modules, investigating the relationship between modules, and validating algorithms. They wanted tools to have some combinations of the following: "1. controlled program execution, 2. control/data flow mapping, 3. automated module exercising, 4. heuristic module

Table 2.3: Linde, Weissman, and Hollingworth Flaw Hypothesis Methodology

Linde [Lind1975]	Weissman [Weis1995]	Hollingworth [Holl1974]
Knowledge of system control structure	N/A	Manual inspection of system documentation
Flaw hypothesis generation	Flaw generation	Generation of penetration hypothesis
Flaw hypothesis confirmation	Flaw confirmation	Hypothesis testing
Flaw generalization	Flaw generalization	Manual evaluation of results
N/A	Flaw elimination	N/A

analysis."

2.4.2.2 Hypothesis of Penetration Patterns

Gupta and Gligor try to apply a more formalized approach to penetration testing, as opposed to the "ad-hoc manner" of FHM [Gupt1991]. They argue that there are a set of design principles that have been found to be consistently violated when penetrations occur. While they do not claim to have the exhaustive list, they present their "penetration-resistance properties:"

- System Isolation (or Tamperproofness)
- System Noncircumventability
- Consistency of System Global Variables and Objects
- Timing Consistency of Condition (Validation) Checks
- Elimination of Undesirable System/User Dependencies

These properties are presented in more detail in Sections 2.4.2.2.1, 2.4.2.2.2, 2.4.2.2.3, 2.4.2.2.4, and 2.4.2.2.5. Their fundamental theory is the "Hypothesis of Penetration Patterns" that says, "*system flaws that cause a large class of penetration patterns can be identified in system (i.e. TCB) source code as incorrect/absent condition checks or integrated flows that violate the intentions of the system designers*" [Gupt1991]. Their automatic tool for penetration testing that they developed is presented in more detail in [Gupt1992] along with more discussion of their Hypothesis of Penetration Patterns.

2.4.2.2.1 System Isolation (or Tamperproofness) Gupta and Gligor define system isolation as having:

1. Parameter checking at system interface,
2. User/system address space separation, and
3. System call selection and transfer of control.

Parameter checking makes sure the parameters are valid. The second item assures that users cannot directly access system space, and the final enumeration makes sure that the transfer of the system from unprivileged user mode to the privileged system mode is at designated control points only.

2.4.2.2.2 System Noncircumventability System noncircumventability makes sure all object references are checked, including reference to object contents, object-status variables, object privileges, and subjects [Gupt1991]. All references must pass through some check.

2.4.2.2.3 Consistency of System Global Variables and Objects Global variables are a necessary part of any operating system. They need to be consistent (have integrity) over time, and require that [Gupt1991]:

- a given global variable should not be alterable by unprivileged users,
- global tables that are alterable by unprivileged users should not overflow or underflow
- a given global table should never contain duplicate entries (e.g., disk-sector allocation entries),
- per-process and system wide resource limits must be enforced, etc.

2.4.2.2.4 Timing Consistency of Condition Checks This is similar to the above Section 2.4.2.2.3, but this refers to making sure that conditions (or validations) that existed before a system call is called is the same when the system call is actually performed. This is sometimes referred to as Time-Of-Check-to-Time-Of-Use (TOCTTOU) [McPh1974]. System checks to system call execution needs to be atomic.

2.4.2.2.5 Elimination of Undesirable System User Dependencies Finally, a user should not be able to cause a Denial of Service (DoS) attack by causing the system to execute a system function (like the UNIX call *panic()*). Other similar examples exist that are outlined in [Gupt1991].

2.4.3 Summary of Penetration Testing

Penetration testing has been the mainstay of finding flaws in computer systems to this day. Through FHM and other penetration methods, teams of users trying to find faults (called "tiger teams" or "red teams"[37]) penetrate systems and patch faults. "Penetrate and patch" is the common methodology of finding errors.

2.5 Computer Attack Taxonomies

Attempts have been made to categorize and classify computer attacks. Some have just listed categories of attacks, while others have formally developed taxonomies. Rushby [Rush1993] overviews a taxonomy of fault-tolerance, but it is not discussed further here. This section will outline the definition and requirements of a taxonomy. In addition, it will discuss the characteristics, features, and attributes of a taxonomy. Chapters 3 and 4 discuss in detail past work done in computer attack taxonomies.

Section 2.5.1 will overview what a taxonomy should contain, if it is to be considered a true taxonomy, and not just a categorization of attack classes. Section 2.5.2 describes the characteristics of a vulnerability in a taxonomy.

2.5.1 Definition and Requirements of a Taxonomy

In this section, a definition and the requirements of a taxonomy is reviewed. This author looked at the properties that past authors have argued need to be included in any security taxonomy. The properties of a taxonomy that are spoken of by John Howard [Howa1997],

[37]Probably named for the US military's use of the opposing force (OPFOR) colored red on maps, while the color of the US and allied forces were blue. It is unknown to the author if the red color came from the former Soviet Union's red flag. Friendly fire is known as fratricide or blue-on-blue.

Ulf Lindqvist and Erland Jonsson [Lind1997], Ivan Krsul [Krsu1998], and Edward Amoroso [Amor1994] are summarized.

Howard [Howa1997] asserts in his Ph.D. dissertation that any taxonomy must have a certain set of properties. Lindqvist [Lind1997] gives a similar list, only changing two categories, and Amoroso [Amor1994] adds a few more properties. Krsul [Krsu1998] and Bishop [Bish1999] give their own lists. Combining the set of properties, we obtain the following list:

- accepted [Howa1997]
- appropriateness [Amor1994]
- based on the code, environment, or other technical details [Bish1999]
- comprehensible [Lind1997]
- completeness [Amor1994]
- determinism [Krsu1998]
- exhaustive [Howa1997, Lind1997]
- internal versus external threats [Amor1994]
- mutually exclusive [Howa1997, Lind1997]
- objectivity [Krsu1998]
- primitive [Bish1999]
- repeatable [Howa1997, Krsu1998]
- similar vulnerabilities classified similarly [Bish1999]
- specific [Krsu1998]
- terminology complying with established security terminology [Lind1997]
- terms well defined [Bish1999]
- unambiguous [Howa1997, Lind1997]
- useful [Howa1997, Lind1997]

A taxonomy should be accepted in the general community, and it must be appropriate for the given assumptions — for example, whether malicious internal users are present or not. Every characteristic (see Section 2.5.2) or item being classified must be based on solid technical details, and not on "...social cause[s] of the vulnerability (malicious or simply erroneous, for example)" [Bish1999]. Some taxonomies presented in Chapter 3 base their classifications on non-technical characteristics, making those taxonomies non-conforming to Bishop's objectives.

The taxonomy must be comprehensible to both security experts and to those less familiar with security; It must be complete, so that every attack is able to fit somewhere in the taxonomy structure. Krsul argues that each characteristic must have a deterministic way to "extract" the feature. By being exhaustive, all possible categories are covered. Of course, each taxonomy could include a category "other" to make it exhaustive.[38] Amoroso writes that the internal and external threats must be able to be distinguished so that a security perimeter analysis can be run.

Each category must be mutually exclusive to each other category. That is, the categories must not overlap. There must be objectivity by the one determining the classification; Krsul said, "The features must be identified from the object known and not from the subject knowing" [Krsu1998]. That is, the characteristic must be "clearly observable" [Krsu1998].

When Bishop calls for a taxonomy to be "primitive", he refers to the choices that are made down a decision tree. Those choices should be able to be answered with a simple "yes" or "no" answer. This would cause the characteristic to be classified in the same way every time the classification is repeated by another party.

While Bishop suggests that all race conditions be classified together, he acknowledges that some race conditions may have other characteristics which may be classified in a different class. The multiple characteristics of a single vulnerability may cause the vulnerabilities to be overlapped into multiple classes. But by doing this, if a single characteristic can be eliminated, it is theorized that the vulnerabilities caused by that characteristic can also be eliminated.

The taxonomy should be specific, but should comply with the established security terminology. The terms of the taxonomy should be well defined (see Section 2.5.2 for more details); that is, "coding fault" and "environmental fault" may not be mutually exclusive or unambiguous. Finally, a taxonomy should be useful, but if one has an "other" category to make all categories mutually exclusive, the usefulness of the "other" category is debatable. There is always a need to be able to expand the taxonomy if new computer attacks come to light, but by putting an attack in "other," one breaks down the structure that the taxonomy seeks to define.

[38]Private e-mail message from Dr. Carl Landwehr, February 2000.

Some of the following taxonomies in Chapter 3 will not follow all of these properties. Whether these categorizations are considered true taxonomies by Howard, Lindqvist or Amoroso's definition will be discussed in a later section.

2.5.2 Characteristics, Features, and Attributes of a Taxonomy

Bishop [Bish1999] agrees with Krsul [Krsu1998] by stating that taxonomies should classify properties of vulnerabilities and not by the vulnerability itself. These **characteristics** are also called **features** or **attributes**. This is consistent with work done in the taxonomies of plants and animals in the past such as Linnaeus [Linn1766].

Bishop also argues that it is inappropriate to classify vulnerabilities by such terms as "coding fault" or "environmental fault" because, "...does a 'coding fault' arise from an improperly configured environment? One can argue that the program should have checked the environment, and therefore an 'environmental fault' is simply an alternative manifestation of a 'coding fault'" [Bish1999].

Krsul quotes numerous encyclopedias to state, "A **taxonomy** is the theoretical study of classification, including its bases, principles, procedures and rules" [Krsu1998]. "A **classification**," he continues, "is the separation or ordering of objects (or specimens) into classes." Later in his dissertation, he again says, "...a taxonomy includes the *theory* of classification, including the procedures that must be followed to create classifications, and the procedures that must be followed to assign objects to classes." As an example, he states that Aslam's [Asla1995, Asla1996] taxonomy, which Krsul himself extended, was not a taxonomy but a "classification scheme" because it did not explain the predictive properties of how the decision was to be made about each level or division. This dissertation presents methodologies of characteristics in Chapter 7.

Each level or division, Krsul derives from the biological sciences, must have a ***fundamentum divisionis***, or a "grounds for a distinction." That is, at each decision point whether to put a characteristic into one category or another, must have a feature that defines the difference between the two categories. He gives as an example that one cannot ask whether a vulnerability is a race condition, or a configuration problem. Since a vulnerability could be both, it would violate the principle of *fundamentum divisionis*.

2.6 Wireless Networking

The ability to access a network without wires is fast become a ubiquitous reality [Lewi1999]. While this section will overview wireless networking and common standards, security aspects of wireless technology will be covered in a later section.

Wireless communication [Rapp1996] is becoming more prevalent with each passing year. Companies such as Pitney Bowes use mobile data systems such as ARDIS, Mobitex, and CDPD (Cellular Digital Packet Data) to facilitate automatic part ordering from field units. In addition to stand-alone systems such as those, field personnel using laptop computers need to be connected to the Internet wherever they go [Daye1997]. Systems such as Mobile-IP [Perk1998, Solo1998] allow a mobile unit to attach itself to the Internet through a system of proxy agents. When a group of two or more mobile computers need to set up an ad-hoc network either at a small meeting or a large international conference, a Mobile Ad-Hoc Network (MANET) can be used.

Multiple access methodologies will be discussed in Section 2.6.1. Wireless communication standards will be covered in Section 2.6.2, while systems and protocols will be presented in Section 2.6.3.

2.6.1 Multiple Access Methodologies

There are many different ways to have multiple stations access the shared medium simultaneously. Different frequencies, time, and codes are used to achieve this result. These multiple access methodologies are discussed below as Frequency Division Multiple Access (FDMA) in Section 2.6.1.1, Time Division Multiple Access (TDMA) in Section 2.6.1.2, and Code Division Multiple Access (CDMA) in Section 2.6.1.3. For more detailed information on these multiple access technologies, consult [Rapp1996].

2.6.1.1 Frequency Division Multiple Access (FDMA)

Since all wireless technologies use different frequencies to transmit information, the frequency band allocated to the particular devices can be divided into channels to be used for uplinks and downlinks. An uplink is the process of sending information from the mobile unit to a

base station; while a downlink is the sending of information in the reverse direction, from a base station to the mobile unit. The band used to uplink information is known as the reverse channel; while the band used to downlink information is known as the forward channel.

One's car radio uses this technology to receive information. There are set frequencies (example: 87.9 MHz) that stations use as their base frequency to transmit their broadcast signal. Even though many radio stations are transmitting at the same time, one can listen to a particular station without interference from the other radio stations by tuning to the different frequencies.

2.6.1.2 Time Division Multiple Access (TDMA)

In another multiple access scheme, all data is transmitted using the same set of frequencies but at different times. The entire timeline is broken into fixed time slots that different users can transmit on. Unlike FDMA, multiple users share the same frequencies. Collisions can and do occur when two or more users transmit at the same time. For example, two people in the room talk at the same frequencies (400 - 4,000 Hz), but a meaningful conversation can occur only when they take turns and allow the other person to talk. A standard T1 (DS1) line uses this multiple access technology, as well as the classic Slotted ALOHA [Robe1975] (Slotted ALOHA is based on the ALOHA multiple access scheme described in [Abra1970]) [Bert1992].

2.6.1.3 Code Division Multiple Access (CDMA)

In CDMA, different users transmit their data on the same frequencies at the same time, but the data is "spread" over the entire frequency band with different "codes." The receiver who knows the code can reassemble the message and process the information. While this sounds less intuitive than FDMA and TDMA, it occurs in life. When a multitude of people get together at a party, many conversations usually occur between different sets of participants. Everyone uses the same voice frequencies at the same time, but the brain recognizes the voice ("code") of a person with whom one is talking.

2.6.2 Wireless Communication Standards

This section will outline some of the the basic wireless communication standards such as AMPS (Section 2.6.2.1), PCS (Section 2.6.2.2), 802.11 (Section 2.6.2.3), and Bluetooth (Section 2.6.2.4). Other protocols and standards will not be discussed further. For a brief outline of the history of cellular and the different types of standards presently available, see Chandran and Valenti's article [Chan2001]. For a more detailed description of many other wireless standards in North America, Europe, and Japan, see Varshney's article [Vars2000] and Rappaport's book [Rapp1996]. A recent article on cellular security is found in [Riez2000].

2.6.2.1 AMPS

The Advanced Mobile Phone System (AMPS) [Youn1979] was created in the late 1970s and was first deployed in 1983. It uses standard frequency modulation (FM) for a carrier of analog signals, and uses the frequency band 824–849 MHz for the reverse channel and 869–894 MHz for the forward channel [Rapp1996]. It is still in use today because of its seemingly ubiquitous towers and the long range it has compared with the newer digital systems. It uses FDMA (Section 2.6.1.1) with 30 kHz channel bandwidth. Other similar analog systems such as narrowband-AMPS (NAMPS) and European Total Access Communication System (ETACS) are covered in detail in [Rapp1996]. As will be shown in more detail later, AMPS is extremely unsecure, as one can easily intercept FM radio transmissions with a simple scanner.[39]

2.6.2.2 PCS

PCS, or the Personal Communication Systems [Ashi1993], seeks to incorporate aspects of an "advanced intelligent network (AIN)" [Rapp1996] to make communications (both voice *and* data) ubiquitous. Standards such as IS-95 (using CDMA) in North America and GSM (Global System for Mobile) (using TDMA) in Europe are becoming the means of implementing this vision. See Rappaport [Rapp1996] for more information on the specific protocols.

[39]Most common scanners on the market today disallow a user from listening on those frequencies of AMPS, but various work-arounds can be found on the Internet to modify the scanner's hardware. Usually, this involves just cutting a diode on the scanner's printed circuit board. This was once demonstrated in a Congressional hearing. The reader is directed to various "underground" sites to obtain further information.

2.6.2.3 802.11

IEEE 802.11 is a physical and data link layer protocol [Bert1992] implementing a Wireless Local Area Network (WLAN). It is a subset of the IEEE 802 family of protocols, such as 802.2 (Token Ring) and 802.3 (Ethernet). It offers wireless networking with 1–2 Mbps data transfer rate using Spread Spectrum or Infrared technologies [Loug1997] and 11 Mbps data transfer rate for IEEE 802.11b [O'Ha1999]. While some 802.11 products are on the market, it is still a relatively new technology. Further details of IEEE 802.11 standard along with a security assessment will be given in Chapter 8.

2.6.2.4 Bluetooth

Bluetooth, named after Harald Bluetooth, the 10th century Viking king,[40] is a consortium of companies (3Com, Ericsson, Intel, IBM, Lucent Technologies, Motorola, Nokia, and Toshiba) bonded together to form a wireless standard. Not only is it a standard, but it is also a product. The hardware consists of a microchip with a radio transceiver. It can be incorporated into a laptop or wireless phone. It can access other ad hoc networks (Section 2.6.3.2) or local access points. It is a short range system, operating at a normal range of 10 m (0 dBm) and an optional range of 100 m (+20 dBm). It is similar to IEEE 802.11 (Section 2.6.2.3) in that it uses 2.4 GHz as its base frequency. It can reach 6 Mb/s in a multiple piconet ad hoc structure. However, it differs from IEEE 802.11 in that it is a "Wireless Personal Area Network (WPAN) specified in IEEE 802.15, Working Group for Wireless Personal Area Networks:

> The 802.15 standards work is a cooperative effort with the Bluetooth SIG. The IEEE 802.15 Working Group provides, in the IEEE 802 family, standards for low-complexity and low-power consumption wireless connectivity. The cooperative effort resulted from a convergence of IEEE standards development activities underway coupled with the formation of the Bluetooth SIG in 1998.[41]

A recent article on Bluetooth is found in [Haar2000].

[40] http://www.bluetooth.com/bluetoothguide/faq/2.asp#

[41] Quote from web page found 27 November 2000 at http://standards.ieee.org/wireless/overview.html

2.6.2.5 Summary of Wireless Communication Standards

This section gives a brief overview of prevailing wireless communication standards today. Paulson's paper covers the differences between IEEE 802.11, HiperLAN, and HomeRF [Paul2000]. The latter two protocols will not be described further in this dissertation.

2.6.3 Systems and Protocols

This section will cover some of the systems and protocols that use the wireless standards above. The most popular systems are Mobile IP (Section 2.6.3.1) and Mobile Ad Hoc Network (MANET) (Section 2.6.3.2).

2.6.3.1 Mobile IP

Mobile IP is a protocol that attempts to give the aspect of mobility to the Internet Protocol (IP). It uses a set of agents (one on the home network called a *home agent* and one on the visited network called a *foreign agent*) to essentially act as a forwarding service. The mobile node registers with a foreign agent expressing the interest in receiving messages from the home network. The home agent intercepts all messages for the mobile system and forwards them to the notified foreign agent who then can reach the mobile node. Route optimization can be used to directly send messages from the mobile node directly back to the home agent, thus bypassing the foreign agent. This forms a triangle of the home agent, foreign agent, and the mobile node; cutting off one leg of the messages' journey. For detailed discussion of Mobile IP, see the two books [Perk1998, Solo1998].

2.6.3.2 Mobile Ad Hoc Networks (MANET)

When a fixed infrastructure is not available, a mobile ad hoc network (MANET) may be needed. This can be a network between two or more computer nodes and some users in the network may not be able to directly "see" other nodes. Thus, routing is accomplished by having some nodes be a mobile router. This causes many security concerns, as will be discussed in later sections. For an overview of MANET, see [Cors1999]; for a paper on how to secure mobile ad hoc networks, see [Zhou1999].

2.7 Summary

This chapter has presented an overview of computer security, assumptions made in designing security into computer systems, problems of computer security, the theory of penetration testing, and wireless networking. Chapter 3 will deal specifically with different types of attacks that have besieged computers in the past and those attacks that affect computers today.

Chapter 3

Computer Attack Taxonomies

Traditionally, security incidents were broken into categories of disclosure of confidential information (loss of confidentiality), loss of integrity, and denial-of-service (DoS) attacks (loss of availability) [Amor1994]. In chapter 3 of his book, Amoroso argues that if one breaks into a computer without disclosing any of its data (losing confidentiality), modifying any of its data (losing integrity) or causing a denial of service (losing service (or availability)), the attack would not fit into any of these neatly defined three categories. He asserts that some other form of taxonomies needed to be developed. The following sections examine different taxonomies that have been created and compare them to each other.

There has been work some done in classifying differing aspects of computer security, including Wagner's work in fingerprinting [Wagn1983], Syverson's taxonomy of replay attacks in cryptographic protocols [Syve1994], Hinke's taxonomy of inference detection approaches [Hink1997], and Linqvist's taxonomy of the security risks of using Commercial Off The Shelf (COTS) equipment [Lind1998]. This research will not concentrate on those taxonomies unless they deal specifically with computer attacks.

Some literature in the field outlines classes of weaknesses in computers [Atta1976] or general classes of attacks [Park1975b]. Some papers that do not consider themselves a taxonomy per se [Neum1989] are later extended and the entire taxonomy is considered a taxonomy [Lind1997].

This chapter will review the past work done in computer attack classifications and will

show that there are similarities among them by referencing past work in taxonomic comparison as well as new work developed in the comparison of taxonomies. Section 3.1 will show the overall types of computer misuse and its perpetrators. At the end of the section, the past work of misuse taxonomy comparison is combined with new work of misuse taxonomy comparison to show how all the misuse taxonomies are similar.

Section 3.2 outlines taxonomies that give an overview of the people trying to get malicious access to computers and the objectives and results of those trials. This encompasses the work done by Howard [Howa1997] and extensions done by Sandia National Laboratories [Chri1999].

Work done by graduate students at Purdue University (Kumar, Aslam, and Krsul) and extensions done by a graduate student at Iowa State (Richardson) is covered in Section 3.3. Lists used by Linde and Attanasio with the Flaw Hypothesis Methodology for penetration testing are given in Section 3.4. Finally, other miscellaneous taxonomies and attacks are outlined in Section 3.5.

3.1 Types of Computer Misuse and Its Perpetrators

This section will review those taxonomies given in the past that describe types of computer misuse and the perpetrators that do that misuse. The taxonomies of Anderson [Ande1980] and Neumann [Neum1989] are covered (Sections 3.1.1 and 3.1.2) as well as the extension of Neumann made by Lindqvist [Lind1997] (Section 3.1.3) and the network security taxonomy of Jayaram and Morse [Jaya1997] (Section 3.1.4). Those four taxonomies are then compared with each other (Section 3.1.5).

3.1.1 Anderson's Penetration Matrix

James P. Anderson in [Ande1980] develops a four cell matrix that covers the types of penetrators, based on whether they are authorized to use the computer and the data/program source. That matrix is shown in Table 3.1.

Anderson continues in his report by giving three subclasses of Class B, Internal Penetration: the masquerader, the legitimate user, and the clandestine user. As one progresses from

Table 3.1: Anderson's Penetrator Matrix [Ande1980]

	Penetrator Not Authorized to Use Data / Program Resource	**Penetrator Authorized to Use Data / Program Resource**
Penetrator Not Authorized Use of Computer	**CASE A:** External Penetrator	
Penetrator Authorized Use of Computer	**CASE B:** Internal Penetrator	**CASE C:** Misfeasance

the masquerader to the clandestine user, it is more difficult to detect with audit trails. The complete outline is seen in the following list:

A. External Penetration
B. Internal Penetration
    a. The masquerader (defeats procedural controls)
    b. The legitimate user
    c. The clandestine user (defeats logical controls)
C. Misfeasance (authorized action in an improper way [Neum1989])

The masquerader is someone pretending to be a legitimate user; from the system's perspective, there is no difference between a masquerader and a legitimate user if the masquerade works perfectly. The legitimate user is one of a "case of misfeasance... misuse of authorized access." A clandestine user, on the other hand, "operate[s] below audit trail or... evade[s] audit trail." There is nothing said about the third class Misfeasance, especially about what differentiates it between an Internal Penetration, legitimate user. However, as we see in Section 3.1.5.1, Neumann and Parker match their nine classes of computer misuse to the Anderson matrix.

3.1.2 SRI Computer Abuse Methods Model

Neumann and Parker developed over the course of six years a model that they call the SRI Computer Abuse Methods Model. Section 3.1.2.1 covers the evolution of the model through the four papers and books published in regard to the manual [Park1989, Neum1989,

Park1992, Neum1995]. Section 3.1.2.2 explains how the nine categories of the SRI Computer Abuse Methods Model was expanded into twenty-six types of attacks.

3.1.2.1 Evolution of the SRI Computer Abuse Methods Model

In 1989, Neumann and Parker published, "A Summary of Computer Misuse Techniques" [Neum1989] in which they outline a series of classes of computer misuse from their data of about 3000 cases over nearly twenty years. Figure 3.1 (page 51) shows their classes and their structure.[1] They comment, "For visual simplicity, the figure is approximated as a simple tree. However, it actually represents a system of descriptors, rather than a taxonomy in the usual sense, in that a given misuse may involve multiple techniques within several classes." [Neum1989, Neum1995]

Parker also published works in which he used the same basic text and described the tree. However, there are some changes in the trees throughout the literature. (See Table 3.2 on page 52 for a side-by-side comparison of the four instances in the literature of the model.)[2] The tree shown in [Park1989] has categories 6 ("Active Abuse") and 7 ("Passive Abuse") reversed. In addition, Parker leaves out the eighth category cited in [Neum1989], "Misuse Resulting from Inaction." In an article on computer crime contained in [Park1992], Parker leaves out the second category, "Hardware Misuse," even though he comments on it in the article (the same basic article text was used in all four references [Park1989, Neum1989, Park1992, Neum1995]). In addition, he changes the third category to "Preprogrammed Use" and continues to leave out the eighth category "Misuse Resulting from Inaction" as in his earlier article [Park1989]. Of all the references, the two Neumann references seem to be the most complete. It is this tree [Neum1989] that Lindqvist and Jonsson extend in [Lind1997] (See Section 3.1.3).

[1] It should be noted that the fourth class, "Setting up subsequent misuse" [Neum1989] had its name changed to "Pest programs for deferred misuse." Figure 3.1 is based on the later published tree in [Neum1995].

[2] Type in boldface indicates minor changes in the wording of the models; whereas, type in boldface and boxed are major changes mentioned here.

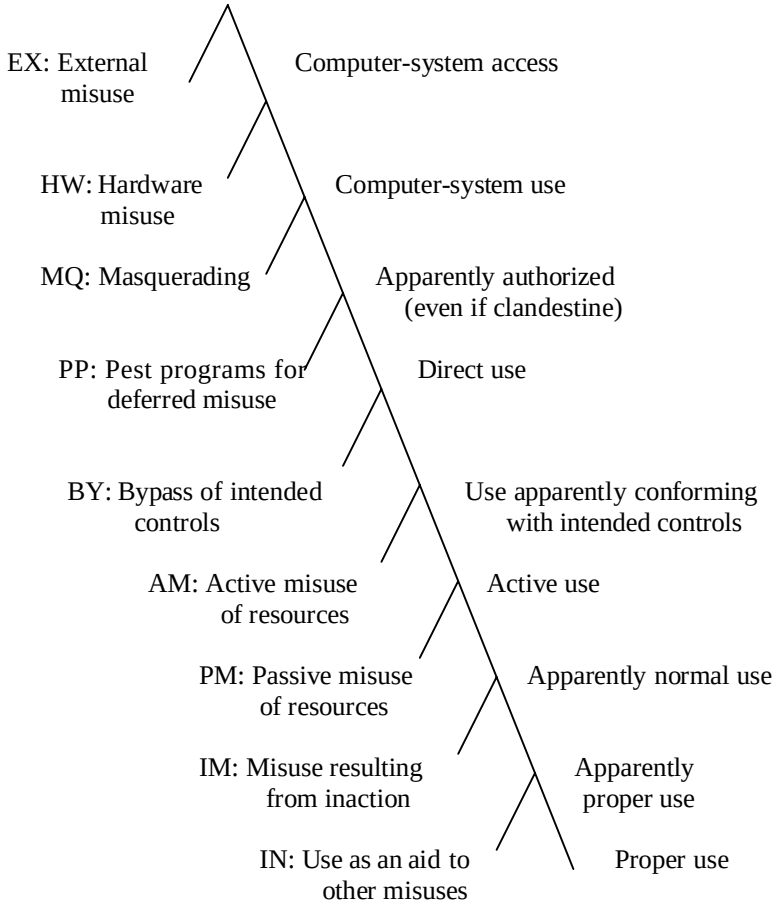

Figure 3.1: Neumann and Parker's SRI Computer Abuse Methods Model [Neum1995]

Table 3.2: Variations of the SRI Computer Abuse Methods Model

[Park1989] p.4	**[Park1992] p.442**	**[Neum1989] p.398**	**[Neum1995] p.101**
1. External misuse Computer system access	1. External abuse Internal	1. External misuse Computer system access	1. External misuse Computer-system access
2. Hardware misuse Computer system use	**N/A** **N/A**	2. Hardware misuse Computer system use	2. Hardware misuse Computer-system use
3. Masquerading **Operating system use**	2. Masquerade **Valid access**	3. Masquerading Apparently authorized use (even if clandestine)	3. Masquerading Apparently authorized use (even if clandestine)
4. Preparatory abuse Direct use	**3. Preprogrammed abuse** **Preprogrammed use**	**4. Setting up subsequent misuse** Direct use	**4. Pest programs for deferred misuse** Direct use
5. Bypass of intended controls Conforming with intended controls	4. Bypass of intended controls Conforming to intended controls	5. Bypassing intended controls Use apparently conforming with intended controls	5. Bypass of intended controls Use apparently conforming with intended controls
7. Passive abuse Passive use	5. Active abuse Active use	6. Active misuse of resources Active use	6. Active misuse of resources Active use
6. Active abuse Active use	6. Passive abuse Passive use	7. Passive misuse of resources Apparently normal use	7. Passive misuse of resources Apparently normal use
N/A **N/A**	**N/A** **N/A**	8. Misuse resulting from inaction Apparently proper use	8. Misuse resulting from inaction Apparently proper use
8. Use as a tool for committing abuse Normal use	7. Use as a tool for external use Normal use	9. Use as an aid to other misuses Proper use	9. Use as an aid to other misuses Proper use

Table 3.3: Neumann and Parker's Categories of Computer Misuse (NP1 – NP9)

NP1	**EXTERNAL**
NP2	**HARDWARE MISUSE**
NP3	**MASQUERADING**
NP4	**PEST PROGRAMS**
NP5	**BYPASSES**
NP6	**ACTIVE MISUSE**
NP7	**PASSIVE MISUSE**
NP8	**INACTIVE MISUSE**
NP9	**INDIRECT MISUSE**

3.1.2.2 Nine Categories and Twenty-Six Attacks

I will classify the nine categories that comprise the SRI Computer Abuse Methods Model as NP1 – NP9[3] as Lindqvist and Jonsson did in their 1997 expansion of this model. See Section 3.1.3 for a overview of Lindqvist and Jonsson's expansion of the SRI Computer Abuse Methods Model. These nine categories (NP1 – NP9) are the same categories as from the SRI Computer Abuse Methods Model shown in Figure 3.1 on page 51, and they are shown in Table 3.3.

Neumann expanded the nine categories of Neumann and Parker's categories of computer misuse into twenty six types of attacks, shown in Table 3.4 on page 54 [Neum1995]. The twenty-six types of computer misuse will be referred to as CM1 – CM26.[4] These nine categories and twenty-six attacks will be compared against other taxonomies in later sections of this document.

3.1.3 Lindqvist and Jonsson's Extension of Neumann and Parker

Lindqvist and Jonsson [Lind1997] extend Neumann and Parker [Neum1989] described in Section 3.1.2 by expanding categories NP5 (Bypassing intended controls), NP6 (Active misuse of resources), and NP7 (Passive misuse of resources). They introduce the concept of dimension: attacks have certain *intrusion techniques* and certain *intrusion results*. This forms the

[3]The prefix "NP" will stand for Neumann and Parker. Although not stated in Lindqvist and Jonsson's paper [Lind1997], the "NP" in their category names also probably means "Neumann and Parker."

[4]The prefix "CM" will be designated as Computer Misuse. For example, Logic bombs (seen in class NP4) will be known as CM14.

Table 3.4: Neumann and Parker's Types of Computer Misuse (CM1 – CM26) [Neum1995]

MODE	**MISUSE TYPE**
EXTERNAL (EX)	
1. Visual spying	Observing of keystrokes or screens
2. Misrepresentation	Deceiving operators and users
3. Physical scavenging	Dumpster-diving for printout
HARDWARE MISUSE (HW)	
4. Logical scavenging	Examining discarded / stolen media
5. Eavesdropping	Intercepting electronic or other data
6. Interference	Jamming, electronic or otherwise
7. Physical attack	Damaging or modifying equipment, power
8. Physical removal	Removing equipment and storage media
MASQUERADING (MQ)	
9. Impersonation	Using false identities external to computer systems
10. Piggybacking attacks	Usurping communication lines, workstations
11. Spoofing attacks	Using playback, creating bogus nodes and systems
12. Network weaving	Masking physical whereabouts or routing
PEST PROGRAMS (PP)	***Setting up opportunities for further misuse***
13. Trojan horse attacks	Implanting malicious code, sending letter bombs
14. Logic bombs	Setting time or event bombs (a form of Trojan horse)
15. Malevolent worms	Acquiring distributed resources
16. Virus attacks	Attaching to programs and replicating
BYPASSES (BY)	***Avoiding authentication and authority***
17. Trapdoor attacks	Utilizing existing flaws
18. Authorization attacks	Password cracking, hacking tokens
ACTIVE MISUSE (AM)	***Writing, using, with apparent authorization***
19. Basic active misuse	Creating, modifying, using, denying service, entering false or misleading data
20. Incremental attacks	Using salami attacks
21. Denials of service	Perpetrating saturation attacks
PASSIVE MISUSE (PM)	***Reading, with apparent authorization***
22. Browsing	Making random or selective searches
23. Inference, aggregation	Exploiting database inferences and traffic analysis
24. Covert channels	Exploiting covert channels or other data leakage
INACTIVE MISUSE (IM/25)	***Willfully failing to perform expected duties, or committing errors of omission***
INDIRECT MISUSE (IN/26)	***Preparing for subsequent misuses, as in off-line preencryptive matching, factoring large numbers to obtain private keys, autodialer scanning***

Table 3.5: Lindqvist and Jonsson's Intrusion Techniques

NP5 Bypassing intended controls	Password attacks	Capture
		Guessing
	Spoofing privileged programs	
	Utilizing weak authentication	
NP6 Active misuse of resources	Exploiting inadvertent write permission	
	Resource exhaustion	
NP7 Passive misuse of resources	Manual browsing	
	Automated searching	Using a personal tool
		Using a publicly available tool

basis of cause and effect in VERDICT (see Section 5.1.3). Their extensions of Neumann and Parker's categories NP5 – NP7, along with their intrusion techniques is given in Table 3.5, and their creation of a classification of intrusion results is given in Table 3.6 (p. 56).

It is to be noted that all the users were in a classroom setting, and since all the users were authorized to access the system, only classes NP5 – NP7 were considered. In a later section, we will show how the intrusion techniques of Lindqvist and Jonsson [Lind1997] are equivalent to some categories of the RISOS study [Abbo1976], the PA study [Bisb1978], and other work of Neumann [Neum1978].

3.1.4 Jayaram and Morse's Network Security Taxonomy

In [Jaya1997], Jayaram and Morse develop a taxonomy of security threats to networks. I shall label them JM1 – JM5.[5] They are shown in Table 3.7 on page 56.

Since Jayaram and Morse only briefly explain their taxonomy's categories (Class of Security Threats), I will quote them in full:

Physical Security breach arising from the theft of components and systems often perpetrated through impersonation.

System Weak Spots Weak spots in operating systems or other system software that a perpetrator exploits for unathorised (sic) access to the systems.

[5]The prefix "JM" will stand for Jayaram and Morse.

Table 3.6: Lindqvist and Jonsson's Intrusion Results

Exposure	Disclosure of confidential information	Only user information disclosed
		System (and user) information disclosed
	Service to unauthorized entries	Access as an ordinary user account
		Access as a special system account
		Access as client root
		Access as server root
Denial of service	Selective	Affects a single user at a time
		Affects a group of users
	Unselective	Affects all users of the system
	Transmitted	Affects users of other systems
Erroneous output	Selective	Affects a single user at a time
		Affects a group of users
	Unselective	Affects all users of the system
	Transmitted	Affects users of other systems

Table 3.7: Jayaram and Morse's Network Security Taxonomy (JM1 – JM5)

JM1	**Physical**
JM2	**System Weak Spots**
JM3	**Malign Programs**
JM4	**Access Rights**
JM5	**Communication-based**

Malign Programs A perpetrator embeds malevolent programs (viruses for example) in a system with the intention of causing destruction to information carried by the system.

Access Rights Legitimate user's identity usurped by 'acquiring' user's access rights through password traps, password cracking etc. to gain access to system resources.

Communication-based Network connectivity opens up opportunities for illegal information access through such means as eavesdropping, spoofing etc.

They also have two "Class of Security Mechanisms" that, "...are necessary to meet the class of security threats as outlined above" [Jaya1997]. The two listed are the following:

- Authentication and Access Control
- Encryption

Because "communication becomes the attractive means to effect intrusions into systems," they list two mechanisms to counter the communication threat:

- Firewalls
- Secure Socket Layer (SSL)

Their "Taxonomic View" [Jaya1997] covers many different levels of abstraction: system (e.g., JM1, JM2, JM5) to the program running on the system (e.g., JM3) to the result (or cause) of the breaches (e.g., JM4). It is included in this chapter as part of the various taxonomies out there, but it is not comprehensive enough for a final taxonomy.

3.1.5 Summary and Comparison of Misuse Taxonomies

This section will show that the four computer misuse taxonomies shown above can be combined and merged. Section 3.1.5.1 shows the comparison of how the Neumann and Parker nine categories (NP1 – NP9) [Neum1989] match the categories of Anderson [Ande1980].

Table 3.8: Comparison of Anderson and Neumann/Parker [Neum1989]

[Ande1980]		[Neum1989]
A. EXTERNAL PENETRATION	↔	Classes 1 & 2 (NP1, NP2)
B. INTERNAL PENETRATION		
a. The masquerader	↔	Class 3 (NP3)
b. The legitimate user		
c. The clandestine user	↔	Classes 4 & 5 (NP4, NP5)
C. MISFEASANCE	↔	Classes 6, 7, 8, & 9 (NP6, NP7, NP8, NP9)

3.1.5.1 Comparison of Neumann and Parker to Anderson's Penetration Matrix

Neumann and Parker compare their nine levels of descriptors to Anderson's matrix (Section 3.1.1). This is illustrated in Table 3.8. External Penetration includes External and Hardware misuse (NP1 & NP2). The masquerader of the internal penetration corresponds to Masquerading (NP3); while the clandestine user corresponds to Pest Programs and Bypasses (NP4 & NP5). Finally, the Misfeasance category of Anderson corresponds to the remaining Neumann and Parker categories, Active, Passive, inactive, and indirect misuse (NP6 – NP9).

We see that the Neumann and Parker [Neum1989] categories NP1 – NP9 are equivalent to Anderson's [Ande1980] categories. Continuing this trend, we shall look at the extensions of Neumann and Parker [Neum1989] by Lindqvist and Jonsson [Lind1997] and how those categories match up.

3.1.5.2 Comparison of Lindqvist and Jonsson to Neumann and Parker

It is shown in Table 3.6 on page 56 that Lindqvist and Jonsson [Lind1997] extend three categories of Neumann and Parker. Because of this, Lindqvist and Jonsson can also be compared to Anderson as shown by Table 3.8 on page 58 in Section 3.1.5.1.

3.1.5.3 Comparison of Jayaram and Morse's Taxonomy to Neumann and Parker

Table 3.9 on page 59 shows that Jayaram and Morse matches the categories of Neumann and Parker. Because of this, Jayaram and Morse can also be compared to Anderson as shown in Table 3.8.

Table 3.9: Comparison of Jayaram & Morse [Jaya1997] and Neumann & Parker [Neum1995]

JAYARAM & MORSE [JAYA1997]	Examples [Jaya1997]	NEUMANN & PARKER [NEUM1995]
PHYSICAL	Theft of Components, Impersonation	External (NP1); Hardware misuse (NP2)
SYSTEM WEAK SPOTS	Flaws in Operating Systems	Bypasses (NP5), Active misuse (NP6); Passive misuse (NP7)
MALIGN PROGRAMS	Viruses, etc.	Pest Programs (NP4), Active misuse (NP6); Passive misuse (NP7)
ACCESS RIGHTS	Password traps, Password cracking, "Acquiring" rights	Bypasses (NP5)
COMMUNICATION-BASED	Eavesdropping, Spoofing, Network attacks	Masquerading (NP3); Eavesdropping (CM5)
Not Used: Inactive misuse (NP8); Indirect misuse (NP9)		

3.2 From the Source to the Objective

This section will show two taxonomies that are very similar to each other. The first is given in Howard's Ph.D. dissertation [Howa1997] (Section 3.2.1). The second was included in a presentation at a conference; however, the presentation was not specifically about the taxonomy (Section 3.2.2).

3.2.1 Howard's CERT Taxonomy

John Howard [Howa1997] categorized the CERT incidents on the Internet from 1989–1995. He created a new taxonomy with types of attackers, tools used, access information such as why it was broken and what was used in the access, results of the break-in, and the objectives of the attack. Results are shown in Figure 3.2 on page 60. This taxonomy forms the basis for the Sandia Laboratory taxonomy shown in Section 3.2.2.

Howard's taxonomy, "...does not attempt to enumerate all computer security flaws, or to enumerate all possible methods of attack, but rather to reorient the focus of the taxonomy toward a process, rather than a single classification category" [Howa1997]. However, his

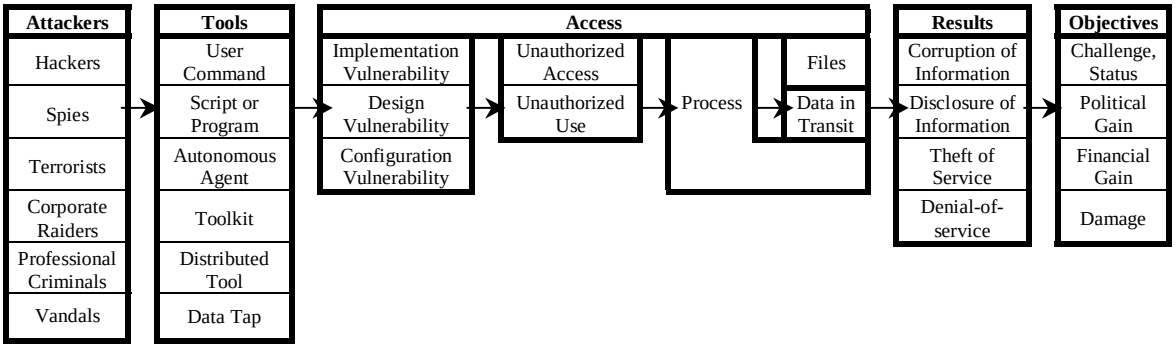

Figure 3.2: Howard's CERT Taxonomy

taxonomy seems to fail to follow his own ideas on what a taxonomy should consist of (Section 2.5.2). Howard wants taxonomies to be mutually exclusive, but his taxonomy fails to do just that. Under the category, *Tools*, both *Script or Program* and *Toolkit* cannot be mutually exclusive because, "Toolkits group scripts programs and autonomous agents together, often with a user-friendly graphical user interface" [Howa1997]. He tries to distinguish between toolkits and scripts by continuing, "What distinguishes toolkits from user commands, scripts or programs (the previous classifications) is that these are grouped together in a toolkit — a toolkit contains a group of tools." That distinction is not mutually exclusive, and the category '*Scripts or Program* is a subcategory of *Toolkit*.

Likewise, his "Attackers" category indeed lists many types of attackers, but they are not mutually exclusive. Depending on one's point of view, a '*terrorist's* actions could be indistinguishable from those of a *vandal*. A *spy* could be a *professional criminal*.

Howard's taxonomy, or the process of an attack, was taken and expanded in Sandia Laboratory's taxonomy, as seen in the next section (Section 3.2.2). However, as we will see, the same problems occur with that taxonomy that occur with Howard's.

3.2.2 Sandia Laboratory Taxonomy

This taxonomy was given in a talk by Christy in the fall of 1999 at the Shadowcon conference held in Dahlgren, VA [Chri1999]. Although his talk was on the cyber threat to the United States government and the legal issues that must be resolved in order to counter those threats, he included a taxonomy that looked very similar to Howard's [Howa1997] taxonomy. When

questioned, Christy said it had come from the Sandia National Laboratory. It is very similar to Howard's, adding some categories and merging a few others. It is presented in Figure 3.3 on page 62.

As was mentioned in 3.2.1, the same shortcomings of Howard's Ph.D. are evident in Sandia's taxonomy. Categories like "Attackers" are expanded, but are not mutually exclusive. While it provides an overview of types of attacks, it does not provide a complete taxonomy.

3.3 Purdue's Taxonomy Work and Extensions

Purdue University has produced a large amount of work on computer attack taxonomies. This section overviews that work. Sections 3.3.1, 3.3.2, 3.3.3, and 3.3.4 summarize the work of Kumar, Aslam, Krsul, and the extension work of Iowa State.

3.3.1 Kumar's Classification and Detection of Computer Intrusions

Kumar [Kuma1995] classifies computer intrusions on UNIX systems using the system logs and colored Petri nets. His "classification scheme" includes four types of signatures ("in increasing order of representability of signatures") [Kuma1995]:

1. Existence
2. Sequence (Interval and Duration)
3. RE[6] Patterns
4. Other patterns

In a 1994 conference paper, intrusion attacks are classified as follows [Kuma1994]:

1. **Existence.** The fact that something(s) ever existed is sufficient to detect the intrusion attempt. Simple existence can often be found by static scanning of the file system. Examples include searching for altered permissions or certain special files.

[6]Regular Expressions.

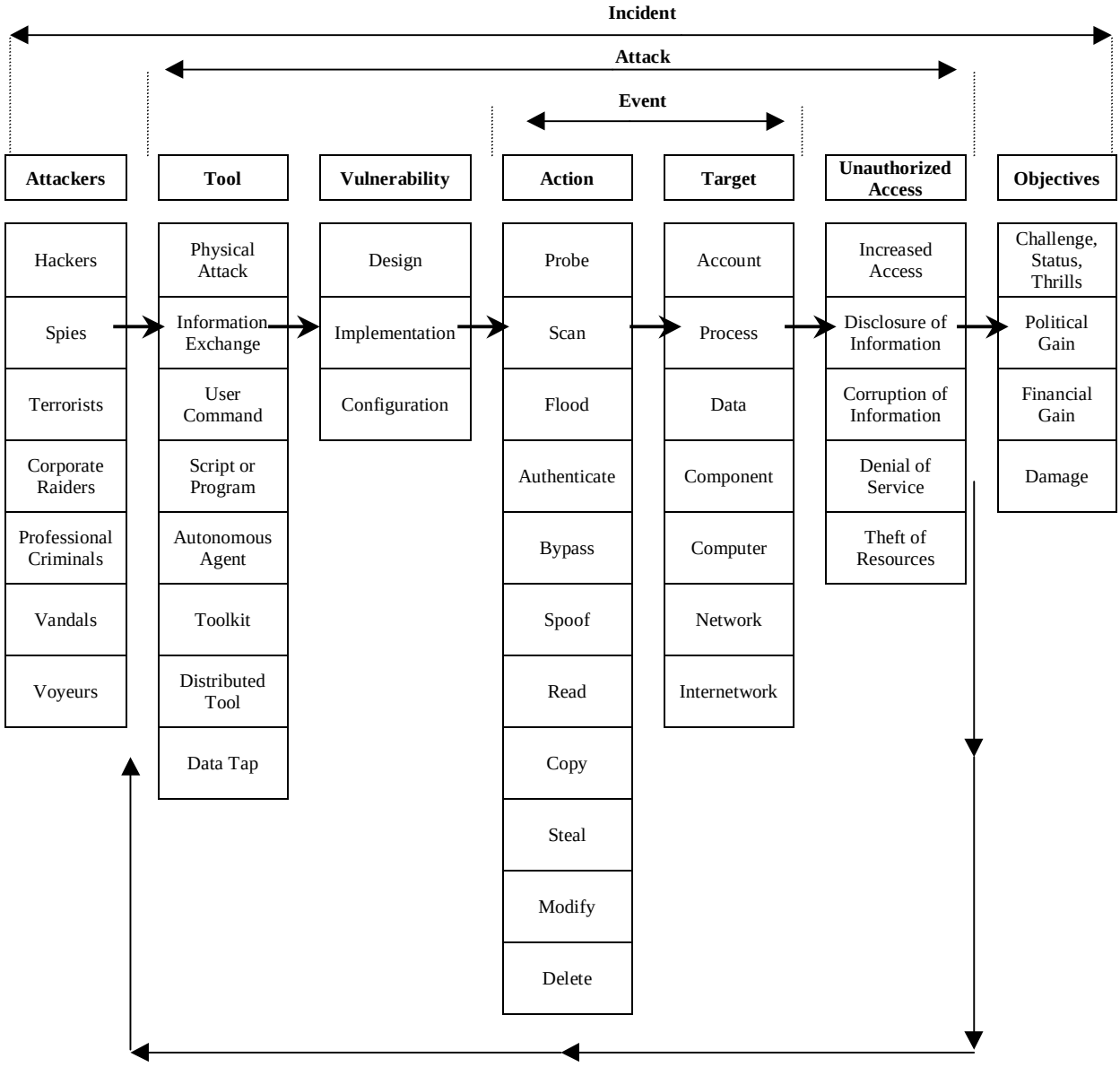

Figure 3.3: Sandia Laboratory Taxonomy

2. **Sequence.** the fact that several things happened in strict sequence is sufficient to specify the intrusion.

3. **Partial order.** Several events are defined in a partial order...

4. **Duration.** This requires that something(s) existed or happened for not more than nor less than a certain interval of time.

5. **Interval.** Things happened an exact (plus or minus clock accuracy) interval apart. This is specified by the conditions that an event occur no earlier and no later than x units of time after another event.

Lindqvist and Jonsson assert in [Lind1997] that because Kumar uses only the system logs to detect an intrusion, he cannot classify attacks that do not appear in the audit logs, such as passive sniffing. Part of Krsul's taxonomy [Krsu1998] includes a reference to IDIOT, a tool that Kumar's work helped create. Kumar's classification will not be further discussed in this dissertation.

3.3.2 Aslam's UNIX Security Taxonomy

This section will discuss Aslam's taxonomy and the criticisms given to it.

3.3.2.1 Aslam's Taxonomy

Aslam's Masters thesis [Asla1995] and his publication of his Master's work [Asla1996] outline a taxonomy of UNIX security flaws. The taxonomy is shown in Figure 3.4. One can compare it to other taxonomies (e.g., RISOS and PA, Sections 4.1 and 4.2), but it is not shown here.[7]

3.3.2.2 Bishop's Critical Analysis of Taxonomies

Bishop [Bish1996c] reviews PA [Bisb1978], RISOS [Abbo1976], and Aslam [Asla1995] and shows how the xterm log file flaw and the *fingerd* buffer overrun flaw cannot fit into any

[7]For examples, it is easy to see that Aslam's 3a) Condition validation error is similar to PA categories P2 and P9 (Validation); Aslam's 3b) is similar to P6 and P7 (Synchronization). See later sections for details on the taxonomies.

1) Operation faults (configuration error)
   1a) Object installed with incorrect permissions
   1b) Utility installed in the wrong place
   1c) Utility installed in incorrect setup parameters
2) Environmental fault
3) Coding fault
   3a) Condition validation error
      3a1) Failure to handle exceptions
      3a2) Input validation error
         3a2a) Field value correlation error
         3a2b) Syntax error
         3a2c) Type and number of input fields
         3a2d) Missing input
         3a2e) Extraneous input
      3a3) Origin validation error
      3a4) Access rights validation error
      3a5) Boundary condition error
   3b) Synchronization error
      3b1) Improper or inadequate serialization error
      3b2) Race condition error

Figure 3.4: Aslam's Taxonomy

category of any taxonomy. He claims that categories are not distinct or well-defined in a decision process. He challenges the readers with open research questions: can we build a new taxonomy with their (Aslam's, Krsul's, Landwehr's) questions (i.e., their decisions about the taxonomy); and can we use the taxonomy questions at a different levels? These questions are discussed in Chapter 5.

3.3.2.3 Krsul's Analysis of Aslam's Taxonomy

As will be discussed in Section 3.3.3, Krsul extended Aslam's taxonomy. He reworked the structure of the taxonomy, and the entire taxonomy is discussed next.

3.3.3 Krsul's Taxonomy

This section will present Ivan Krsul's taxonomy of computer attacks [Krsu1998]. This taxonomy is an extension of Aslam [Asla1995].

3.3.3.1 Krsul's Extension of Aslam

Krsul's Ph.D. [Krsu1998] and his proposal [Krsu1997] extended Aslam's database [Asla1995] and made a new taxonomy. It is quite complicated and detailed. The four main categories are:

- **Design**
- **Environmental assumptions**
- **Coding faults**
- **Configuration errors**

The entire taxonomy is presented in Figures 3.5 and 3.6 on pages 66 and 67.

3.3.3.2 Bishop's Vulnerabilities Classification Scheme

Bishop argues similarly to Krsul's presentation [Bish1996a] when he states we need to come up with a commonly agreed upon vocabulary. Bishop references an idea by Mike Raugh of Interconnect Technologies to create a thesaurus of vulnerability terms to be able to find these in independent organization of data. Characteristics are starting to come about in taxonomic literature.

3.3.4 Iowa State's Extension to Krsul

Richardson at Iowa State [Rich1999, Rich2001] develops a "Database of Vulnerabilities to Support the Study of Denial of Service Attacks." Denial of Service (DoS) attacks are the reason for this study; hence the taxonomy and solutions aim to solve the DoS problem. There are three categorizations that are used. Six hundred and thirty (630) attacks from web databases are cataloged in the following ways:

The first takes Aslam's taxonomy [Asla1995, Asla1996] and expands it to include the following three categories of attack: *specification weakness*,[8] *implementation weakness*,[9] and *brute force attacks*.[10]

[8]Specification weakness is derived from Aslam's category of "emergent faults" (e.g., improper installation) [Asla1996, Rich2001].

[9]Implementation weakness is derived from Aslam's category of "coding faults" (e.g., during development) [Asla1996, Rich2001].

[10]Brute force attacks is a new category added by Richardson.

(1) Design
(2) Environmental assumptions
  (2-1) Running program
    (2-1-1) Execution path
      (2-1-1-1) contains x
      (2-1-1-2) is at most x
      (2-1-1-3) is at least x
    (2-1-2) Name
      (2-1-2-1) is free of shell metacharacters
      (2-1-2-2) length of name is at most x
    (2-1-3) Environment
    (2-1-4) User running the program
  (2-2) User input
    (2-2-1) Content
      (2-2-1-1) is at most x
      (2-2-1-2) is at least x
      (2-2-1-3) matches regular expression
      (2-2-1-4) is free of shell metacharacters
      (2-2-1-5) is 7 bit ASCII
  (2-3) Environment variable
    (2-3-1) Name
    (2-3-2) Content
      (2-3-2-1) length is at most x
      (2-3-2-2) length is at least x
      (2-3-2-3) matches regular expression
      (2-3-2-4) is free of shell metacharacters
  (2-4) Network stream
    (2-4-1) Content
  (2-5) Command line parameter
    (2-5-1) Content
  (2-6) System library
    (2-6-1) Return
  (2-7) File
    (2-7-1) Name
    (2-7-2) Content
    (2-7-3) Owner
    (2-7-4) Permissions (Mode)
  (2-8) Directory
    (2-8-1) Name
    (2-8-2) (--- not assigned ---)
    (2-8-3) Owner
    (2-8-4) Permissions (Mode)
  (2-9) Program string
    (2-9-1) Content
  (2-10) Network IP packet
    (2-10-1) Source Address
    (2-10-2) Data Segment
    (2-10-3) Checksum
  (2-11) Directory, Running Program
    (2-11-1) Directory name, Running Program Privileges, Name of user that ran the program
  (2-12) File, Running Program
    (2-12-1) File permissions, Running program privileges, user that ran the program
    (2-12-2) File name, Running program privileges, User that ran the program
(3) Coding faults
(4) Configuration errors

Figure 3.5: Krsul's Taxonomy, Part I

(2-4-1) Content
  (2-4-1-1) is at most x
  (2-4-1-2) is at least x
  (2-4-1-3) is free of shell metacharacters
(2-5-1) Content
  (2-5-1-1) length is at most x
  (2-5-1-2) length is at least x
  (2-5-1-3) is 7 bit ASCII
(2-6-1) Return
  (2-6-1-1) length is at most x
  (2-6-1-2) length is at least x
  (2-6-1-3) is 7 bit ASCII
(2-7-1) Name
  (2-7-1-1) (--- not assigned ---)
  (2-7-1-2) is a valid file name
  (2-7-1-3) (--- not assigned ---)
  (2-7-1-4) is the same object as x
  (2-7-1-5) is final
(2-7-2) Content
  (2-7-2-1) length is at most x
  (2-7-2-2) length is at least x
  (2-7-2-3) is a known program
  (2-7-2-4) is a long file
  (2-7-2-5) is a known type
  (2-7-2-6) is 7 bit ASCII
  (2-7-2-7) matches regular expression x
(2-7-3) Owner
(2-7-4) Permissions (Mode)
(2-8-1) Name
(2-8-2) (--- not assigned ---)
(2-8-3) Owner
(2-8-4) Permissions (Mode)
(2-9-1) Content
  (2-9-1-1) length is at most x
  (2-9-1-2) is 7 bit ASCII
  (2-9-1-3) is free of shell metacharacters
  (2-9-1-4) is valid file name
  (2-9-1-5) matches regular expression x
  (2-9-1-6) is free of HTML tags
(2-10-1) Source Address
(2-10-2) Data Segment
  (2-10-2-1) Length is at least x
(2-10-3) Checksum
(2-11-1) Directory name, Running Program Privileges, Name of user that ran the program
  (2-11-1-1) is in valid user space for the user that invoked the program
  (2-11-1-2) user that invoked the program can read the directory
  (2-11-1-3) User that invoked the program can create files in the directory
  (2-11-1-4) user that invoked the program can write to files in the directory
(2-12-1) File permissions, Running program privileges, User that ran the program
  (2-12-1-1) User that invoked the program can read the file
  (2-12-1-2) User that invoked the program can write to the file
(2-12-2) File name, Running program privileges, User that ran the program
  (2-12-2-1) is a valid temporary file
  (2-12-2-2) is in valid user space for the user that invoked the program

Figure 3.6: Krsul's Taxonomy, Part II

The second takes Krsul's categories and adds a few new categories [Krsu1998]. The way of categorizing the error is found in a 4-tuple that can be expressed in the following sentence: "The [object affected] has been [effect on object] using [method or mechanism used] via [input type]" [Krsu1998, Rich2001].

The third method is one of mechanisms. There are six mechanisms developed, with the sixth mechanism consisting of four subcategories. The six mechanisms are the following:

1. Buffer overflows
2. IP Fragmentation attacks
3. Other *incorrect data* attacks
4. Overwhelm with Service requests
5. Overwhelm with data
6. Poor authentication or access control (broken down into 4 subcategories)[11]

The four categories of the sixth mechanism are the following:

6.1. Poor Authentication Scheme

6.2. IP Spoofing

6.3. Data Poisoning

6.4. Other misc.[12] protection shortcomings[13]

Richardson looks for clustering and develops countermeasures for these mostly DoS attacks. While the proposed set of solutions of DoS attacks seem good, there are some questions about the top level of the first taxonomic division. "Brute force" is a way of defeating a weakness, and that weakness can be either in the specification or the implementation. The three categories proposed by Richardson (specification weakness, implementation weakness, and brute force) are not on the same level of abstraction. For example, if a password scheme is weak and can be brute force attacked, is that a weakness of specification (one needs to

[11] [Rich2001], p. 82.
[12] misc. is an abbreviation in [Rich2001] for miscellaneous
[13] [Rich2001], p. 83.

have a stronger algorithm), a weakness of implementation (perhaps it is coded wrong as in "universal passwords" described in [Youn1987]), or both?

The second set of divisions using Krsul's categories are neither exhaustive nor mutually exclusive. The first of the 4-tuple is the "device / object affected". Two examples of a "device" are "stack data" and "user files". However, since most user files include stack data, the two are not mutually exclusive.[14] The other categories include the following: "object / effect on object,"[15] "method / method or mechanism used," [16] and "input type" [Krsu1998]/[Rich2001].[17]

In summary, although the Iowa State work deals with Denial of Service (DoS) attacks, its taxonomy is lacking in the same way as Krsul. The categories are not mutually exclusive, and thus do not constitute a correct taxonomy.

3.4 Flaw Hypothesis Methodology Penetration Lists

In papers dealing with the Flaw Hypothesis Methodology (FHM) (See Section 2.4.1), authors listed sections of the computer or operating system that were prime candidates for potential compromise. Two such papers, Linde [Lind1975] (Section 3.4.1) and Attanasio [Atta1976] (Section 3.4.2) give generous examples. Although some categories may match previously listed categories in the PA [Bisb1978] or RISOS [Abbo1976] reports, they are not matched with their respective PA or RISOS categories but are given only for future thought.

[14]The following is a list of "device / object affected" by Richardson. (Those objects with a ISSL prefix are those items added by ISSL where Richardson attended, Iowa State University's Information Systems Security Laboratory, http://www.issl.org): CPU: CPU time, OS: Operating system, Netport: network port, Packets: network packets, User files, System files, System names, User program, System info, Shell command, Password, Stack code, Stack data, Stack return, Static data, Public files, System program, Outfiles, Directory, Partition, Heap code, Heap data, Webpages, Websession, Email, Names, A_ net_ connections, Issl_ net_ services.[Rich2001], p. 81.

[15]Richardson includes the following: crash(ed), exhausted, bound, exported, mounted, closed, terminated, executed, replaced, changed, read, appended, created, displayed, predictable, changed_ owner, changed_ permission, loaded, presented, debugged, locked, cleartexted, and not_ logged [Rich2001], p. 81.

[16]Richardson has these listed as the following: ISSL_ brute_ Force, incprot (auth/permission), ISSL-incorr_ imp_ error (fragmented/offset), proxy, incorr_ imp (environment), special characters, dot_ dot (/../../), configuration error, inappropriate capability, inherit_ priveledges (sic), mod_ name, back ticks (\\), hidden_ mount, verify_ fail, modifying_ environment, relative_ paths, system_ call, infinite_ loop, core_ dump, incprot (cgi-bin), and ISSL_ improper_ data (buffer overflows & other wrong data) [Rich2001], p. 81.

[17]Richardson has these listed as Netdata and Store [Rich2001], p. 81.

3.4.1 Linde's Generic System Functional Flaws

Linde [Lind1975] lists six classes to study for penetration results in the Flaw Hypothesis Model (FHM):

- I/O Control
- Program and Data Sharing
- Access Control
- Installation management / operational control
- Auditing and surveillance
- Non-software weaknesses

In addition, he lists in Appendix A of his paper generic system flaws that were used in penetration testing. They are the following:

- Authentication
- Documentation
- Encryption
- Error Detection
- Implementation
- Implicit Trust
- Implied Sharing
- Interprocess Communication
- Legality Checking
- Line Disconnect
- Modularity
- Operator Carelessness
- Parameter Passing by Reference vs. Passing by Value
- Passwords
- Penetrator Entrapment
- Personnel Inefficiency
- Privity
- Program Confinement
- Prohibitions

- Residue[18]
- Security Design Omissions
- Shielding
- Threshold Values
- Use of Test and Set
- Utilities

Linde continues in Appendix listing generic operating system attacks:

- Asynchronous
- Browsing
- Between Lines
- Clandestine Code
- Denial of Access
- Error Inducement
- Interacting Synchronized Processes
- Line Disconnect
- Masquerade
- "NAK" Attack
- Operator Spoof
- Permutation Programming
- Piecewise decomposition
- Piggy Back
- Trojan Horse
- Unexpected Operations
- Unexpected Parameters
- Wire Tapping

[18]There appears to be an error in [Lind1975] because the bullet after Residue and before Security Design Omissions is "Magnetic tape, disc space, and core residue can often be easily read and searched for sensitive information; temporary files and buffers are the most common sources." This sentence appears to be Linde's last sentence for Residue. In addition, all of his attacks are listed in alphabetical order. From these observations, I can probably safely say that this "Magnetic tape..." bullet is a printing error.

3.4.2 Attanasio's FHM Penetration Characteristics

In 1976, Attanasio [Atta1976] lists sections and characteristics of the computer that they used with the Flaw Hypothesis Methodology (FHM) to try and find weaknesses. They are the following:

- Implicit or explicit resource sharing mechanisms.
- Man-machine interfaces administered by the operating system.
- Configuration management controls.
- Identity-authentication controls.
- Add-on features, design modifications, and design extensions.
- Parameter checking.
- Control of security descriptors.
- Error handling.
- Side effects.
- Parallelism.
- Access to microprogramming.
- Complex interfaces.
- Duplication of function.
- Limits and prohibitions.
- Access to residual information.
- Violation of design principles.

While more of a listing of possible sources of errors and not a taxonomy per se, it does show areas where a computer may be attacked. In addition, it lists *characteristics* of a successful attack.

3.5 Other Taxonomies and Attacks

This section will cover taxonomies and other attack lists that were not covered in previous sections. They include matchings of Brinkley [Brin1995] (Section 3.5.1) and Knight [Knig2000] (Section 3.5.2). In addition, the taxonomies of Beizer [Beiz1990] (Section 3.5.3); SRI Security Breaching Incidents (Section 3.5.4); Perry and Wallich [Perr1984] (Section

3.5.5); Dunnigan [Dunn1995] (Section 3.5.6); Parker [Park1975b, Park1989, Park1992] (Section subsec:Parker's Taxonomies); Straub and Widom (Section 3.5.8); and Ristenbatt (Section 3.5.9) are outlined.

3.5.1 Brinkley's Computer Misuse Techniques

Brinkley and Schell [Brin1995] list four areas of computer misuse:

1. **Theft of computational resources**
2. **Disruption of computational resources**
3. **Unauthorized disclosure of information**
4. **Unauthorized modification of information in a computer**

They explain that the first two (dealing with resources) are much different than the latter two (dealing with information itself). Defense against theft and disruption of resources can be dealt with by having physical controls (separate computer room, physical logging of users, etc.) and passwords.

Defense against disclosure and modification of information is done with different means. Therefore, they then take the last two categories and expand them into six classes:

1. **Human error**
2. **User abuse of authority**
3. **Direct probing**
4. **Probing with malicious software**
5. **Direct penetration**
6. **Subversion of security mechanism**

Human error encompasses all errors which cannot be controlled by the attacker. Mistakes happen, and sometimes, unauthorized disclosure or modification of information results. User abuse of authority is when a person in authority abuses their privileges and allows disclosure or modification of data.

Brinkley and Schell distinguish between direct probing and probing with malicious software. The former, they argue, arises when users just "test" the system to see what they can get away with. Everything that the users try is "allowed," but may result in higher privileged account access. In the latter, software specifically designed to break the system

is used. Malicious intent is evident, and can be manifest in Trojan horses,[19] viruses, time bombs, logic bombs,[20] or worms[21] [Denn1990b].

Direct penetration is the "bypassing of indented security controls" [Brin1995]. It may use probing with malicious software in trying to achieve this goal. From this revelation, these classes do not make a legitimate taxonomy, since the classes are not mutually exclusive; however, the authors probably did not mean to make a total "taxonomy" of attacks. Finally, subversion of security mechanism, "involves the covert and methodical undermining of internal system controls to allow unauthorized and undetected access to information within the computer system" [Brin1995]. One student of Roger R. Schell's, Philip A. Myers wrote his M.S. thesis on this point, "Subversion: The Neglected Aspect of Computer Security" [Myer1980].

I have matched the six qualities of Brinkley and Schell's taxonomy with Computer Misuse (CM) or Neumann and Parker (NP) categories. The results are presented in Table 3.10 on page 75.

[19]The term "Trojan horse" is attributed in a footnote on page 1303 of [Salt1975] to D. Edwards with a reference to Branstad [Bran1973]. In Branstad's paper, James Anderson and Daniel Edwards discuss generic weakness in operating systems and, "A program which executes a desired function correctly, but has illegitimate side effects was coined a "Trojan Horse." As a side note, D. Edwards is referenced as "Daniels Edwards" and "Dan Edwards"; the first ("Daniels") may be a typographical error for "Daniel."

[20]A historical side note: Donn Parker claims to have invented this term ["The Trojan Horse Virus and Other Crimoids" [Denn1990b], pp. 544, 551]:

> A "crimoid" is an elegant, intellectually interesting method of computer abuse that receives extensive coverage in the news media.... If an appealing name is a contributing factor [for "converting a computer misuse into a crimoid"], it would be useful for responsible professionals in information technology to use less attractive naming conventions for computer misuse. This seems to be commonly done with good effect in the criminal justice community where "rip off" is avoided in favor of "larceny" and "theft." I am as guilty as anyone in this regard, having coined the terms "data diddling," "logic bomb," and "crimoid." It may be useful for use to resolve to avoid "cute" terms in the future.

[21]In addition to worms and viruses, Waltz [Walt1998] and Ahuja [Ahuj1996] add "bacteria" to the mix. While worms are self-replicating agents that travel through the network and viruses attach themselves to programs, bacteria are:

> *Bacteria* — A bacterium is an independent, self-replicating agent program that creates many versions of itself on a single machine, and as the multiplying versions are executed, increased storage space and processing time are acquired. Its geometric growth and resource capture properties enable it to deny service to legitimate users. Unlike a virus, bacteria programs do not attach to a host program. [Walt1998]

Table 3.10: Comparison of Brinkley and Schell with Other Taxonomies

Brinkley and Schell [Brin1995]	**Matchings**
Human Error	***N/A***
User Abuse of Authority	**CM20** (Incremental attacks (Salami attacks)
Direct Probing	**CM22** (Browsing)
Probing with Malicious Software	**NP4** (Pest programs)
Direct Penetration	**NP5** (Bypasses) "Bypassing of intended security controls"
Subversion of Security Mechanisms	**CM17** (Trapdoor attacks) [Myer1980]

3.5.2 Knight's Vulnerability Taxonomy

Knight has recently produced a draft version of his taxonomy [Knig2000]. It is drawn and its taxonomic levels are matched with previous taxonomic work; it is shown in Table 3.11.

Most categories match other previous taxonomies. He defines a vulnerability as having five parts:

- **Fault** *(How it came to be; based on Aslam/Krsul/Spafford)*
- **Severity** *(What degree of compromise)*
- **Authentication** *(Does intruder have to successfully register?)*
- **Tactic** *(Who exploits who based on location)*
- **Consequence** *(Outcome)*

Knight defines the *fault* as based on the Aslam/Krsul/Spafford [Asla1996] taxonomy, and he further divides *severity* into six levels of how high of level is gained in an attack:

- Administrator access
- Read restricted
- Regular user access
- Spoofing
- Non-detectability *(logging system disabled so nothing recorded)*
- DoS (Denial of Service)

Authentication is whether the user has to be authenticated by the system in order to launch the attack. Tactic is divided into five categories:

- Internal Tactic
- Physical Access Tactic
- Server Tactic

Table 3.11: Knight's Taxonomy and Comparison [Knig2000]

	Affects Person	Affects Computer
Instantaneous	**Social Engineering**	**Logic Error**
Requires a duration of time	**Policy Oversight**	**Weakness**

Vulnerability Type [Knig2000]	Examples	Other Taxonomy Comparisons	
LOGIC ERROR	Application Specific	**NP3 – NP9** (Masquerading, Pest programs, Bypasses, Active misuse, Passive misuse, Inactive misuse, Indirect misuse)	
	Operating System		
	Network Protocol Design		
	Forced Trust Violations		
WEAKNESS	Eavesdropping	**CM5** (Eavesdropping)	
	Weak Passwords	**CM18** (Authorization attacks)	
	Custom Obscure Security	***N/A***	
	Encryption	**NP9** (Indirect misuse)[1]	
SOCIAL ENGINEERING	Theft	**CM8**[2]	**NP2**[4]
	Sabotage	**CM7**[3]	
	Internal Spying	**CM1**[5]	**NP1**[7]
	Information Fishing	**CM3**[6]	
POLICY OVERSIGHT	Physical Protection Policy	***N/A***	
	Data Protection Policy		
	Personnel Protection Policy		
	Information Divulgence Policy		

1: Offline password cracking
2: Physical removal
3: Physical attack
4: Hardware misuse
5: Visual spying
6: Physical scavenging
7: External

- Client Tactic
- Man-in-the-Middle Tactic

Besides defining vulnerability in a detailed way, he maps vulnerability on the time required to exploit — instantaneous to years. An example of instantaneous would be a logic bug or a theft of a document. An example of years would be the brute force of cracking. I have shown the matching of some of his examples, but his four main categories of **Social Engineering**, **Logic Error**, **Policy Oversight**, and **Weakness** form the core of his taxonomy.

3.5.3 Beizer's Bug Taxonomy

Beizer included a large and detailed taxonomy of software bugs in his book [Beiz1990]. They seem to be divided into three types of bugs: those in the **design** phase, the **implementation/coding** phase, and the **maintenance** phase. He enumerates his taxonomy by a 4 digit number, with each number representing a level of the taxonomy; an 'x' represents a global flag (all encompassing) for that level. The top level is outlined in Table 3.12 on page 78.

3.5.4 SRI Security Breaching Incidents

In 1976, the Stanford Research Institute (SRI) collected 355 security breaching incidents,[22] and divided them into seven violation categories: Intentional violations — internal; Intentional violations — computer department; Intentional violations — external; Aura of computer; Disaster; Accidents; and Miscellaneous. Ninety percent of the violations are intentional violations, i.e., listed in the first three categories.

The security breaching incidents in the first three categories: Intentional violations — internal; Intentional violations — computer department; and Intentional violations — external), were *each* subdivided into the same seven major subcategories. The major subcategories are the following: Physical access; System access; Data manipulation — external; Data manipulation — internal; Misapplication of services; Unprotected activities; and Physical theft.

[22] After all the cases had been reviewed, 64 cases were discarded mostly because, "...the available information was so sparse or so vague as to make the meaningful identification of applicable safeguards impossible" [Niel1976]. Hence, 291 cases were classified.

Table 3.12: Beizer's Bug Taxonomy

Numeric Designator	Level	Phase of Genesis
1xxx	Requirements	Design
2xxx	Features and Functionality	
3xxx	Structural Bugs	Implementation / Coding
4xxx	Data	
5xxx	Implementation and Coding	
6xxx	Integration	
7xxx	System, Software, and Architecture	
8xxx	Test Definition and Execution	Maintenance
9xxx	Other, unspecified	

Almost all of the major subcategories (with the exception of System Access) were *each* divided yet again into the same minor subcategories. Finally, some of the minor subcategories were yet further divided into "sub-subcategories."[23]

Seventy-one total "violation" categories were created. Enumerating the categories, major subcategories, and minor subcategories reveals the following:

- If one of the seven categories had no subcategories (such as Miscellaneous), that category was counted as one of the seventy-one "violation" categories.
- If one of the seven categories had major subcategories,[24] the category having major subcategories was not counted as a "violation," but instead each of the major subcategories were.
- Continuing down the levels, if a major subcategory had minor subcategories (all did but System Access), the major subcategory having minor subcategories was not counted as a "violation," but each of the minor subcategories was.

Fifty-one percent of the total 355 came from data manipulation, both internal and external.[25] The list of protection mechanisms (or safeguards) is arranged in categories and subcategories such as System Logging Functions, Encryption, Storage and Backup, and Access Controls. These safeguard categories are standard practice today [Niel1976].

3.5.5 Perry and Wallich: An Attack Matrix

In their 1984 paper, *Can Computer Crime be Stopped?* [Perr1984], the authors state that in 1964 (at the time of the paper, twenty years ago), mainframes were the form of computing. Computer crime, "...fell into one of the four categories:

1. **physical destruction** (ruining the computer or the tapes);

[23]The designation of "sub-subcategories" is not listed in Nielsen et al.'s report but was coined to designate yet further divisions of subcategories.

[24]The first three categories all did: Intentional Violations — Internal; Intentional Violations — Computer Department; and Intentional Violations — External.

[25]Twenty-four percent of the cases were classified as "Data manipulation — external"; while twenty-seven percent of the cases were classified as "Data manipulation — internal." The sum of 24% and 27% is the cited 51%.

2. **information destruction** (erasing tapes);
3. **data diddling** (changing data input or the data in the computer...); and
4. **unauthorized use of computer services**. [Perr1984]"

In 1974, (at the time of the paper, ten years ago) timesharing allowed the "theft of service," that is, using the computer without paying for the time. A matrix was introduced with six types of computer crime, six types of people using the computer, what damage could be done, and what could be done to stop it. Amoroso redraws, slightly simplifies, and leaves out the defenses of the matrix in his book [Amor1994]; however, the original Perry and Wallich matrix is shown in Table 3.13. Listings above the dotted lines show how the action (physical destruction, information destruction, etc.) can be accomplished by the attacker (operators, programmers, etc.).

3.5.6 Dunnigan and Cohen: Deception and Security

While not dealing with computer attacks, Dunnigan and Nofi [Dunn1995] classify deception techniques (of traditional warfare) into nine categories. In 1996, Fred Cohen [Cohe1996] suggests how each of these techniques can be used with a computer in information warfare (Section 2.1.2), in particular, IP address forgery:

- **Concealment**: hiding forces using "natural cover, obstacles, or simply great distance." [Dunn1995] Cohen states DoS attacks use IP address forgery as concealment of identity.

- **Camouflage**: hiding forces using artificial means (tree branches, etc.) By making an attack on a business look like an attack from a University instead of a competitor, Cohen likens this to Dunnigan's camouflage.

- **False and Planted Information**: giving harmful information to the enemy (but helpful to you); misinformation. Cohen states, "IP address forgery can be used to create the impression that a particular site is acting maliciously in order to create friction or lead a defender to falsely accuse an innocent third party." [Cohe1996]. This example is very similar to his example for camouflage; however, in warfare the difference between camouflage and false and planted information are very different. In wireless

Table 3.13: Perry and Wallich Attack Matrix [Perr1984]

	Operators	**Programmers**	**Data entry**	**Internal users**	**Outside users**	**Intruders**
Physical destruction	Bombings & shootings; Short circuits; Defenestration Screen operators; Reduce access					
Information destruction	Direct action (erasing disks) Screen operators; Backup data	Direct action; Trojan horse programs Reduce access; Audit program code			Direct action; Trojan horse programs Prevent access to information, programs	Via modem and network Prevent log-in
Data diddling		Trojan horse programs Audit program code	Direct action (false data) Separate functions; Check data consistency			
Theft of services		Same opportunities any user has Audit computer use		Direct action without authorization Audit computer use	Via a modem Audit computer use	
Browsing				Direct access without authorization Institute data-access controls	Via a modem Control file access	
Theft of information	Physical theft of tapes, disks, printouts Control access			Direct access Institute data-access controls	Via a modem Control file access	

information attacks, signals may be able to be camouflaged in noise while fictitious log files may be planted to convince the target that a different attack occurred.

- **Ruses**: using "enemy equipment or procedures to deceive" (using enemy uniforms, frequency bands, etc.) [Dunn1995] If a security vendor launches attacks at a client to convince the client that more security measures are needed, the attacks themselves would be considered ruses [Cohe1996].

- **Displays**: making the enemy see what is not there (such as logs painted like artillery).[26] One can make a display with IP address forgery by giving the impression that many sites are attacking the target instead of one [Cohe1996].

- **Demonstrations**: intentionally showing off some aspect of forces to confuse enemy as to what really is going on; imply an action but that action may not be carried through (such as moving naval ships into a contested area to "show force"). "IP address forgery has been used to demonstrate a potential for untraceable attacks as a way to convince defenders not to try to catch attackers." [Cohe1996]

- **Feints**: a demonstration carried through and in an attempt to hide the main attack, some distance away. A similar means can be done in computer attacks. By feinting one type of little attack, a larger, more deadly attack can come from another site, leaving the target trying to fight the little fire instead of preparing for the larger attack [Cohe1996].

- **Lies**: outright falsehoods (through the media, radio, etc.). Performing a man-in-the-middle attack to impersonate a loyal friend of the defender in order to talk about how to "solve" the problem can be considered a lie [Cohe1996].

- **Insight**: intellectual and psychological knowing of what the knowing what the enemy will do. By launching different types of probes and attacks, the attacker can gain insight as to the level of responses each attack gets and how they will respond to future attacks [Cohe1996].

[26] This fake artillery is known as a "Quaker gun" since Society of Friends members (Quakers) are conscientious objectors to war.

3.5.7 Parker's Taxonomies

This section will review the contributions of Parker and his contributions to the computer attack taxonomic literature. Section 3.5.7.1 tells how Parker extended the classic three security classes: confidentiality, integrity, and availability. Section 3.5.7.2 shows his attack classes derived from much actual data on computer attacks.

3.5.7.1 Extensions to the Three Basic Categories of Security

In 1991, Parker [Park1991] expanded the three basic types of categories (confidentiality, integrity, and availability) into five by adding the categories *authenticity* and *utility*. He defines authenticity as data being valid; "Authenticity of information refers to its extrinsic correct or valid representation of that which it means to represent" [Park1991]. In contrast to integrity, Parker defines integrity as, "...means that all of the information is present and accounted for (not necessarily accurate or correct)."

Utility is the, "...state of being useful or fit for some purpose, designed for use or performing a service." Parker gives an example in which he asserts that if all U.S. monetary values stored in a computer were changed to their equivalent in yen from dollars, the data would still have the other four attributes (confidentiality, integrity, availability, and authenticity). However, the information would not be authentic by his definition, that is, a "valid representation of that which it *means* to represent" (Emphasis added). It means to represent dollars, not yen. Another example he gives of a loss of utility but not availability is a computer without power. Also, in that instance, the availability of the computational power is denied, contrary to Parker's assertion. He presents a table with the five categories along with control and loss examples with regard to five levels of abstraction: information, applications, operating system, hardware, and users that is not discussed further in this dissertation.

3.5.7.2 Parker's Classes of Attack

Parker lists eight primary functional vulnerabilities in computer systems [Park1975b]:

- Poor controls over manual handling of input/output data (147 out of 375 cases during the 17 years of study). This includes weak controls in data handling tasks, audit trails,

and access restrictions.

- Weak or nonexistent physical access controls (46 out of 375). Disgruntled employees played a large factor in these cases.
- Computer and terminal operations procedures, including espionage and sale of services and data.
- Weaknesses in business ethics (deception, fraud, etc.).
- Weaknesses in the control of computer programs including oversight of programmers and programs.
- Operating system access and integrity weaknesses, usually occurring on university campuses as students probed the computer systems for weaknesses.
- Poor controls over access through impersonation due to time-sharing services such as obtaining passwords.
- Weaknesses in magnetic tape control (theft, destruction, etc.).

Parker lists 17 categories of computer abuse methods and detection in [Park1989] :

1. Eavesdropping and Spying
2. Scanning
3. Masquerading
4. Piggybacking and Tailgating
5. False Data Entry (Data Diddling)
6. Superzapping
7. Scavenging and Reuse
8. Trojan Horses
9. Computer Viruses
10. Salami Techniques
11. Trap Doors
12. Logic Bombs
13. Asynchronous Attacks
14. Data Leakage

15. Computer Program Piracy
16. Computer and Computer Components Larceny
17. Use of Computers for Criminal Enterprise

Parker and Neumann [Neum1989] developed the nine-level tree of classes (not a taxonomy per se [Park1992]) referred to in Parker's writings as the **SRI Computer Abuse Methods Model** [Park1989, Park1992]. See Section 3.1.2 for a detailed description of this model.

In a 1992 chapter on computer crime, Parker [Park1992] identifies four main categories of computer crime that were identified in the 1970s amended of the US Criminal Code:

- The introduction of fraudulent records or data into a computer system
- Unauthorized use of computer-related facilities
- The alteration or destruction of information or files
- The stealing, whether by electronic means or otherwise, of money, financial instruments, property, services, or valuable data.

Parker expands his 1989 categories and cites other computer abuse studies that show different dimensions:[27]

- **By ways in which information loss occurs**: loss of integrity, authenticity, utility, confidentiality and availability.
- **By type of loss**: physical damage and destruction from vandalism, intellectual property loss, direct financial loss and unauthorized use of services.
- **By the role played by computers**: object of attack, unique environment and forms of assets produced, instrument, and symbol.
- **By type of act relative to data, computer programs, and services**: external abuse, masquerading, preparatory abuse, bypass of intended controls, passive abuse, active abuse, and use as a tool for committing an abuse.
- **By type of crime**: fraud, theft, robbery, larceny, arson, embezzlement, extortion, conspiracy, sabotage, and espionage.

[27]Boldface in listing added by author for clarity when comparison of differing taxonomies occurs later in dissertation.

- **By modus operandi**: physical attacks, false data entry, superzapping, impersonation, wire-tapping, piggybacking, scavenging, Trojan horse attacks, trapdoor use, asynchronous attacks, salami techniques, data leakage, logic bombs, and simulation.
- **By skills required**:
  - No programming skills required
    - Physical scavenging
    - Spying
    - Masquerading
    - Entering false data
    - Theft
  - Programming skills required
    - System scavenging
    - Eavesdropping
    - Scanning
    - Piggybacking and tailgating
    - Superzapping
    - Trojan horse attacks
    - Virus attacks
    - Salami attacks
    - Using trapdoors
    - Using logic bombs
    - Asynchronous attacks
    - Leaking data
    - Pirating
    - Use in criminal enterprises

3.5.8 Straub and Widom's Motivation-Security Response Taxonomy

Straub and Widom outline a four-part taxonomy of types of attackers in [Stra1984]. While this is not a taxonomy on the attacks themselves, it lists the motivation of the attacker, the response that should be given to the attacker, and groups that would conform to the taxonomic types. The taxonomy is given in Table 3.14 on page 87.

Table 3.14: Motivation-Security Response Taxonomy [Stra1984]

Type	Motivation	Response	Groups
I	Ethical Ignorance	Deterrence by policy information	professional abusers
II	Personal Gain	Deterrence by aversive conditioning	amateur criminals white collar criminals embezzlers
III	Anti-Social Motives	Prevention by access-control	career criminals system hackers deranged individuals
IV	Corruption	Detection by surveillance	corrupt high officials corrupt experts

3.5.9 Ristenbatt's Methodology for Network Communication Vulnerability

Ristenbatt describes his methodology as expanding on the Data Link Vulnerability Analysis (DVAL) by: "1) assembling the range of network performance measures; 2) originating a network taxonomy; 3) extending the network techniques and susceptibilities; and 4) including the security of the control data (to deny spoofing)" [Rist1988]. The four components of DVAL are: susceptibilities, interceptability, accessibility, and feasibility.

A tree structure taxonomy of types of networks is given based on the following descending-level tree information: transfer strategy (single hop or multiple hop network); networking transfer control method (circuit or packet switched); link transfer structure (dedicated or shared); typical link access techniques (multiple access techniques and channel access protocols); typical topological architecture; and system examples. This network taxonomy is derived from an earlier taxonomy that is not discussed further in this dissertation.[28]

Network susceptibilities are determined from three "perspectives": topology; communication protocols; and management and control. Ristenbatt gives thoughtful questions at multiple levels (topology; physical layer; data link layer; network layer; and management and control) to consider in determining the network susceptibility to vulnerabilities. Finally, he gives checklists for interceptibility, accessibility, and feasibility issues.

[28]Anderson, G.A., Jensen E.D., "Computer Interconnection Structures: Taxonomy, Characteristics, and Examples," Computing Surveys, 7, 197–213 (December 1975).

3.6 Summary

This chapter has outlined different types of attack taxonomies in the literature that can be levied against computers. The definition of a taxonomy was reviewed, and (depending on how one counts them) sixteen computer attack taxonomies presented were presented. All of these taxonomies were combined and similarities were shown among most of them. The next chapter continues the discussion of taxonomies and discusses attack taxonomies dealing primarily with operating system flaws.

Chapter 4

Operating System Integrity Flaws

This chapter overviews those computer attack taxonomies that deal primarily with flaws of operating system integrity. Operating system flaws have been the main study of focus in two of the most seminal taxonomies in the past, the Research In Secured Operating Systems (RISOS) [Abbo1976] (Section 4.1) and the Protection Analysis (PA) [Bisb1978] (Section 4.2) reports. These reports shall be reviewed as well as others by Neumann [Neum1978] (and trapdoor attacks in [Neum1995]) (Section 4.3), McPhee [McPh1974] (Section 4.4), Landwehr [Land1994] (Section 4.5), Lindqvist [Lind1997] (Section 4.6.6), and Bishop [Bish1995] (throughout the chapter).

In addition to reviewing those operating system integrity flaw taxonomies, I extend Bishop's work [Bish1995] of the comparison of the RISOS and PA studies and show that all the taxonomies listed above and those in Chapter 3 (not just RISOS and PA) have similar characteristics and can be matched.

4.1 RISOS: Research In Secured Operating Systems

One of the first taxonomies to be developed was given in the RISOS project [Abbo1976], and it was explained in more detail by a team member in [Koni1976]. They, "...developed a methodology of investigation based on the 'test-team' concept, the use of computerized test tools, and the systematic categorizing of all known flaws into generic classes" [Koni1976]. They used the tools to help the teams find the flaws. This was a similar concept to the Flaw

Hypothesis Methodology (FHM) covered in Section 2.4.1. Their entire taxonomy consisted of a "syntax" that is given below:

> A [*Class of User*] user acquires the potential to compromise the integrity of an installation via a [*Class of Integrity Flaw*] integrity flaw which, when used, will result in unauthorized access to a [*Class of Resource*] resource, which the user exploits through the method of [*Category of Method*] to [*Category of Exploitation*].

One selects each element from the list of given elements in the taxonomy. These elements are enumerated and explained in the below sections: Class of User (Section 4.1.1), Class of Integrity Flaw (Section 4.1.2), Class of Resource (Section 4.1.3), Class of Method (Section 4.1.4), and Category of Exploitation (Section 4.1.5). An example that the authors give is the following [Abbo1976]:

> An "applications" user acquires the potential to compromise the integrity of an installation via an "operating system" integrity flaw which, when used, will result in unauthorized access to an "information" resource, which the user exploits through the method of "scavenging" to "read/ transcribe data."

In the following sections, examples are quoted at length, and all of the examples given in the final report [Abbo1976] are given due to the potential unavailability of the report.

4.1.1 Class of User

Abbott et al. [Abbo1976] and Konigsford [Koni1976] divide the user into three categories: **Applications**, **Service** and **Intruder**. Applications and Service Users had subcategories as described below.

Applications users are subdivided into *Producers* and *Consumers*, where producers are the programmers who make products and consumers are users who use the products and data the producers make. All applications users have authority to use the system. If an applications user becomes a penetrator, it is similar to Case B and C (Internal Penetrator, Misfeasance) of Anderson's Penetrator Matrix (Table 3.1) on page 49.

Service users are subdivided into *Systems* and *Administrative*; systems service users maintain the computer but also have physical access to the computer; hence they could change the tapes that the computer loads (for example, its operating system or other programs). While Administrative Service users do not have physical access to the machine, they do have a great amount of access. And although tapes are usually do not need to be loaded, this type of user is usually equivalent to 'root' account in a UNIX system or an 'administrator' account in Windows NT.

Intruders are not permitted to use the computers; hence, any amount of access to the computer is considered illegal. They have "possible" malicious intent. This corresponds to Case A (External Penetrator) of Anderson's Penetrator Matrix (Table 3.1 on page 49).

4.1.2 Class of Integrity Flaw

Integrity Flaws are divided into six categories: **Physical Protection**, **Personnel**, **Procedural**, **Hardware**, **Applications Software**, and **Operating System**. They are briefly described below, as only a paragraph or two is given in either of the reports [Abbo1976, Koni1976].

Physical Protection Integrity Flaws include, "telecommunications interception, mixed-security-level access to terminals, unauthorized access to a computer room, and exposure to natural disasters" [Abbo1976, Koni1976].

Personnel Integrity Flaws involve breakdown in secure personnel interactions. They include, "...sabotage, collusion, and user error" [Abbo1976, Koni1976].

Procedural Integrity Flaws are dependent on where the user is. This involves bypassing procedures through the use of spoofing and Trojan horses.

Hardware Integrity Flaws include problems with the hardware itself such as missing the [hard wired] terminal disconnect or introduction of instruction codes to allow a user to perform a normally restricted procedure.

Applications Software Integrity Flaws include, "inadequate user-user isolation, insufficient control over access to data, and exploitable errors in program logic" [Abbo1976, Koni1976]. It is to be noted that some of these are repeated in the operating system integrity flaws listed below.

Operating System Integrity Flaws were divided into the following seven categories: *Incomplete parameter validation*, *Inconsistent Parameter validation*, *Implicit sharing of privileged/confidential data*, *Asynchronous validation / inadequate serialization*, *Inadequate identification/authorization/authentication*, *Violable prohibition/limit*, and *Exploitable logic error*. They are covered in much greater detail below in Section 4.1.6.

4.1.3 Class of Resource

Resources are subdivided into **Information**, **Service**, and **Equipment**.

Information Resources include all system and user files.

Service Resource, "represent... the unimpaired operation of the installation.... [and] ...all the capabilities of the operating system" [Abbo1976, Koni1976].

Equipment Resource is all the pieces that insure the correct operation the computers on the premise.

4.1.4 Category of Method

Methods are listed as **Interception**, **Scavenging**, **Preemption**, and **Possession**.

Interception involves, "...the interruption of communication or connection" [Abbo1976, Koni1976].

Scavenging involves the searching of something for information from various sources. Tapes are given as an example in [Abbo1976, Koni1976]. Another example not given in the reports is the method of scavenging from a dumpster.[1]

4.1.5 Category of Exploitation

The categories of exploitation that are listed in the reports [Abbo1976, Koni1976] as self-explanatory and are listed below as well.

Denial of Possession/Use has examples or subcategories of: *Steal equipment*, *Destroy equipment*, *Degrade service*, *Interrupt service*, and *Destroy data*.

[1] This is known as dumpster diving.

Table 4.1: Research in Secured Operating Systems (RISOS) categories (R1 – R7)

R1	**INCOMPLETE PARAMETER VALIDATION**
R2	**INCONSISTENT PARAMETER VALIDATION**
R3	**IMPLICIT SHARING OF PRIVILEGED / CONFIDENTIAL DATA**
R4	**ASYNCHRONOUS-VALIDATION / INADEQUATE-SERIALIZATION**
R5	**INADEQUATE IDENTIFICATION / AUTHENTICATION / AUTHORIZATION**
R6	**VIOLABLE PROHIBITION / LIMIT**
R7	**EXPLOITABLE LOGIC ERROR**

Denial of Exclusive Possession/Use lists *Read/Transcribe data* and *Steal service* as subcategories and examples.

Modification lists *Alter data* and *Alter equipment* as its examples and subcategories.

4.1.6 RISOS Operating System Security Flaws

The RISOS team divides Operating System Integrity Flaws into seven categories which are used by Bishop [Bish1995] to compare them to the Protection Analysis [Bisb1978] Flaws (See Section 4.6.2 on page 132). I shall refer to them as R1 – R7.[2] These seven categories will be matched in later sections with the Protection Analysis (PA) categories and other studies. The seven errors are listed here in Table 4.1 and are discussed in detail in Sections 4.1.6.1 – 4.1.6.7.

4.1.6.1 R1: Incomplete Parameter Validation

Incomplete Parameter Validation is the first RISOS Operating System Integrity Error. When one process calls another routine, the parameters passed between the processes or routines must be validated for the following: *Presence or absence*, *Data types and formats*, *Number and order*, *Value ranges*, *Access rights to associated storage locations*, and *Consistency among parameters (e.g., overlapping storage locations)* [Abbo1976, Koni1976]. Konigsford succinctly summarizes the flaw pattern:

- A superior process[3] does not adequately validate parameter attributes.

[2] The prefix 'R' will stand for RISOS.

[3] "any process that possesses at least one capability to a greater degree than another (inferior) process" [Koni1976].

- A superior process does not properly reiterate parameter validation.
- A superior process validates a parameter under some but not all conditions of invocation.

Today, environment variables, which are parameters into a program, are sometimes incompletely checked. Examples of this are included in Section A.1.10 on page 243.

4.1.6.2 R2: Inconsistent Parameter Validation

Inconsistent Parameter Validation is the second RISOS Operating System Integrity Error. This is similar to the first Integrity Error, *Incomplete Parameter Validation*, but it goes beyond the first Integrity Error because R1 assumes that routines completely check the parameters coming in. Konigsford describes the flaw pattern as:

- Mutually inconsistent, but individually adequate, sets of validity criteria [Koni1976].

However, if multiple routines do not check the parameters the same way, problems can occur. An example given in [Abbo1976, Koni1976] shows problems between two different routines, one of which accepts a blank in a master-file-index entry, while the other routine does not. Categories that are given under this topic are the following [Abbo1976, Koni1976]:

System routine does not adequately validate parameter attributes. For example, "the control program verifies an initial I/O transfer. However, it does not verify that the initial I/O transfer will not cause illegal modification to subsequent I/O transfers." Another example is, "A system routine validates that the lower bound of a user's buffer lies within storage to which the user is authorized access, but neglects to verify that the upper bound also lies within authorized storage."

System routine does not properly reiterate parameter validation. An example, "In a chained list of I/O commands, only the first I/O command is verified or all but the last I/O command are verified."

System routine validates a parameter under some conditions, but not under all conditions of the invocation. An example, "A 'confused-deputy' control-program service routine adequately verifies parameters when directly invoked by a user, but not when

a user's parameters are indirectly passed to the first service routine by a second service routine." This is the pass-through problem described by McPhee [McPh1974] in Section 4.4 on page 115.

4.1.6.3 R3: Implicit Sharing of Privileged/Confidential Data

Implicit Sharing of Privileged/Confidential Data is the third RISOS OS Integrity Error. Two subcategories are given: Explicit transfer of information and Implicit transfer of information. I am not sure why the reports ([Abbo1976, Koni1976]) Explicit and Implicit would be listed as subcategories of Implicit. Konigsford [Koni1976] lists two flaw patterns of this category as:

- Privileged information located in storage accessible to an inferior process.
- Indirect transfer of privileged information to an inferior process via acknowle[d]gement or timing.

The examples in the reports [Abbo1976, Koni1976] that seem to follow the two flaw patterns given above are given below:

Explicit transfer of information. For example, "While servicing a user request, the control program uses a user-accessible buffer to scan master-file-index entries. The user asynchronously reads the buffer and obtains another user's file-index password during this activity." Another example: "The control program does not erase blocks of storage or temporary file spaces when they are reassigned to another user ('unerased blackboard')." The final given is, "A user's password is still legible through the overstrike characters on his terminal printout, or a user's password is listed on his batch output if his job command is flushed due to incorrect syntax."

Implicit transfer of information. The example given is a timing covert channel used by the user to crack a password: "A user decomposes a password piecewise by locating it on a virtual memory page boundary and noting page faults or by precisely timing variations in the execution time required by a password-checking routine." This is very similar to the *differential power analysis* attack, where the power load of a device (such as a smartcard) is

monitored for different occurrences of instructions. Because the power not only varies with the different instructions, different computations in a particular cipher have different power outputs. By monitoring those and knowing how a cipher is designed, one can partially determine how the cipher is implemented [Koch1999]. Also see an article on a *cipher instruction search attack* [Kuhn1998] and another article on programming in the presence of a hostile opponent [Ande1995].

4.1.6.4 R4: Asynchronous Validation / Inadequate Sharing

Asynchronous Validation / Inadequate Sharing is the fourth Integrity Error given in the RISOS project. It has elements of the PA project's P1 (*Consistency of Data Over Time*) and P6 (*Serialization*) [Bisb1978]. Two subcategories are given by the RISOS reports:

Asynchronous modification of user (inferior process) storage. The pattern flaw given by Konigsford [Koni1976] for this first subcategory is:

- "Asynchronous modification of storage shared by inferior and superior processes"

The switching of a piece of data after it has been validated for a process or subroutine and before the actual time of use is known as the "Time-Of-Check-to-Time-Of-Use" (TOCTTOU) error [McPh1974]. Examples of such are given in Konigsford [Koni1976] and Abbott [Abbo1976]:[4] "A user performs asynchronous I/O into his parameter list to illegally modify a previously validated system call." A second example: "A user performs I/O into a checkpoint/restart file so that his process is given additional, but unauthorized, privileges when restarted." And a third:

> Within the calling sequence to invoke control-program services, a user substitutes an indirect addressing instruction, that increments its target address after each execution, for the simple indirect instruction expected by the control program. The control program does not adequately validate the calling sequence and consequently validates one set of parameters on the first indirect access and then utilizes unvalidated parameters on its second indirect access.

[4]Only the first two examples were given in Abbott [Abbo1976].

All of these listed are examples of TOCTTOU.

Inadequate serialization / control of protected storage. The second part of the pattern flaw given by Konigsford is:

- "Inadequate serialization of storage access among (trusted) superior processes"

This lack of control over the protected storage allows a user to modify the data for the control routine (usually the return address) to either point to his own code, or modify the privilege level of the program (see Section A.1.8 on page 241 on noncaptive environments for a particular attack similar to these listed below). Examples given in Abbott and Konigsford [Abbo1976, Koni1976] are given below:

> A user issues a system call that (in part) sets values in an I/O control table and then returns control to the user. The user then issues a second, different system call that also (in part) stores values in the I/O control table — thus overlaying a portion of the previously set values in such a way as to gain unauthorized I/O privileges for the I/O performed in conjunction with the first system call.
>
> A user induces a system routine to overlay its own parameter/storage area. The user supplies an address where code is to be stored by the system routine upon return of control to the user. This user-supplied address overlays the initial word of a buffer where the system routine has stored a return-jump instruction.

These two examples have the additional similarity to the Protection Analysis P6 (*Serialization*) and P1 (*Consistency of Data Over Time*) [Bisb1978].

4.1.6.5 R5: Inadequate Identification/Authentication/Authorization

Inadequate Identification/Authentication/Authorization is the fifth of the RISOS Integrity Errors. This error overviews the task of the operating system to uniquely and correctly identify and authenticate resources. This covers the spectrum from a user logging into a computer to the correct program or library being loaded by the OS. Two subcategories are given with examples below from Konigsford and Abbott et al. [Koni1976, Abbo1976].[5]

[5]Abbott et al. [Abbo1976] contains the same examples as Konigsford [Koni1976], except for one in Konigsford that is not in Abbott et al.

Inadequate resource identification/isolation. The first example given in Abbott [Abbo1976] is, "A user program with the same name as a system program is preloaded by a user and is then accepted and used by the system." This is similar to the "Naming" error in Protection Analysis [Bisb1978] and the "Non-unique identification of system resources" also found in McPhee [McPh1974]. The second example given in [Abbo1976] is:

> A system routine assumes the validity of a system control table whenever the control table is located in system storage to which users do not have direct write access. In fact, it is possible for a user to create a counterfeit control table and have it copied into system storage by certain control program service routines, such as the storage deallocation routine.

While it is true that the system did not adequately identify (naming) and authenticate the "system control table" that it was using, the root of the error is a wrong assumption that things are correct if a user cannot write to the memory. Validation is the key problem.[6]

Bypass of controlled-access security. The second category of the this Integrity Error is any bypass of security. This is a very general category and is not necessarily mutually exclusive to the first category listed above (Inadequate resource identification/isolation). Five examples are given in [Koni1976], the first four of which are in [Abbo1976].

- A user legally bypasses the file-initialization (open) routine and its security mechanisms by utilizing a basic file-access method.
- A user obtains system privileges by taking a legal exit from abnormal job-termination (i.e., abort) processing.
- A user obtains system privileges by discovering and using a "trap door" exit to the system (i.e., unadvertised capability) meant for system-maintenance programmer use.
- An operating system that does not prevent a user from simulating its log-out and log-in functions permits an unattended (hardwired) terminal to simulate a logged-out terminal and to obtain another user's password during a simulated log-in process.

[6]Perhaps a cryptographic hash function would help verify the correct programs and libraries.

- A user is able to obtain unauthorized privileges from the system operator, because the operating system does not prevent a user from simulating control-program messages to the system operator.

The first is a bypass of domain by using something exposed (the basic file-access method) coupled with the OS's insufficient validation. Neumann and Parker [Neum1995] categorize many instances of trap doors in their computer misuse model (TD1 – TD8). Validation is the key to the remainder of the examples.

4.1.6.6 R6: Violable Prohibition/Limit

Violable Prohibition/Limit is the sixth of the RISOS Operating System errors. Two subcategories and their examples are given below [Abbo1976, Koni1976]:

Violable system limit. Simply put, this is any process that violates a limit set by the system. The most common type of attack is the buffer overflow (see Section A.1.15 on page 245). When this limit is crossed (by lack of validation), a security domain may be breached. Examples include:

- A user is supposedly constrained to operate only within an assigned partition of main storage, while, in fact, the user may access data beyond this partition.
- A user is supposedly constrained in the amount of system queue space available to his process, when, in fact, the user may create an "illogical request" consisting of an uninterruptible (sic), endless loop on a system call that eventually uses up all the control program's queue space. This causes the system to crash.

Violable system procedural prohibitions. This is slightly different from the *violable system limit* in that it does something that the documentation of the system says not to do. It would seem that RFCs would be ripe for this type of exploit because the documents contain words such as **must**, **may**, and **should**. Violations of these parts of the protocol may lead to ripe exploitation. For example, if an RFC says that one system **should** take a particular action, the other system needs to make sure that it can handle the situation when the action does not occur as it normally should.

4.1.6.7 R7: Exploitable Logic Error

Exploitable Logic Error is the final RISOS Operating System Integrity Errors. It involves any logic error that causes something in the system to be exploited or broken. There are three types, and they are listed below with examples [Abbo1976, Koni1976]:

Incorrect error-handling sequencing.

- The operating system fails to update a count of unsuccessful log-in attempts if a user presses the interrupt key (NAK)[7] on his terminal just after submitting a password guess.

This example is a sequencing error coupled with incorrect validation.

Instruction side-effects.

- The operating system uses full-word arithmetic on a half-word return address supplied by the user. If the value supplied is -1, this causes an overflow into the index field of the word and a return to the user in control state.

- An operating system uses a particular indirect-addressing instruction in user space to access some parameters. The user substitutes a similar indirect instruction that increments an index register after each execution, thus creating a flaw.

These examples are caused by incorrect validation and on the second, exposed representations.

Incorrect resource allocation/deallocation.

- The same tape or core block is assigned to two users at the same time.

This example is caused by bad authentication (validation). Overall, incorrect deallocation is a problem that needs to be fixed.

4.1.6.8 Summary

The previous section has outlined the seven types of errors given in the two reports of Abbott and Konigsford [Abbo1976, Koni1976]. Section 4.2 details the Protection Analysis (PA) report and how the ten PA error types are derived.

[7] **N**ot **A**c**K**nowledged.

4.2 PA: Protection Analysis

The Protection Analysis (PA) project at ISI [Bisb1978] wanted to automatically detect vulnerabilities in software. The RISOS project [Abbo1976] and the PA [Bisb1978] report are two of the earliest and most fundamental security taxonomies. Like the RISOS report, the goal of the PA project was to, "...further understand operating system security vulnerabilities...." In addition, the PA project sought to, "...where possible, identify automatable techniques for detecting such vulnerabilities in existing system software." A separate report containing an annotated bibliography and an index to terminology is cited as [Carl1978a].

Section 4.2.1 will cover the original goals and work of the Protection Analysis project. In a later section (Section 4.6.2 on page 132), the comparisons of RISOS and PA done by Bishop [Bish1995] and my own comparisons are shown.

4.2.1 Protection Analysis Original Work

The PA report implied that the RISOS project (Section 4.1) at the System Development Corporation (SDC) dealt with penetrating a system only by using skilled people, primarily with the "penetrate and patch" approach described in Section 2.4 on page 30. The PA report continued by contrasting the RISOS project's work with their own. The strategy of the PA initially consisted of the following steps:

- Collection of "raw" error descriptions.
- Rerepresentation of raw error descriptions in a more formalized notation (producing "raw error patterns").
- Elimination of superfluous features and abstraction of specific system elements into system-independent elements to develop generalized error patterns.
- "Feature extraction": instantiation of generalized features and searches for instances of these features in the target operating system, and the description of their relevant contexts.

- Comparison of combinations of features instances and their contexts with the features and relations expressed in the appropriate error patterns.

This "pattern-directed protection evaluation" was further expanded in a related report [Carl1975]. Thus, the fist goal was to take errors and build them into "raw error patterns" which could be made into "generalized patterns." It was then hoped to find major error types: "Subsequent instantiation of the generalized patterns by replacing the more general features with their more specific counterparts in particular classes of operating systems or particular functional areas might be expected to reveal previously undiscovered operating system errors" [Bisb1978]. Secondly, it was hoped, "...this approach might result in an empirically sound taxonomy of operating system vulnerabilities and their causes..." [Bisb1978]. This would be done by a theoretical program that would be given the developed "raw error patterns" and could detect them in any program, including an operating system.

After collecting more than 100 errors from six operating systems (TENEX, MULTICS, EXEC-8, GCOS, UNIX, and OS/360), they found it difficult to, "write down a pattern that satisfactorily captured the essence of the error" [Bisb1978]. In addition, the comparison between a generic error pattern and the code itself proved difficult. They would have to do a two step process:

1. "Normalizing" the target system by extracting the information relevant to the evaluation and representing it in the form required by a comparison procedure.

2. Executing the comparison procedure.

They continue:

> Such an ideal is clearly out of reach, however. There exists no model into which the protection-relevant features of an existing system can be mapped and in which they can be related for comparison with given patterns, general enough to apply to wide classes of errors and systems. It is even difficult to determine with precision which elements of existing systems are relevant to protection and which are not.

Table 4.2: Protection Analysis (PA) categories (P1 – P10)

P1	**CONSISTENCY OF DATA OVER TIME**: (Integrity must be maintained)
P2	**VALIDATION OF OPERANDS**: (Integrity of input data)
P3	**RESIDUALS**: (Information that is "left over")
P4	**NAMING**: (Must have resolution in objects; No ambiguity)
P5	**DOMAIN**: (Security boundaries must be maintained)
P6	**SERIALIZATION**: (Some objects are not concurrent)
P7	**INTERRUPTED ATOMIC OPERATIONS**: (Some objects are atomic)
P8	**EXPOSED REPRESENTATIONS**: (Data hiding must be maintained)
P9	**QUEUE MANAGEMENT DEPENDENCIES**: (Overflowing bounds)
P10	**CRITICAL OPERATOR SELECTION ERRORS**: (Programming errors)

They ended up using a partially automated approach. The first was to automatically search for simple instantiations of part of the pattern and then to manually compare those features to see if they interacted with each other to form the error. They met with some success [Bisb1976].

4.2.2 Protection Analysis Taxonomy Ten Error Types

Bisbey and Hollingworth [Bisb1978] were asked by their sponsors (ARPA)[8] if the "protection analysis process was bounded — i.e., whether the number of error categories was both finite...." They then sought to take the gathered errors and find error categories. Bisbey comments, "We were to subsequently work from the postulated error categories to develop automatable search strategies rather than to pursue the pattern-directed approach of gradually building up a set of empirically based categories" [Bisb1978].

The developed ten categories of errors, but realized that those errors covered different levels of abstraction. Although listed in the report as errors one through ten, I will refer to these categories as P1 – P10.[9] The ten errors are listed here in Table 4.2.

In the PA report, two errors had subtypes. The third error, Residuals (P3) had three subtypes: Access, Composition, and Data. I will label them P3A, P3B, and P3C respectively. All data is considered to be in a sequence of cells. Access residuals refer to the access capabilities (or pointer capabilities) of the cell. Pointers to other data items that are not

[8]Advanced Research Projects Agency, the name after which the ARPANET was derived.
[9]The prefix 'P' will stand for **P**rotection Analysis.

fully deleted create errors with dangling pointers. Bisbey [Bisb1978] cites access residuals as, "Incomplete revocation or deallocation of the access capabilities of the cell." Composition residuals consist of errors regarding that cell's context (i.e., the whole picture) with other cells. Bisbey gives as an error of this type as the, "Incomplete destruction of the cell's context with other cells or objects." Finally, data residual errors consist of those values within the cell itself. Bisbey reports the residual error as, "Incomplete destruction of old values within the cell."

Domain errors (P5) had two subtypes: Information associated with the wrong domain; and Incorrect enforcement at a domain crossing. I shall label those errors P5A and P5B respectively.

This section outlined the results of the Protection Analysis (PA) report. Section 4.6.3 on page 139 overviews Peter Neumann's reducing the ten categories of PA down to nine categories, while a later section (Section 4.6.2 on page 132) will compare PA with RISOS. But first, ten attacks of the PA taxonomy are given an explained by quoting the words of the original authors [Bisb1978] with my comments.

4.2.2.1 P1: Consistency of Data Over Time

Bisbey and Hollingworth [Bisb1978] explain the first error type, P1:

> Operating systems continuously make protection-related decisions based on data values contained within the system data base as well as on values which have been submitted to and validated by the system.
>
> In order for a correct protection decision to be made (in the absence of other types of protection errors), the data must be in a consistent state, and remain in a specific relationship with other data items during the interval in which the protection decision is made and the corresponding action taken.

As we shall see later, this along with type P6 (Serialization) is where the TOCTTOU[10] errors come from.

[10]Time-of-Check-to-Time-of-Use [McPh1974].

4.2.2.2 P2: Validation of Operands

The second type of error, P2 (Validation) is explained by Bisbey and Hollingworth [Bisb1978] as the following:

> Within an operating system, numerous operators are responsible for maintaining the system's data base and for changing the protection state of processes or objects known to the system. Many of these operators are critical in the sense that if invalid or unconstrained data are presented to them, a protection error results.

This is a broad category, in which error type P9 (Queue Management Dependencies) is a subtype, or an error type that can be combined with this one (P2). This error type will form the basis for the first of the four basic error types of VERDICT, Validation (Section 5.3.1 on page 158).

4.2.2.3 P3: Residuals

The third type of error, Residuals (P3) is broken into three subtypes, P3A (Access Residuals), P3B (Composition Residuals), and P3C (Data Residuals). They are explained in the original report as the following:

> A generally accepted error type is that of the "residual," i.e., information which is "left over" in an object when the object is deallocated from one process and allocated to another. Several types of residual errors exist, including the following:
>
> 1. Access residuals (P3A): Incomplete revocation or deallocation of the access capabilities to the object or cell.
> 2. Composition residuals (P3B): Incomplete destruction of the cell's context with other cells or objects.
> 3. Data residuals (P3C): Incomplete destruction of old values within the cell.

This error type forms the basis of the fourth error type in VERDICT, Deallocation (or Residuals) (Section 5.3.4 on page 159).

4.2.2.4 P4: Naming

The fourth type, Naming (P4) can occur with Access residuals (P3A), but is a broader error. Bisbey [Bisb1978] explains:

> Names are used within operating systems to distinguish objects from one another. There are many ways in which name binding errors can lead to protection errors. For example, often the naming scheme does not have enough resolution (or does not use that resolution) to distinguish properly between named objects. This results in those errors typified by a user creating an ambiguity by naming objects with the same name as a previously named (or about to be named) object with the system, as a result, referencing the wrong object.

As will be shown in later sections, this error is the result of some combination of Validation, Exposure, or Randomness errors (Sections 5.3.1, 5.3.2, and 5.3.3 on pages 158, 158, and 158 respectively).

4.2.2.5 P5: Domain

The fifth PA taxonomy category is that of Domain. This is a broad error, the result of other errors (see Section 5.1.3 for a discussion on Cause and Effect. The PA report describes domain as:

> A domain is an authority specification over an object or set of objects (usually thought of in terms of an address space). Enforcement of domains is typically limited to the resolution of the hardware protection mechanism provided by the computer. Many of the errors in operating systems are the direct result of one of two types of domain-related errors:
>
> 1. Information associated with the wrong domain.
> 2. Incorrect enforcement at domain crossing.

Information associated with the wrong domain is the result of exposure of some object that can be assessed from a different domain that what was intended, and incorrect enforcement at domain crossing is the result of bad validation.

4.2.2.6 P6: Serialization

The sixth error given is Serialization. This is very similar to the seventh category seen in the next section (Section 4.2.2.7), Interrupted Atomic Operations. In fact, Bisbey and Hollingworth say, "...it became immediately evident that the error type "Interrupted Atomic Operations" was a special manifestation of this error category and should be treated in the same context" [Bisb1978]. It is also similar to the first category (Consistency of Data Over Time) in TOCTTOU[11] operations. Bisbey gives a generalized short definition:

> Within any operating system, there are resources to which the operating system must not only control access, but also prevent concurrent use or otherwise enforce orderly use. This problem, known as "serialization" is of particular importance in multiprogramming systems where serialization errors often result in protection errors.

4.2.2.7 P7: Interrupted Atomic Operations

The seventh error, Interrupted Atomic Operations, as shown in the quotation above (Section 4.2.2.6) is a subtype of Serialization. The PA report gives a very short explanation [Bisb1978]:

> Several protection errors have appeared in which the enforcement of a protection policy was based on the assumed uninterrruptability (sic) of an operation. In each of the cases, the operation was in face interruptable (sic), resulting in a protection error.

4.2.2.8 P8: Exposed Representations

The eighth error is Exposed Representations. This error forms the basis of the second error in VERDICT, Exposure (Section 5.3.2 on page 158). Bisbey and Hollingworth explain the error as follows [Bisb1978]:

> To each user, an operating system presents an abstract machine consisting of the hardware user instruction set plus the pseudo-instructions provided through

[11] Time Of Check to Time Of Use [McPh1974].

> the supervisor call/invocation mechanism. The pseudo-instructions, in general, allow the user to manipulate abstract objects for which representations and operations are not provided in the basic hardware instruction set. Inadvertent exposure by the system of the representation of the abstract object, the primitive instructions which implement the pseudo-instructions or the data structures involved in the manipulations of the abstract object can sometimes result in protected information being made accessible to the user, thereby resulting in a protection error.

Exposure, as we will see later (Section 5.3.2) can be either a cause of another error, or an effect of another error (such as Residuals). In CVE (Common Vulnerabilities and Exposures),[12] the database created by a number of companies, differentiates between vulnerabilities, which are or cause flaws, and exposures. Exposures, like the giving of information through the *finger* service, reveals information that could be used directly or indirectly for the exploitation of a vulnerability [Zimm1991, Stev1994].[13]

4.2.2.9 P9: Queue Management Dependencies

The next to last error given, Queue Management Dependencies, is very much related to the second error, Validation. The PA report [Bisb1978] states:

> This error type broadly includes those errors characterized by improper or incomplete handling of boundary conditions in manipulating data structures such as system queues or tables. The consequence is generally a system crash or lockup resulting in gross denial of service. We distinguish this from legitimate denial of service conditions when the system is merely overloaded, but still functioning according to the scheduling algorithm design specifications.

This error is the same as incorrect validation or validation that is not even done. It is probably the most common error in security vulnerabilities today [Cowa2000].

[12] http://cve.mitre.org
[13] [Stev1994], pp.481–483.

The last part of the PA statement about "legitimate denial of service conditions" has been studied in more detail in Gligor [Glig1983]. Denial of Service (DoS) attacks are caused by errors, many of them buffer overflows, or Queue Management Dependencies.

4.2.2.10 P10: Critical Operator Selection Errors

The last error was designed to designate any other error that did not fit into the previous nine. Anything that the human did that incorrectly (human error) fit into this error. Bisbey and Hollingworth [Bisb1978] conclude their descriptions of errors:

> This error type includes those errors in which the implementer invoked the wrong function, statement, or instruction resulting in the program performing the wrong function. In a sense, this is a catch-all category, since every programming error can ultimately be so classified.

The RISOS project had as its final category a similar catch-all error, Exploitable Logic Error (R7) [Abbo1976].

4.2.3 Reformulation and Reduction of Original PA Categories

The Protection Analysis team analyzed in detail four categories: "Inconsistency of a Single Data Value Over Time";[14] "Validation of Operands"; "Residuals"; and "Serialization" [Abbo1976]. It was the intent of those reports to create tools to find errors in programs so that they could be automatically eliminated. The individual reports are listed below:

- Consistency of Data over time (P1) was developed into a separate report, *Protection Errors in Operating Systems: Single Data Value Over Time* [Bisb1975].
- Validation of Operands (P2) was developed into *Protection Errors in Operating Systems: Validation of Critical Conditions* [Carl1976].
- Residuals (P3) was developed into *Protection Errors in Operating Systems: Residuals* [Holl1976].

[14] Probably P1: Consistency of Data Over Time.

- Serialization (P6) was expanded in *Protection Errors in Operating Systems: Serialization* [Carl1978b].

While they were studying the fourth error type "serialization", "...it became immediately evident that the error type 'Interrupted Atomic Operations' was a special manifestation of this error category and should be treated in the same context" [Abbo1976]. Therefore, it appears that P6 (Serialization) and P7 (Interrupted Atomic Operations) can be grouped together. This will be commented on again in Section 5.2.3 on page 155 when I combine categories in the merged taxonomy.

4.2.4 PA: Four Global Error Categories with Subcategories

The PA project made four "global error categories" from the ten original categories:

1. **Domain Errors**
2. **Validation Errors**
3. **Naming Errors**
4. **Serialization Errors**

In the PA final report [Abbo1976], those four errors are listed with each of the ten original errors listed beneath them. The numbers of the original errors are not listed in the report's table; however, I place in parentheses the error type (P1 – P10, P3A – P3C, P5A – P5B) that I think the authors meant.

- **Domain Errors**
  - Exposed Representation Errors (P8)
  - Attribute Residual Errors (P3C)
  - Composition Residual Errors (P3B)
  - Originally Catalogued Domain Errors (P5 (including P5A, P5B))
- **Validation Errors**
  - Queue Management/Boundary Errors (P9)
  - Originally Catalogued Validation Errors (P2)
- **Naming Errors**

  - Access Residual Errors (P3A)
  - Originally Catalogued Naming Errors (P4)

- **Serialization Errors**
  - Multiple Reference Errors (P1?/P4?)
  - Interrupted Atomic Operator Errors (P7)
  - Originally Catalogued Serialization Errors (P6)

It is noted that I do not match error P10, Critical Operator Selections Errors; since it is a catch-all error, it does not have a one-to-one match and was probably not included in the listing. It is also to be noted that the error, "Consistency of Data Over Time" is not listed in their table, and "Multiple Reference Errors" listed under "Serialization Errors" does not have a one-to-one match. I see two possible solutions:

Either "Multiple Reference Errors" could be a match for "Consistency of Data Over Time" because "Consistency..." does not appear. The "multiple references" could refer to the two accesses one makes between the time of check and the time of use (TOCTTOU). As I show later, TOCTTOU falls under the "Consistency of Data Over Time" category.

Or "Multiple Reference Errors" could be a match for P4, "Naming." The PA report (quoted above) comments that "...name binding errors can lead to protection errors." It continues, "This results in those errors typified by a user creating an ambiguity by naming objects with the same name as a previously named (or about to be named) object with the system, as a result, *referencing* the wrong object." (Emphasis added.)

4.3 Neumann and Parker's Trapdoor Attacks

This section will show the eight types of trapdoor attacks (TD1 – TD8)[15] given by Neumann and Parker [Neum1995] along with examples. These attacks were given as a subtype to Neumann and Parker's Bypass attacks (NP5), specifically CM17 (Trapdoor attacks). (See Section 3.1.2 for an overview of the nine categories of computer misuse and the 26 types of computer misuse.) Neumann expanded computer misuse type 17 (CM17: trapdoor attacks) into eight subtypes, shown in Table 4.3.

[15]In Neumann's book [Neum1995], the attacks were listed as a – h. I will label the first attack, "Improper identification and authentication" as TD1, where the prefix "TD" will stand for **T**rap **D**oor attack. All the other trapdoor attacks will be labeled similarly as TD1 – TD8.

Table 4.3: Neumann and Parker's Types of Trapdoor Attacks [Neum1995]

Subtype	Mode of attack (within type 17 in Table 3.4, page 54)
TD1	Improper identification and authentication
TD2	Improper initialization and allocation
TD3	Improper finalization (termination and deallocation)
TD4	Improper authentication and validation
TD5	Naming flaws, confusions, and aliases
TD6	Improper encapsulation, such as accessible internals
TD7	Asynchronous flaws, such as atomicity anomalies
TD8	Other logic errors

4.3.1 Neumann and Parker Trapdoor Types

Explanations and examples of each trapdoor attack category from [Neum1995] include the following:

4.3.1.1 TD1: Improper Identification and Authentication

A generic error that occurs whenever little or no validation is employed, many exploits can occur because of this error. For example, in UNIX, the .rhosts is a file that lists the users who can log into the computer (without a password) if they have the same user name. Designed for easy access across a group of machines (e.g., computing lab), it has led to many more illegitimate entries. Because once a cracker has access to one machine, the intruder can get into *every* other machine that has a similar .rhosts entry. This was one of the techniques of the Morris Internet worm (Section 2.3.1.2, page 28).

In another example that Neumann gives in [Neum1995], early versions of sendmail, the Mail Transfer Agent (MTA) on most UNIX computers had a debug command. Spafford comments on the Internet worm in [Spaf1989a, Spaf1989c]:

> The worm would issue the DEBUG command to *sendmail* and then specify a set of commands instead of a user address. In normal operation, this is not allowed, but it is present in the debugging code to allow testers to verify that mail is arriving at a particular site without the need to invoke the address resolution routines. By using this option, testers can run programs to display the state of

> the mail system without sending mail or establishing a separate login connection. The debug option is often used because of the complexity of configuring sendmail for local conditions, and it is often left turned on by many vendors and site administrators.

Both of these errors falls under Neumann's [Neum1995] category of "inadequate identification, authentication, and authorization of users, tasks, and systems." They both did not authenticate before giving almost unlimited privileges to the incoming user.

4.3.1.2 TD2: Improper Initialization and Allocation

As in everyday life, an improper (and hence unknown and unstable) foundation yields to an unstable house. Neumann cites as one example, "improper domain selection" and another example as "implicit or hidden sharing of privileged data."

4.3.1.3 TD3: Improper Finalization (Termination and Deallocation)

In most systems, deletion does not remove the data but only sets a delete flag (or a free flag) indicating the area of memory (be it a file, directory structure, or a allocated piece of memory) is available for the next allocation. This can cause problems with other programs because the data is not really deleted. The object can be retrieved purposely or accidentally (see 5.3.4 on page 159).

4.3.1.4 TD4: Improper Authentication and Validation

This is so similar to the first trapdoor attack that I quote from Neumann himself:

> User authentication, system authentication, and other forms of validation are often a source of bypasses. Causes include improper argument validation, type checking, and bounds of checks; failure to prevent permissions, quotas, and other programmed limits from being exceeded; and bypasses or misplacement of control.

In the first trapdoor attack, "Inadequate identification, *authentication*, and authorization *of users*, tasks, and systems" (Emphasis added), user authentication is mentioned. Similarly,

the previous quote from the same book also mentions "user authentication." It will be shown in Section 5.3.1 on page 158that these can both be categorized as one error types. The *finger* attack in the Morris Internet Worm that Neumann mentions was caused by a bounds checking error with *gets()* [Spaf1989a, Spaf1989c].

Later in the chapter, Neumann continues by giving examples of authentication and authorization vulnerabilities in passwords:

1. Exhaustive trial and error
2. Guessing of passwords (common names, etc.)
3. Capture (sniff) unencrypted passwords
4. Derivation of passwords (dictionary attacks)
5. Universal passwords (password and encrypted password concatenated) [Youn1987]
6. Absence of passwords (e.g., .rhosts in first type of trapdoor attack)
7. Non-atomic checking
8. Editing password file
9. Trojaning the system

4.3.1.5 TD5: Naming Flaws, Confusions, and Aliases

Neumann gives three examples: aliases (either two pointers to the same item, or "multiple names with inconsistent effects that depend on the choice of name"), search path anomalies (e.g., whether the local directory is checked first or last in the search path will determine whether a program in the local directory with the same name as a system file will be run), and programs depending on directory structures (e.g., absolute vs. relative structures).

4.3.1.6 TD6: Improper Encapsulation, Such as Accessible Internals

Inside procedures and processes, information hiding must be maintained so that side effects will not be generated. This is similar to the "need-to-know" policy of classified documents where compartmentalization keeps information contained within from leaking out to those people who do not need to know.

4.3.1.7 TD7: Asynchronous Flaws, Such as Atomicity Anomalies

Neumann calls these "sequencing problems." He mentions race conditions: "...a situation in which the outcome is dependent on internal timing considerations. It is **critical** if something else depending on that outcome may be affected by its nondeterminism; it is **noncritical** otherwise." TOCTTOU is also given as an example.

4.3.1.8 TD8: Other Logic Errors

This is a catch-all and extremely encompassing category, but examples given include reading a scratch tape before using it.

4.3.2 Conclusions

Neumann [Neum1990] takes the work done by himself and Parker [Neum1989] one year earlier and shows how their nine descriptors map into the Trusted Computer Security Evaluation Criteria (TCSEC) [DoD1985] and the Information Technology Security Evaluation Criteria (ITSEC). The ITSEC was developed into what is now called the Common Criteria (CC). For more information on the CC, see [Pfle1997] and [CCEB1994]. This topic will not be investigated further in this dissertation.

This section showed the trapdoor attacks by Neumann. In a future section, I will show in Section 4.6.3 on page 139 that these trapdoor attacks match previously published taxonomies.

4.4 McPhee's Seven Classes of Integrity Flaws

McPhee's 1974 paper [McPh1974] introduces the concept of Time-Of-Check-To-Time-Of-Use (TOCTTOU) and lists seven classes of integrity flaws. Section 4.4.1 covers TOCTTOU and Section 4.4.2 details the seven classes of integrity error.

4.4.1 Time-Of-Check-To-Time-Of-Use (TOCTTOU)

The first use in the literature of the term Time-Of-Check-To-Time-Of-Use (TOCTTOU) is seen in McPhee's paper [McPh1974]. It is used a lot in his examples of his seven categories

Table 4.4: McPhee's Classes of Integrity Problems [McPh1974]

Designation	Class of Integrity Problem
M1	System data in the user area
M2	Nonunique identification of system resources
M3	System violation of storage protection
M4	User data passed as system data
M5	User-supplied address of protected control blocks
M6	Concurrent use of serial resources
M7	Uncontrolled sensitive system resources

to show how TOCTTOU can be exploited and what security measures need to be in place to counter it. Briefly, there is a finite time between the time a critical variable is checked for validity and the time that it is used. During that finite time, a perpetrator may be able to exploit the variable by modifying it so that the changed value of the variable is used without the verification of it.

4.4.2 McPhee's Seven Classes

In addition, McPhee identifies seven types of system integrity problems. He comments that new criteria, "result from the change in philosophy from 'accidental error' philosophy to the 'adversary' philosophy which says that nothing the unauthorized program can do, accidentally or deliberately, can be allowed to compromise system security controls." Computers have become much faster than those in 1974. McPhee continues, "It should be noted that less than 100 percent complete validity checks and other integrity-related 'omissions' in previous systems were not generally due to poor design or coding. In many cases they reflect valid trade-offs with respect to critical design-point / performance considerations relative to earlier releases of OS/360 systems."

McPhee lists and details seven classes of integrity flaws. I shall label them M1 – M7,[16] and they are listed in Table 4.4. They will be explained in more detail in the following sections.

[16]The prefix 'M' will stand for McPhee.

4.4.2.1 M1: System Data in the User Area

The first of McPhee's integrity flaws involves system data (that is, privileged data) in the user area (that is, accessible to the user). This is equivalent to the Protection Analysis (PA) (Section 4.2) category P8 (Exposed Representations) as well as the RISOS (Section 4.1) R3 (Implicit sharing of privileged/confidential data). Data, especially sensitive system data, should not be accessible to the user because it is stored in the user area. Users can modify the data (especially if it is an address) to gain high access on the machine.

4.4.2.2 M2: Nonunique Identification of System Resources

When a system resource is not uniquely identified, substitution for the legitimate item may be accomplished. Whether it is the name of the program or a library which the program references, it must be unique. McPhee states:

> The general solution to the problem can only be stated as the reverse of the problem; that is, the system control program must maintain and use sufficient data (protected form the user) on any sensitive control program resource, to uniquely distinguish that resource from any other control program or user resource. To be more specific than this, one must have a knowledge of the particular type of resource involved in the problem, as can be seen from the following examples:
>
> - To be uniquely identified, a program must be identified by both name and library....
> - Certain types of resources such as copies of programs can be requested and used by both the user and the control program concurrently. In this case, the control program must identify the resource as belonging to both the control program and the user to ensure that the user is not able to delete the resource while the control program is still using it....

Validity checking is the key to containing this type of error.

4.4.2.3 M3: System Violation of Storage Protection

Present day UNIX has a *setuid()* command where a user program's ID (UID) is made to be run as root (with no restrictions) in order to bypass some restrictions of the current operating system. For example, the UNIX *passwd* needs to change the */etc/passwd* file in order to change a password. Only someone with root privileges should be able to do this. The problem comes when programs exit abnormally (maliciously or benignly) or are made to execute a shell with the root privilege still intact. These "setuid" programs are considered a security hazard in today's UNIX environment, applied by programmers liberally to every program that needs any system-level access. Bishop [Bish1999] and others suggest that setuid programs are a major source of problems in the UNIX system.

In 1974, IBM's OS/VS2 Release 2 had a similar technique called the *key-switch technique* [McPh1974] that made:

> ...a system program, performing an operation in behalf of a user program, appear to be a user program for the duration of that operation. By switching from system key to user key, the system routine ensures that it will suffer the same validity-check failures as the user program would have suffered had it attempted to perform the operation itself.

These "keys" that McPhee mentions in his article seem to be a set of levels (or rings) of protection levels. McPhee continues by defining this third integrity error:

> *System violation of storage protection* is a problem where a system routine, operating in one of the privileged system keys (0–7), performs a store or fetch operation in behalf of a user routine without adequately validating that a user-specified location actually is in an area accessible to him.

This seems to be a validation error. This integrity error matches with other generic validation errors in the PA, RISOS, and other taxonomies.

4.4.2.4 M4: User Data Passed as System Data

Similar to the previous error of "System Violation of Storage Protection" (Section 4.4.2.3), "User Data Passed as System Data" is caused by incorrect validation. This error occurs when a user calls a system routine A that in turn calls system routine B. Because system routine is called by system routine A, it may not validate (or validate as much) as if it were called by the user program directly. Because of the "Incomplete Parameter Validation" (R1) or perhaps the more appropriate error description "Inconsistent Parameter Validation" (R2), user data is passed to a system routine as system data. Depending on the system routine B, it could cause many system errors.

4.4.2.5 M5: User-supplied Address of Protected Control Blocks

The fifth integrity error that McPhee lists is the "User-supplied Address of Protected Control Blocks." Under some (limited) circumstances, McPhee argues, users should be able to provide the system with the address of a protected control block. One example he gives is a, "...system control block that describes his allocation/access to a particular resource (such as a data set) to identify that resource from a group of similar resources (for example, a user may have many data sets allocated)" that the user should have access to [McPh1974].

With proper validation, he claims it would work and would be advantageous. However he cautions, "Inadequate validity checking in this situation creates an integrity exposure since the user program can provide its own (counterfeit) control block in place of the system control block and thereby cause a virtually unlimited array of integrity problems depending on exactly what sensitive data the system may be keeping in the control block involved" [McPh1974]. By his own words, a validation and exposure errors are the root cause (and effect) of this potential problem.

4.4.2.6 M6: Concurrent Use of Serial Resources

The sixth integrity error is the concurrent use of serial resources. McPhee comments that there are two causes of errors, the TOCTTOU problem and an integrity problem caused by

a user purposely causing errors with SVCs[17] as seen the:

> ...VS2 Release 2, serialization mechanisms have been introduced in certain SVCs to prevent the user from utilizing multi-tasking to pass the same resource simultaneously to two parts of the system never designed to process that resource simultaneously. In general, the reason for the original lack of of a serialization mechanism in such SVCs was the fact that only a deliberate user error would be likely to produce that situation, an event that did not have to be accounted for under the "accidental error" philosophy.

As stated in the introduction of this Section 4.4, computers must protect against deliberate attacks. One cannot assume that the user is benign. For even if the users were benign, someone could assume (falsely) the role of a legitimate user and wreak havoc. As will be seen in a later section, synchronization errors are caused by incorrect validation and incorrect exposure.

4.4.2.7 M7: Uncontrolled Sensitive System Resources

The final integrity error that McPhee lists is uncontrolled sensitive system resources. Similar to the SUID programs described in "User-supplied address of protected control blocks" (Section 4.4.2.5), some user programs need to have some system level privileges. McPhee comments on the problem:

> Because there has been no way in the past for the control program to effectively differentiate the class of programs that require such special services from the totality of user programs, these special services have generally been made available to all user programs without restriction.

They solved the integrity violations, yet still allowed some system access for special needs by the *Authorized Program Facility* (APF). APF will not be discussed further in this dissertation (see [McPh1974]). However, if this problem is not solved in computers, exposure and sharing of privileged data can be achieved. All of the McPhee integrity flaws will be discussed and compared to other taxonomies in a later section (Section 4.6).

[17] **S**uper**V**isor **C**all [McPh1974].

4.4.3 Conclusions

In the conclusion of the paper, McPhee argues that:

> There are at least two essential design concepts that must exist in order to provide system integrity:
>
> - System/user isolation
> - User/user isolation

He continues:

> Cost and risk are the key concepts. Security, or system integrity, does not have to be 100 percent foolproof. It only has to be at a level where the cost and risk involved in breaking that security exceed the benefits to be gained by doing so, or exceed the cost and risk of obtaining the same benefits in another way....
>
> Perhaps the single key factor in achieving this level of system integrity has been the "fix all exposures" approach adopted very early in the integrity effort for VS2 Release 2. This approach, in effect, says that any integrity exposure is to be fixed, no matter how unlikely it is that it could be used to violate system security.

Validation, to be discussed in Section 5.3.1 and Exposures in Section 5.3.2 on pages 158 and 158 are two of the key aspects of this paper. Together, as will be seen later, are key to solving the TOCTTOU problem.

4.5 Landwehr's Taxonomy Survey Paper

Landwehr et al. [Land1994] published a comprehensive survey of computer taxonomies in [Land1994]. In addition, they provide a taxonomy of their own. Finally, they publish and place into their taxonomy 50 actual flaws cited in the literature. Landwehr et al. describe the characteristics of a taxonomy and how it formed their thought process in creation of their taxonomy:

> A taxonomy is not simply a neutral structure for categorizing specimens. It implicitly embodies a theory of the universe from which those specimens are drawn. It defines what data are to be recorded and how like and unlike specimens are to be distinguished. In creating a taxonomy of computer program security flaws, we are in this way creating a theory of such flaws, and if we seek answers to particular questions from a collection of flaw instances, we must organize the taxonomy accordingly.

They broke down attacks into flaws by, "genesis (how), time of introduction (when), and location (where)" [Land1994]. Each instance of an attack is placed into each of the three categories (genesis, time of introduction, and location).

Landwehr et al's taxonomy is described in the sections below.

4.5.1 By Genesis

This section (the "how" of error introduction) is the most key part of the taxonomy to this dissertation. Table 4.5 shows security flaws by genesis.

The first level of distinction is whether it was introduced intentionally or inadvertently. Landwehr et al. argues that there are different strategies for countering an intentional versus an inadvertent context. For example, inadvertent errors can be countered by more resources in code checking and development. Intentional errors can be countered by more penetration testing, trustworthy programmers, and virus scanners. Their reason for dividing the genesis category into these two subcategories is to, "...collect data that will provide a basis for deciding which strategies to use in a particular context" [Land1994]. Bishop [Bish1995] counters:

> The basic problem... is that it confuses the vulnerability with the exploitation of the vulnerability. Specifically, a Trojan horse may use the inadequacy of identification when it violates a security policy, and exploits a vulnerability to give the user confidential information. The distinction between "intentional" and "inadvertent" is more the "exploitation of the vulnerability" and the "vulnerability"

Table 4.5: Landwehr et al.'s Flaws by Genesis [Land1994]

<table>
<tr><td rowspan="8">Genesis</td><td rowspan="6">Intentional</td><td rowspan="3">Malicious</td><td rowspan="2">Trojan Horse</td><td>Non-Replicating</td></tr>
<tr><td>Replicating (virus)</td></tr>
<tr><td colspan="2">Trapdoor</td></tr>
<tr><td colspan="2">Logic/Time Bomb</td></tr>
<tr><td rowspan="3">Non-Malicious</td><td rowspan="2">Covert Channel</td><td>Storage</td></tr>
<tr><td>Timing</td></tr>
<tr><td colspan="2">Other</td></tr>
<tr><td rowspan="6">Inadvertent</td><td colspan="3">Validation Error (Incomplete / Inconsistent)</td></tr>
<tr><td colspan="3">Domain Error (Including Object Re-use, Residuals, and Exposed Representation Errors)</td></tr>
<tr><td colspan="3">Serialization / aliasing (Including TOCTTOU Errors)</td></tr>
<tr><td colspan="3">Identification / Authentication Inadequate</td></tr>
<tr><td colspan="3">Boundary Condition Violation (Including Resource Exhaustion and Violable Constraint Errors)</td></tr>
<tr><td colspan="3">Other Exploitable Logic Error</td></tr>
</table>

itself. The extra layer of "intent" detracts from the identification of the specific nature of the flaw.

For classifying inadvertent flaws, they say they draw from the RISOS (Section 4.1) and the PA (Section 4.2) studies. In Section 4.6.5, I show that the RISOS taxonomy is indeed what is compared for the inadvertent categories; however, PA and RISOS do not seem to be the major draw for the intentional categories.

4.5.2 By Time of Introduction

In order to determine when in the software development cycle a flaw is introduced, Landwehr et al. produce the second phase (or axis as Bishop calls it [Bish1995]) of their taxonomy. It is shown in Table 4.6. They divide the time of introduction into three parts:

> ...*development* phase, which covers all activities up to the release of the initial operational version of the software, the *maintenance* phase, which covers activities leading to changes in the software performed under configuration control after

Table 4.6: Landwehr et al.'s Flaws by Time of Introduction [Land1994]

<table>
<tr><td rowspan="4">Time of Introduction</td><td rowspan="3">During Development</td><td>Requirement / Specification / Design</td></tr>
<tr><td>Source Code</td></tr>
<tr><td>Object Code</td></tr>
<tr><td colspan="2">During Maintenance</td></tr>
<tr><td colspan="2">During Operation</td></tr>
</table>

> the initial release, and the *operational* phase, which covers activities to patch software while it is in operation, including unauthorized modifications (e.g., by a virus).

Bishop [Bish1995] questions that definition:

> But when precisely does "in operation" mean? When software is in the beta test stage, it may be "in operation" at sites other than the developer's site, yet that is considered "development." Further, Landwehr's example of an operational flaw is the infection of a program with a virus. From our point of view, the infection is not a flaw but the exploitation of a flaw (which is that the protections are incorrectly set). So this classification needs to be made more explicit.

Landwehr et al. [Land1994] seem to answer part of Bishop's objections by continuing:

> Although the periods of operational and maintenance phases are likely to overlap, if not coincide, they reflect distinct activities, and the distinction seems to fit best in this part of the overall taxonomy.

Landwehr states that the three phases may overlap (thus making them not mutually exclusive). In addition, in his earlier quote, he states that the development phase includes, "...activities up to the release of the initial operational version of the software...." If we count the beta release as the "initial operational version," the updates at the developer's site that Bishop talks about would be in the maintenance phase, and thus they could overlap. However, even if the beta test is still considered in the development phase, Landwehr et al. does account that the two phases could overlap.

Table 4.7: Landwehr et al.'s Flaws by Location [Land1994]

<table>
<tr><td rowspan="10">Location</td><td rowspan="8">Software</td><td rowspan="6">Operating System</td><td>System Initialization</td></tr>
<tr><td>Memory Management</td></tr>
<tr><td>Process Management / Scheduling</td></tr>
<tr><td>Device Management (including I/O, networking)</td></tr>
<tr><td>File Management</td></tr>
<tr><td>Identification / Authentication</td></tr>
<tr><td>Other / Unknown</td></tr>
<tr><td rowspan="2">Support</td><td>Privileged Utilities</td></tr>
<tr><td>Unprivileged Utilities</td></tr>
<tr><td colspan="2">Application</td></tr>
<tr><td colspan="3">Hardware</td></tr>
</table>

As far as whether a virus is to be considered a flaw or an exploitation of a flaw seems to be irrelevant in talking about the time of introduction. The virus could be introduced at any of the three phases, development, maintenance, or operation. The virus is indeed not a flaw but an exploitation of a flaw, or more likely, multiple flaws.

Once a flaw is determined to be introduced during development, Landwehr argues that the flaw could be introduced during the requirement/specification/design, the source code, or the object code. Errors introduced in the object code are rare, Landwehr concedes, but he references the Thompson paper on trusting trust [Thom1984].

4.5.3 By Location

The third dimension (or axis [Bish1995]) in Landwehr's taxonomy is flaws by location. This dimension is shown in Table 4.7. After dividing the location into hardware and software, he divides the software location into operating system, support, and application.

He defines operating system functions as, "...memory and processor allocation, process management, device handling, file management, and accounting, although there is no standard definition" [Land1994]. There is much more that Landwehr gives, essentially most

everything except for those programs of support software and application software. Operating system software flaws are divided into six categories plus an *other unknown* category: system initialization, memory management, process management (including I/O and networking), file management, and identification/authentication. The reason for the division given is stated: "We have chosen the categorization above partly because it comes closer to reflection the actual layout of typical operating systems, so that it will correspond more closely to the physical structure of the code a reviewer examines."

The support software, which Landwehr concludes comprise, "...compilers, editors, debuggers, subroutine or macro libraries, database management systems, and any other programs not properly within the operating system boundary that many users share...." are divided into privileged and unprivileged utilities. Privileged utilities, like daemons, tend to have more flaws, but unprivileged even has some.

Flaws occurring in application is recognized, but not categorized. The authors recognize that this could be done and leave it to others to complete in the future.

Bishop argues for a slightly modified third dimension: "Landwehr's third axis suggest an alternate characterization of location. Where the flaw occurs is not so important as whom it affects and who can exploit it" [Bish1995]. This is similar to the cause and effect in Section 5.1.3.

4.5.4 Conclusions

Howard [Howa1997] (See Section 3.2.1 on page 59) criticizes Landwehr et al. because they use terms such as "Trojan horse, trapdoor, logic/time bomb for which there are no accepted definitions" [Howa1997]. Although Landwehr et al. give in their paper fairly standard definitions, they are a little vague. The authors quote that, "A time-bomb might be placed within either a replicating or nonreplicating Trojan horse." However, "Trojan Horse" and "Logic/Time Bomb" are on the same level. It seems to violate the *fundamentum divisionis* as spoken by Krsul [Krsu1998] (Section 2.5.2 on page 40), but the authors seem to allow the non-mutually exclusive divisions (see quote from [Land1994] below).

Howard asserts that since Landwehr's taxonomy includes categories that encompass

"other" errors (catch-all categories), it is not exhaustive; he also claims that the taxonomy is ambiguous. It is important to realize that Landwehr et al. did not attempt to create, "...simply a neutral structure for categorizing specimens" [Land1994]. They continue:

> Divisions and subdivisions are provided within the categories... Where feasible, these subdivisions define mutually exclusive and collectively exhaustive categories. Often, however, especially at the finer levels, such a partitioning is infeasible, and completeness of the set of categories cannot be assured. In general, we have tried to include categories only where they might help an analyst searching for flaws or a developer seeking to prevent them.

He continues:

> A given case may reveal several kinds of security flaws. For example, if a system programmer inserts a Trojan horse that exploits a covert channel[18] to disclose sensitive information, both the Trojan horse and the covert channel are flaws in the operational system; the former will probably have been introduced maliciously, the latter inadvertently.

The cause and effect model in Section 5.1.3 on page 152 is commented upon. The thesis of the Landwehr paper was to show how, when, and where errors occur in the development process.[19]

The authors recognized the limitations of their taxonomy. They know it is, "...an approach for *evaluating* problems in systems as they have been built." They also realize that, "the assignment of a flaw to a category may rest on relatively fine distinctions." Their 50 flaws document are just a small set of data, and statistically valid conclusions cannot be made from such a set. Although the taxonomy may not meet the stringent standards of taxonomies as set in Section 2.5.1 on page 37, it does give the system user an idea of how, when, and where errors come from. This is precisely what they intended to show.

[18]Covert channel: a communication path in a computer system not intended as such by the systems' designers.

[19]Personal conversation with Dr. Carl Landwehr, May 1999.

4.6 Summary and Comparison of Operating System Integrity Flaws

This section shows that the categories of the taxonomies outlining flaws of operating systems are similar, and most of them are shown to match. Although the various OS taxonomies are similar, the categories of the taxonomies do not always match one-to-one. An example given in a category of one taxonomy may match another example in another taxonomy, but the categories form two different aspects of the example.

4.6.1 Neumann's Nine Levels of PA

In 1978, Neumann authors "Computer System Security Evaluation" [Neum1978] in which he describes the work done at the USC Information Sciences Institute (ISI). This work he describes, the then soon to be released Protection Analysis (PA) report, is described in Section 4.2 on page 101. Neumann takes the ten categories of PA and reduces them to nine categories, under four main categories:

(A) improper protection (initialization and enforcement);
(B) improper validation;
(C) improper synchronization;
(D) improper choice of operation or operand.

Neumann continues by listing the nine categories under his four major categories:

> PROTECTION (initialization and enforcement):
>
> (1) improper choice of initial protection domain;
>
> (2) improper isolation of implementation detail;
>
> (3) improper change (e.g., a value or condition changing between its time of validation and its time of use);
>
> (4) improper naming;
>
> (5) improper (incomplete) deallocation or deletion;

VALIDATION:

(6) improper validation;

SEQUENCING:

(7) improper indivisibility;

(8) improper sequencing;

OPERATION CHOICE:

(9) improper operation or operand selection.

I shall call the nine categories N1 – N9[20] where they are elsewhere referenced in this dissertation. It is to be noted that the four major categories are not orthogonal; in fact, even in Neumann and Parker's 26 types of computer misuse, the entire list is not considered a taxonomy, but more of a list of descriptors, as more than one type of computer misuse can occur in any given attack.

A few items should be noted here. First, when Neumann combined the ten categories of PA into nine, he does not mention in the paper which of the two categories he combined. However, in a private e-mail from Dr. Neumann, he said that Validation of Operands (P2) (that is, the integrity of input data) was the same as Queue Management Dependencies (P9) (that is, overflowing bounds). Both deal with what is presently known as buffer overflows (See Section A.1.15 on page 245 for a more detailed description of this type of error). Neumann [Neum1978] combines the Validation (P2) and Queue Management Dependencies (P9) into one error, Improper Validation (N6).[21]

[20]The prefix 'N' will stand for Neumann.

[21]In an e-mail on March 27, 2000, I wrote to Dr. Neumann the following:

> Dr. Neumann:
>
> I am writing my dissertation on taxonomy of computer attacks and wondered if you could clarify a point in one of your papers.
>
> In your 1978 paper, Computer System Security Evaulation (sic), National Copmputer (sic) Conference Proceedings, pp. 1087-1095, you mention that "two of the 10 ISI categories [Protection Analysis errors] are closely related and have been lumped together here."
>
> I am not sure which two you refer to. I have reviewed all the errors and am wondering whether you lumped the ISI category, Queue Management Dependencies under your (6) improper validation or whether it fit under (9) improper operation or operand selection.

In addition, Neumann mentions that three categories have been investigated further: N3 (Improper Change), N5 (Improper (incomplete) deallocation or deletion), and N6 (Improper validation). At that time in 1978, three categories of the Protection Analysis [Bisb1978] were investigated further: P1 (Consistency of data over time) [Bisb1975], P2 (Validation of operands) [Carl1976], and P3 (Residuals) [Holl1976]. P6 (Serialization) [Carl1978b] was published in April 1978. Indeed, N3 matches P1, N5 matches P3, and N6 matches P2. For more information on those reports, see Section 4.2.

I match the nine Neumann categories (N1 – N9) with the ten categories of PA (P1 – P10). This is important because when Bishop [Bish1995] compares the RISOS and PA studies (Section 4.6.2), he uses his own numbering scheme which is based on that of Neumann [Neum1978]. Bishop's numbering scheme matches one-to-one exactly to Neumann's scheme (N1 – N9). Table 4.8 shows the result of my comparison of Neumann and PA as well as the numbering scheme of Bishop [Bish1995]. It is to be noted that Bishop used 1A, 1B, etc. where I preface the numbering system with a 'B'.[22]

> Since PA's Queue Management Dependencies relates to improper handling of boundary conditions (such as buffer overflow), I am guessing that it falls under (6), but I wanted to see if you had any comments on this.
>
> Thank you for any assistance you can give.

His response was received on the same day (27 March 2000):

> Daniel,
>
> Improper validation is a good bet, although there may also be an aspect of sequentialization (sic) problems as in nonatomic (sic) transactions. At any rate, you might think about all of the distributed denial of service attacks, and how they do or do not fit in. The problem with any taxonomy is that cases are not pure – the classes are inherently somewhat overlapping, and cases often involve multiple classes.
>
> So, don't try to make too much out of that old paper. I have not gone to too much trouble to update it – it appears more or less intact in my book – but I suspect I might do some things differently today.
>
> Let me know when you are done. I would like to see what you come up with.
>
> Best wishes, Peter

[22]The prefix 'B' will stand for Bishop.

Table 4.8: Comparison of Neumann [Neum1978], Bishop [Bish1995], and PA [Bisb1978] Categories

BISHOP [BISH1995]	**NEUMANN [NEUM1978]**	**PA CATEGORY**
	PROTECTION	
B1A	**N1**: Improper choice of initial protection domain	**P5**: Domain
B1B	**N2**: Improper isolation of implementation detail	**P8**: Exposed representations
B1C	**N3**: Improper change (e.g., TOCTTOU)	**P1**: Consistency of data over time
B1D	**N4**: Improper naming	**P4**: Naming
B1E	**N5**: Improper (incomplete) deallocation of deletion	**P3**: Residuals
	VALIDATION	
B2	**N6**: Improper validation	**P2**: Validation of operands **P9**: Queue management dependencies
	SEQUENCING	
B3A	**N7**: Improper indivisibility	**P7**: Interrupted atomic operations
B3B	**N8**: Improper sequencing	**P6**: Serialization
	OPERATION CHOICE	
B4	**N9**: Improper operation or operand selection	**P10**: Critical operator selection errors

Table 4.9: Comparison of Bishop [Bish1995], PA, and RISOS Categories

Bishop [Bish1995] / PA [Bisb1978]	Lough's PA Matchings	RISOS [Abbo1976]
B2 / P2: Validation of operands **B2 / P9**: Queue management dependencies	**P2** **P9**	**R1**: Incomplete parameter validation **R2**: Inconsistent parameter validation
B1B / P8: Exposed representations **B1C / P1**: Consistency of data over time **B3B / P6**: Serialization **B4 / P10**: Critical operator selection errors — —	**P8** — — — **P3** (Residuals) **P5** (Domain)	**R3**: Implicit sharing of privileged / confidential data
B1C / P1: Consistency of data over time **B3B / P6**: Serialization **B3A / P7**: Interrupted atomic operations	**P1** **P6** **P7**	**R4**: Asynchronous validation / inadequate serialization
B1C / P1: Consistency of data over time **B1D / P4**: Naming	**P1** **P4** **P2** (Validation of operands) **P9** (Queue management dependencies)	**R5**: Inadequate identification / authorization / authentication
B2 / P2: Validation of operands / **B2 / P9**: Queue management dependencies **B1B / P8**: Exposed representations **B4 / P10**: Critical operator selection errors **B1A / P5**: Domain (?) – (see text)	**P2** **P9** — — —	**R6**: Violable prohibition limit
B1A – B4 (Any)	**P1 – P10 (Any)**	**R7**: Exploitable logic error
Unassigned PA categories: **B1E / P3**: Residuals	—	—

4.6.2 Comparison of RISOS and PA (Bishop and Lough)

Bishop [Bish1995] was the first to show the relationships among the taxonomy categories and show that they are similar. He compares the Research In Secured Operating Systems (RISOS) [Abbo1976] (Section 4.1 on page 89) and the Protection Analysis (PA) [Bisb1978] (Section 4.2 on page 101) studies to show how one can be mapped to another.

I also compare the RISOS and PA categories; in addition, I compare my work to Bishop's and show my matchings, which are quite similar to Bishop's. Reasons for the matchings are given in the following sections. Table 4.9 shows the results of all the comparisons.

4.6.2.1 R1/R2: Incomplete/Inconsistent Parameter Validation

Both Bishop and I agree that B2/P2 (Validation of Operands) match the combined categories of R1 and R2 (Incomplete Parameter Validation, Inconsistent Parameter Validation). The categories verify and validate input parameters. When R1 and R2 are combined, it becomes a one-to-one match.

4.6.2.2 R3: Implicit Sharing of Privileged/Confidential Data

In Bishop's paper [Bish1995], R3 (Implicit Sharing of Privileged/Confidential Data) is the category in which he places covert channels. He states that, "The RISOS study focuses on the exploitation of the flaws rather than the nature of the condition which causes them." He cites two specific types of covert channels, timing and storage.

> Modulating the load average is one such channel;[23] sending information through the creation and deletion of a file is another.[24] The RISOS study lumps these very different methods into one class, whereas the Protection Analysis study separates the two. The storage channel would fall into PA category 1c (improper change),[25] since it involves monitoring changes to another process' files in a shared area; the timing channel would fall into category 1b (improper isolation of implementation detail),[26] since the timing information is a detail of implementation that can be monitored. Other methods of exploiting covert channels could fall into PA categories 3b (improper sequencing)[27] and 4 (improper choice of operator or operand)[28] as well, if the method of signalling involved flaws in those categories.

I agree with Bishop in matching R3 with P8. Exposed Representations (P8) *cause* R3 (Implicit Sharing of Privileged/Confidential Data). An example is given in the RISOS report (see Section 4.1.6.3) about a user-accessible buffer storing master-file-indices. This buffer is

[23] Author's note: timing covert channel.
[24] Author's note: storage covert channel.
[25] Author's note: Bishop category 1c is equivalent to PA category P1.
[26] Author's note: Bishop category 1b is equivalent to PA category P8.
[27] Author's note: Bishop category 3b is equivalent to PA category P6.
[28] Author's note: Bishop category 4 is equivalent to PA category P10.

exposed to any user even though it should not be; hence again P8 is the *cause* of R3. There are other examples in Section 4.1.6.3.

The broad definition of "improper change" (B1C) is derived from P1 (Inconsistency of Data Over Time) [Bisb1978]. But what does "improper" mean? If one were to send information covertly through the modulation of the system loading, I do not see how that would be improper. Yes, it is a covert channel, and yes, it is something a "normal" user would not do. (Perhaps an intrusion detection system like Denning's IDES model would notice this "abnormal" behavior and notify the system administrator [Denn1986, Denn1987]). But I see "improper" as something that should not be done under the security policy. The system's load level will indeed vary as more items are run on the system. If the viewing the system load is a problem, it should be eliminated.

The "other methods of exploiting covert channels" that Bishop mentions are unspecified. Implicit sharing of privileged or confidential data could be achieved by improper serialization (B3B/P6) or a critical operator selection error (B4/P10). It could also be achieved by numerous other errors, improper naming (P4), Validation (P2/P9), etc. Because the "other methods" are unspecified and P10 matches better with "Exploitable logic error" (R7), I do not match P10 with either R3 or "Violable prohibition limit" (R6).

I equate P3 (Residuals) with R3 because of an example in the RISOS report [Abbo1976]: "The control program does not erase blocks of storage or temporary file space when they are reassigned to another user ('unerased blackboard')." The words "unerased blackboard" are a one-to-one match with P3 (Residuals).

I also equate P5 (Domain) with R3. The reason is that when privileged or confidential data is shared (implicitly or explicitly), a domain is crossed that should not be. Covert channels, as Bishop outlines in his 1995 paper [Bish1995], cause information (confidential or privileged) to be shared across domains. Covert channels transfer information from one domain to another (if you consider each user in their own domain). The master-file-indices example given above is another example that domain is broken; the separate domain between the OS service routine and the user space was breached.

4.6.2.3 R4: Asynchronous Validation / Inadequate Serialization

This error is known as the "Time-Of-Check-To-Time-Of-Use" or TOCTTOU [Abbo1976, Koni1976]. An example "Stop-Process-Error" given in the Protection Analysis Report [Bisb1978] in a section talking about the "Inconsistency of a Single Data Value over Time" has a process modifying a value between the time it is checked and the time it is passed to a traffic controller; this too is a TOCTTOU error. Since P1 is "Consistency of data over time," P1 is the same as R4.

The categories P6 (Serialization) and P7 (Interrupted atomic operations) are considered one error ([Abbo1976], Section 4.2.3). R4 is inadequate serialization and so is P6. Therefore, P1, P6, and P7 match one-to-one to R4.

4.6.2.4 R5: Inadequate Identification/Authorization/Authentication

Consistency of data over time (P1) and Naming (P4) both match the R5 category. An example given in the RISOS report and Konigsford's report's tables says [Abbo1976, Koni1976]:

> A system routine assumes the validity of a system control table whenever the control table is located in system storage to which users do not have direct write access. In fact, it is possible for a user to create a counterfeit control table and have it copied into system storage by certain control program service routines, such as the storage deallocation routine.

P1 can be shown to match by noting that improper authentication would cause the inconsistency of data over time. In this example, the table was in protected memory; however, the user could copy a counterfeit table. There is also a naming (P4) flaw. The user could copy the counterfeit table this because of a TOCTTOU error; Abbott and Konigsford comment in the body of their texts:

> An inadequate identification/isolation flaw can be created whenever one system routine relies upon mechanisms (implemented elsewhere in the system) to ensure the isolation of system resources and, hence, the adequacy of their identification. This may be a bad policy if the mechanisms are not adequate.

> For example, a program must be identified by both program name and by the name of the library from which it was loaded. Otherwise, it is very easy for a user to preload a counterfeit program whose name is the same as some control program routine (that must be dynamically loaded when required) and to have this counterfeit routine used by the control program in place of the authentic routine.
>
> To accomplish this, the user generates an activity that will result in a control program request for this routine. The loader will see that the named (counterfeit) routine is already loaded (which is legitimate) and will set up the control program to use the counterfeit program.

We can see that the inadequate identification/authorization/authentication (R5) was caused by the inconsistency of data over time (P1) and a program name switch, i.e., naming error (P4). As we will see later in Section 5.2.3 on page 155, a synchronization error (TOCTTOU) is a result of improper validation and exposure. Bad validation in the loader allowed the user in the above example to perform a TOCTTOU resulting in a naming error to bypass the authentication (R5). In addition, a naming error (P4) matches one-to-one with the "inadequate identification" in the R5 category.

However, the main matching of this category is that validation is performed incorrectly. Therefore, "Validation of operands" (P2) and "Queue management dependencies" (P9) match this category very closely.

4.6.2.5 R6: Violable Prohibition Limit

A violable prohibition limit is simply a limit that is able to be violated (i.e., violable) even though it is forbidden (prohibited). It is also known as a buffer overflow (Section A.1.15, page 245). Bishop matches four categories to this error: P2 (Validation of operands), P9 (Queue management dependencies), P8 (Exposed representations) and P10 (Critical operator selection errors). In addition, he postulates that perhaps one could also match P5 (Domain) if you considered that the "domain" in his definition of R6, "being able to manipulate data outside one's protection domain" includes the "initial protection domain" in Neumann's [Neum1978] modified PA definition of P5: "improper choice of initial protection domain."

The matching of both Bishop's and mine of the P2 (Validation of operands) and P9 (Queue management dependencies) categories to R6 is a straightforward match. Incorrect validation can cause a process to overrun a buffer (i.e., queue) with an operand; I loosely define "operand" as any input to a procedure, function, buffer, program, etc. By overrunning a buffer, one can modify data locations beyond the buffer causing flaws.

Any prohibition on the system that can be violated could count in this R6 error. For example, one example given in Abbott and Konigsford ([Abbo1976, Koni1976]) see Section 4.2) is when a user omits notification to the OS of an I/O error-processing function. In this example, *not* doing something that should be done gives the same error as doing something that should not be done. Bishop matches P8 (Exposed representations) to R6; the example given in Abbott and Konigsford is, "a user is supposed to be constrained to operate only within an assigned partition of main storage, while in fact, the user may access data beyond this partition." The user can access (and perhaps see what is being accessed) something that should not be exposed to the user. Although P8 is about representations exposed, the fundamental meaning of R6 is that of a limit that is able to be broken. The user was not supposed to see some information that was seen. The nevertheless-seen information was what was the *cause* for the violable prohibition limit, but I do not believe that it matches one-to-one. P8 better matches R3 (Implicit sharing of privileged/confidential data).

I believe Bishop matches B4/P10 (Critical operator selection errors) with R6 because of how Neumann [Neum1978] words the error: Improper choice of operand or operation. The definition of Neumann's N9 and Bishop's rewording makes the match to R6 clearer. Neumann's examples are, "use of the wrong function, producing incorrect results; use of an unfair scheduling algorithm, producing correct results for each scheduled process, but denying service completely to certain users" [Neum1978]. Bishop's examples are, "using unfair scheduling algorithms that block certain processes or users from running; using the wrong function or wrong arguments" [Bish1995]. By saying that the "unfair scheduling algorithms that block certain processes...", he connects that thought to something (like the scheduling algorithms) that can be violated. However, as will be shown in the next section (Section 4.6.2.6), P10 can be used to match any of the RISOS errors. Any logic error is a human error.

4.6.2.6 R7: Exploitable Logic Error

Bishop and I match any PA error, especially P10 (Critical operator selection errors). Indeed, the PA report itself [Bisb1978] says that P10 is a "...catch-all category, since every programming error can ultimately be so classified." The RISOS report has an example about the OS failing to update an invalid password count if an interrupt key is pushed at the "incorrect" time during the login sequence. This could be a sequencing error (P7), or just that the OS should not allow that key to be "exposed" (P8) to the input procedure. Although this and other examples in the RISOS report (see Section 4.1 for all the example errors given) may be classified under different categories, the theme of the last error in each of the PA and RISOS reports is a operator/logic error, which match well.

4.6.2.7 Missing Categories

There are two "missing" categories in Bishop's matchings, B1A/P5 (Domain) and B1E/-P3 (Residuals). Bishop gives a possible matching of P5 (Domain) with a RISOS category (R6): "...the Protection Analysis model includes initial state (category 1a),[29] whereas all the RISOS categories speak to enforcement (although it could be argued that the 'protection domain' referred to in category 6 means that domain specified by the security policy, in which case RISOS category 6 includes initial state)" [Bish1995][30]. Even though this is not a direct quote from the RISOS report [Abbo1976], it seems to be the closest to the first example given: "A user is supposed to be constrained to operate only within an assigned partition of main storage, while in fact, the user may access data beyond this partition."[31]

[29] Bishop's category 1a is equivalent to Protection Analysis P5.

[30] Bishop defines RISOS category 6 (Violable prohibition/limit as, "...being able to manipulate data outside one's protection domain..." [Bish1995].

[31] In an e-mail on November 15, 2000, I wrote to Dr. Bishop the following:

> Matt:
>
> In your paper, A Taxonomy of UNIX System and Network Vulnerabilities, CSE-95-10, you describe RISOS category 6 (Violable prohibition/limit) as "being able to manipulate data outside one's protection domain." I am unable to find this exact quote in the RISOS report; did you get this idea from the first example in the RISOS report: "A user is supposed to be constrained to operate only within an assigned partition of main storage, while in fact, the user may access data beyond this partition?"
>
> Thanks.

His response was received on November 27, 2000:

For the second category B1E/P3 (Residuals), he does not equate it with any of the RISOS categories. Perhaps this was a mistake, because with the "exception" quoted above, he says, "...the two schemes overlap."[32]

Domain is the most unique Protection Analysis categories. This is because it always seems to be an *effect* of some other error, not a *cause*. See Section 5.1.3 for more information on cause and effect which becomes the basis for the categories in SERVR (Section 5.2.4) and VERDICT (Chapter 5) taxonomies.

4.6.3 Neumann Trapdoor Attacks and Protection Analysis (PA)

In 1978, Neumann collapsed the Protection Analysis (PA) [Bisb1978] categories from ten to nine (see Section 4.6.1). In 1995 [Neum1995], he lists eight trapdoor attacks that he derives from his and Donn Parker's 1989 summary of computer misuse technique paper [Neum1989]. I match the eight trapdoor attacks with the Protection Analysis (PA) [Bisb1978] categories and show the result in Table 4.10. His trapdoor attacks matched for the most part a one-to-one with the PA attacks by the following reasons:

TD1 — P2, P9 Improper identification and authentication (TD1) is a direct match of improper validation (P2/P9).

TD2 — P5 Improper initialization and allocation (TD2) has the closest match to improper

> Yes, from what I remember — it was a long time ago ...
>
> Matt

[32] In another e-mail on November 15, 2000, I wrote Dr. Bishop the following:

> Matt, I'm writing about your 1995 paper: A Taxonomy of UNIX System and Network Vulnerabilities (CSE-95-10) in my dissertation and wondered if you could comment on why 1e (improper deallocation or deletion) didn't have a match with a RISOS category.
>
> Hope all is going well out there.

His response was also received on November 27, 2000:

> Hi, Daniel,
>
> It's a (bad) typo. It should have been in 6, "Violable prohibition" (namely, of getting information supposedly inaccessible).
>
> Hope this helps,
>
> Matt

Table 4.10: Comparison of Neumann and Parker's Trapdoor Attacks and PA [Bisb1978]

Trapdoor Attacks [Neum1995]	**Lough's PA Matching Categories**
TD1: Improper identification and authentication	**P2**: Validation of operands **P9**: Queue management dependencies
TD2: Improper initialization and allocation	**P5**: Domain
TD3: Improper finalization (termination and deallocation)	**P3**: Residuals
TD4: Improper authentication and validation	**P2**: Validation of operands **P9**: Queue management dependencies
TD5: Naming flaws, confusions, aliases	**P4**: Naming
TD6: Improper encapsulation, such as accessible internals	**P8**: Exposed representations
TD7: Asynchronous flaws, such as atomicity anomalies	**P1**: Consistency of data over time **P6**: Serialization **P7**: Interrupted atomic operations
TD8: Other logic errors	**P10**: Critical operator selection errors

domain (P5). If some aspect of the computer system is not initialized or allocated correctly, a domain may not be set correctly. Of all the trapdoor attacks, this is the weakest match, but it is acceptable.

TD3 — P3 Improper finalization (termination and deallocation (TD3) matches with Residuals (R3) because they both deal with getting rid of data securely and how to exit properly. This is a direct match.

TD4 — P2, P9 Improper authentication and validation (TD4) again directly matches improper validation (P2/P9). This is similar to TD1, and the two trapdoor categories should probably be combined; in the summary of taxonomies, the two categories (TD1, TD4) are indeed combined.

TD5 — P4 Naming flaws, confusions, aliases (TD5) with its "naming" in the description is a one-to-one match of Naming (P4).

TD6 — P8 Improper encapsulation, such as accessible internals (TD6) is another one-to-one match, with Exposed representations (P8) the matching PA category.

TD7 — P1, P6, P7 Asynchronous flaws, such as atomicity anomalies (TD7) has a number of matchings in the Protection Analysis (PA) categories. Because it deals with race conditions, sequencing, and TOCTTOU, there are matches with the following: Consistency of data over time (P1), Serialization (P6), and Interrupted atomic operations (P7).

TD8 — P10 Other logic errors (TD8) is a catch-all error. It can refer to pure logic errors (like a missing *not* in either hardware or software), but in general it is most similar to the last category in the RISOS study (R7: Exploitable Logic Error) and the last category in the PA report: Critical operator selection errors (P10).

4.6.4 McPhee Comparison with Other Taxonomies

I compare McPhee's seven classes of integrity flaws with the seven classes of RISOS (R1 – R7; Section 4.1), the ten classes of Protection Analysis (PA) (P1 – P10; Section 4.2), and the nine classes of Neumann derived from PA (N1 – N9; Section 4.6.1). Matchings of McPhee's categories are shown in Table 4.11. Notes on the specific matchings are included below.

4.6.4.1 M1: System Data in the User Area

When matching categories with M1: System data in the user area, N1 and N2 (Improper choice of initial protection domain; Improper isolation of implementation detail) along with their one-to-one counterparts P5 and P8 (Improper Domain; Exposed Representations) are included because they deal with implementation detail that is exposed and domains that are broken. All of these errors (N1, N2, P5, and P8) are equivalent to R3: Implicit sharing of privileged/confidential data. However, the matchings are not always one-to-one.

P3 (Residuals) by itself matches with R3 (implicit sharing of privileged/confidential data) because of data that is improperly deallocated causes privileged/confidential data to be shared. The reverse is not necessarily the case. Just because some data is shared (such as M1: System data in the user area) does not mean that it was caused by residuals (P3). However, M1 is included in McPhee's comparison because of an example given in Section 4.1.6.3 which a control program does not erase deleted blocks.

Table 4.11: Comparison of McPhee, RISOS, PA, and Neumann

RISOS	PA	NEUMANN	MCPHEE [MCPH1974]
R3	**P3, P5, P8**	**N1, N2**	**M1**: System data in the user area
R5	**P4**	**N4**	**M2**: Non-unique identification of system resources
R1, R2, R5, R6	**P2, P9**	**N6**	**M3**: System violation of storage protection
R1, R2, R5, R6	**P2, P9**	**N6**	**M4**: User data passed as system data
R1, R2, R3, R5, R6	**P2, P8, P9**	**N2, N6**	**M5**: User-supplied address of protected control blocks
R4	**P1, P6, P7**	**N3, N7, N8**	**M6**: Concurrent use of serial resources
R3	**P8**	**N1, N2**	**M7**: Uncontrolled sensitive system resources
R7	**P10**	**N9**	**All**
—	**P3**	**N5**	**Not used**

R1: Incomplete parameter validation
R2: Inconsistent parameter validation
R3: Implicit sharing of privileged / confidential data
R4: Asynchronous-validation / inadequate-serialization
R5: Inadequate identification / authentication / authorization
R6: Violable prohibition / limit
R7: Exploitable logic error

P1: Consistency of data over time
P2: Validation of operands
P3: Residuals
P4: Naming
P5: Domain
P6: Serialization
P7: Interrupted atomic operations
P8: Exposed representations
P9: Queue management dependencies
P10: Critical operator selection errors

N1: Improper choice of initial protection domain
N2: Improper isolation of implementation detail
N3: Improper change
N4: Improper naming
N5: Improper (incomplete) deallocation of deletion validation
N6: Improper validation
N7: Improper indivisibility
N8: Improper sequencing
N9: Improper operation or operand selection

4.6.4.2 M2: Non-unique Identification of System Resources

With one-to-one matchings, M2 matches with "Inadequate identification of R5 (Inadequate identification/authentication/authorization), P4 (Naming), and N4 (Improper naming).

4.6.4.3 M3: System Violation of Storage Protection

As stated in Section 4.4.2.3, this is a problem that allows the system routine acts on behalf of a user routine without adequate validation. Because of this, M3 matches with the validation categories of P2 (Validation of operands), N6 (Validation of operands), and R5 (Inadequate identification/authentication/authorization). Even though it is a system violation, I do not match it to any buffer overflow categories such as P9 (Queue management dependencies) or R6 (Violable prohibition/limit).

4.6.4.4 M4: User Data Passed as System Data

However, when user data is passed as system data (and not the system passing user data as in M3), buffer overflows are included with this validation error. These include the following: P2 (Validation of operands), P9 (Queue management dependencies), R1 (Incomplete parameter validation), R2 (Inconsistent parameter validation), R6, and N6.

4.6.4.5 M5: User-supplied Address of Protected Control Blocks

This error occurs because a protected control block's address is supplied by the user. Something that should not be exposed (P8: Exposed Representations) was, and it was not validated (P2/P9: Validation of operands / Queue management dependencies). These three PA categories match one-to-one with Neumann's N2 (Improper isolation of implementation detail) and N6 (Improper validation). Privileged or confidential data is shared (R3) because of inadequate authentication (R5).

4.6.4.6 M6: Concurrent Use of Serial Resources

This is the serialization error of other taxonomies. Inadequate serialization / Interrupted atomic operations (P6/P7) and data not consistent over time (P1) along with the RISOS

error of similar name (R4) match closely. It is to be noted that in the summary of taxonomy categories (Table 4.14), M6 is in the same grouping as R4 and R5; however, R5 is not equivalent to M6 in the McPhee comparison (Table 4.4). Consistency of Data over Time in Table 4.14 matches closely with R4 and R5, but the two categories do not overlap like P2 and P9 (Validation of operands; Queue management dependencies).

4.6.4.7 M7: Uncontrolled Sensitive System Resources

Some sensitive system resources may be exposed to those processes that should not have access to them. Because of this, I match M7 with those errors that show exposure (P8: Exposed representations) and the sharing of data that should not be shared (R3: Implicit sharing of privileged/confidential data).

4.6.5 Landwehr's Comparison to Other Taxonomies

I match Landwehr's categories with those in RISOS, and some in PA. I also use some of Neumann and Parker's Computer Misuse (CM) types, since they were more equivalent than PA categories. The results are presented in Table 4.12.

Bishop does not like the "Genesis" taxonomy of Landwehr's scheme [Land1994] because of ambiguities in the time of introduction category. He takes Landwehr's categories and extends them to six dimensions: nature of the flaw (PA categories), time of introduction (modified Landwehr categories), exploitation domain, effect domain, minimum number of components needed to exploit the vulnerability, and the source of the identification of the vulnerability.

4.6.6 Lindqvist and Jonsson's Operating System Flaws

This section will show that the work done by Lindqvist and Jonsson [Lind1997] as shown in Section 3.1.3 matches similar categories of operating flaws in other studies such as RISOS [Abbo1976], PA [Bisb1978], and Neumann [Neum1978]. Lindqvist and Jonsson documented a classroom experiment to see what types of attacks groups of students could levy against a test server. Since all users were authorized on the file server, all attacks fell within three

Table 4.12: Comparison of Landwehr's Flaw by Genesis to RISOS; PA; and Neumann and Parker [Land1994]

<table>
<tr><td rowspan="11">Genesis</td><td rowspan="7">Intentional</td><td rowspan="4">Malicious</td><td rowspan="2">Trojan Horse</td><td>Non-Replicating</td><td colspan="2">CM13</td></tr>
<tr><td>Replicating (virus)</td><td colspan="2">CM16</td></tr>
<tr><td colspan="2">Trapdoor</td><td colspan="2">CM17</td></tr>
<tr><td colspan="2">Logic/Time Bomb</td><td colspan="2">CM14</td></tr>
<tr><td rowspan="3">Non-Malicious</td><td rowspan="2">Covert Channel</td><td>Storage</td><td>P1</td><td rowspan="2">CM24</td></tr>
<tr><td>Timing</td><td>P8</td></tr>
<tr><td colspan="2">Other</td><td colspan="2">N/A</td></tr>
<tr><td rowspan="5">Inadvertent</td><td colspan="3">Validation Error (Incomplete / Inconsistent)</td><td colspan="2">R1, R2</td></tr>
<tr><td colspan="3">Domain Error (Including Object Re-use, Residuals, and Exposed Representation Errors)</td><td colspan="2">R3</td></tr>
<tr><td colspan="3">Serialization / aliasing (Including TOCTTOU Errors)</td><td colspan="2">R4</td></tr>
<tr><td colspan="3">Identification / Authentication Inadequate</td><td colspan="2">R5</td></tr>
<tr><td colspan="3">Boundary Condition Violation (Including Resource Exhaustion and Violable Constraint Errors)</td><td colspan="2">R6</td></tr>
<tr><td colspan="3">Other Exploitable Logic Error</td><td colspan="2">R7</td></tr>
</table>

Computer Misuse [Neum1995]

CM13: Trojan horse attacks *(Implanting malicious code, sending letter bombs)*
CM14: Logic bombs *(Setting time or event bombs (a form of Trojan horse))*
CM16: Virus attacks *(Attaching to programs and replicating)*
CM17: Trapdoor attacks *(Utilizing existing flaws)*
CM24: Covert channels *(Exploiting covert channels or other data leakage)*

Protection Analysis (PA) [Bisb1978]

P1: Consistency of data over time *(Timing covert channel [Bish1995])*
P8: Exposed representations *(Storage covert channel [Bish1995])*

Research In Secure Operating Systems (RISOS) [Abbo1976]

R1: Incomplete parameter validation
R2: Inconsistent parameter validation
R3: Implicit sharing of privileged / confidential data
R4: Asynchronous-validation /inadequate-serialization (Including TOCTTOU)
R5: Inadequate identification / authentication / authorization
R6: Violable prohibition / limit
R7: Exploitable logic error

categories of the SRI Computer Abuse Methods Model (Section 3.1.2), NP5 – NP7.

I have matched all categories with categories from Neumann and Parker's computer misuse [Neum1995], RISOS [Abbo1976], PA [Bisb1978], and Neumann's Computer System Security Evaluation [Neum1978]. My results are presented in Table 4.13.

It is to be noted that in their expansion of NP6 (Active Misuse), they lump together CM21 (Denials of Service) with CM19 (Basic Active Misuse). In fact, CM21 is a subset of CM19 because Neumann [Neum1995] defines CM19 as "creating, modifying, using, *denying service*,...." (Emphasis added).

4.6.7 Lough's Comparison of OS Flaws

It is not trivial to match the categories of the past taxonomies. Few categories match one-to-one. I matched some categories because of the nature of the examples given in the original documents. Examples were used to match the categories because they seemed to imply what the thoughts of the documents' authors were. However, I thought some of the examples fit better in other categories of the same taxonomy, or that categories should be combined in the taxonomy itself (e.g., R3 and R5).

The various categories of the past taxonomies are matched with the eight general flaw categories. However, the categories themselves in the table may not match each other one-to-one; they may be a one-to-many matching. This is why, for example, the following occurs: P4 (Naming) *partially* matches R5 (Inadequate identification/authentication/authorization). R5 matches with P2 and P9 (Validation of operands and Queue management dependencies). One cannot apply the transitive property of mathematics and say that (because of the tables of matchings) if A = B and B = C, A = C. P4 is *not* the same as P9 (Queue management dependencies). But both P4 and P9 match R5 (Inadequate identification/authentication/authorization).

A few research questions remain about Table 4.14. First, should category 6 (Serialization / Atomic Operations) be split into two categories? The Protection Analysis (PA) [Bisb1978] splits the two, but admits that P7 (Interrupted atomic operations) was a subcategory of P6 (Serialization).

On a similar vein, what category does the Time-Of-Check-to-Time-Of-Use (TOCTTOU)

Table 4.13: Comparison of Lindqvist and Jonsson [Lind1997] with Other Taxonomies

NP5: Bypassing Intended Controls	Password attacks	Capture[1]	CM18, R5, R7
		Guessing	CM18, R5, R6
	Spoofing privileged programs		CM17, CM18, P5, R5, N1
	Utilizing weak authentication		CM18, P2, P9, R5, N6

NP6: Active Misuse of Resources	Exploiting inadvertent write permission	P5, R3
	Resource exhaustion[2]	P8, P10, R6

NP7: Passive Misuse of Resources	Manual browsing		CM22
	Automated searching	Using a personal tool	
		Using a publicly available tool	

1: "Interception" [Abbo1976]
2: "Degrade service" [Abbo1976]

Computer Misuse [Neum1995]
CM17: Trapdoor attacks *(Utilizing existing flaws)*
CM18: Authorization attacks *(Password cracking, hacking tokens)*
CM22: Browsing *(Making random or selective searches)*

Protection Analysis (PA) [Bisb1978]
P2: Validation of operands *(Validation of operands)*
P5: Domain *(Security boundaries must be maintained)*
P8: Exposed representations *(Data hiding must be maintained)*
P9: Queue management dependencies *(Overflowing bounds)*
P10: Critical operator selection errors *(Programming errors)*

Research In Secure Operating Systems (RISOS) [Abbo1976]
R3: Implicit Sharing of Privileged / Confidential Data
R5: Inadequate identification / authentication / authorization
R6: Violable prohibition / limit
R7: Exploitable logic error

Neumann's Computer System Security Evaluation [Neum1978]
N1: Improper choice of initial protection domain
N6: Improper validation

[McPh1974] fit into? On one hand, the data that is being checked and used should be consistent over time; that is, it should fall into category 1. However, the atomic operation of the check and use should not be broken, so it would seem to fit into category 6. Perhaps characteristics could be combined to create such a security flaw. That is, the combination of consistency and serialization could combine to make the TOCTTOU flaw. Krsul and Bishop [Krsu1998, Bish1999] advocate the use of security characteristics in creating taxonomic categories. My listing of OS flaws and their counterparts in major taxonomic studies is the start to creating this list and taxonomy of security characteristics. I show how this table is used to develop The SERVR Taxonomy and then later the refined VERDICT in Chapter 5.

4.7 Summary

This chapter has outlined different types of operating system integrity flaws, which have been presented in the literature. This dissertation has taken the work of Bishop [Bish1995] of combining the PA and RISOS taxonomies and has expanded it to include McPhee [McPh1974], Neumann [Neum1978], Neumann and Parker's Trapdoor Attacks [Neum1995], and Landwehr et al.'s taxonomy [Land1994]. Taxonomy categories were combined, showing that they cover the same types of attacks.

These matchings can be combined with the taxonomies of Chapter 3 to show that although some taxonomies are at differing levels, *all* published taxonomies relate to each other in some way. This summary of operating system integrity flaws forms the basis for the new taxonomy VERDICT presented in the next chapter, Chapter 5. Chapter 6 shows that VERDICT covers all categories of selected past taxonomies. Because all taxonomies have been shown to relate to each other, VERDICT will be shown superior to other computer attack taxonomies, including those with operating system integrity flaws.

Table 4.14: Summary of Operating System Integrity Flaws

SUMMARY OF TAXONOMY CATEGORIES	[BISB 1978]	[NEUM 1978]	[BISH 1995]	[ABBO 1976]	[NEUM 1995]	[MCPH 1974]
1. Consistency of Data Over Time (Integrity must be maintained); Improper change; TOCTTOU	**P1**	**N3**	**B1C**	**R4** **R5**	**TD7**	**M6**
2. Validation of operands; Improper validation; Boundary condition, overflowing bounds; Inadequate authentication / identification / authority (weak passwords)	**P2** **P9**	**N6**	**B2**	**R1** **R2** **R5** **R6**	**TD1,** **TD4**	**M3** **M4** **M5**
3. Residuals; improper deallocation	**P3**	**N5**	**B1E**	**R3**	**TD3**	**M1**
4. Naming (must have resolution in objects – no ambiguity)	**P4**	**N4**	**B1D**	**R5**	**TD5**	**M2**
5. Domain – covert channels (security boundaries must be maintained)	**P5**	**N1**	**B1A**	**R3**	**TD2**	**M1** **M7**
6. Serialization; improper sequencing / Atomic operations; improper indivisibility, race conditions	**P6 /** **P7**	**N8 /** **N7**	**B3B /** **B3A**	**R4**	**TD7**	**M6**
7. Exposed representations (data hiding must be maintained); improper isolation of implementation detail	**P8**	**N2**	**B1B**	**R3**	**TD6**	**M1** **M7**
8. Improper operation / operand selection; logic error	**P10**	**N9**	**B4**	**R7**	**TD8**	**—**

R1: Incomplete parameter validation
R2: Inconsistent parameter validation
R3: Implicit sharing of privileged / confidential data
R4: Asynchronous-validation / inadequate-serialization
R5: Inadequate identification / authentication / authorization
R6: Violable prohibition / limit
R7: Exploitable logic error

P1: Consistency of data over time
P2: Validation of operands
P3: Residuals
P4: Naming
P5: Domain
P6: Serialization
P7: Interrupted atomic operations
P8: Exposed representations
P9: Queue management dependencies
P10: Critical operator selection errors

N1/B1A: Improper choice of initial protection domain
N2/B1B: Improper isolation of implementation detail
N3/B1C: Improper change
N4/B1D: Improper naming
N5/B1E: Improper (incomplete) deallocation of deletion validation
N6/B2: Improper validation
N7/B3A: Improper indivisibility
N8/B3B: Improper sequencing
N9/B4: Improper operation or operand selection

TD1: Improper identification and authentication
TD2: Improper initialization and allocation
TD3: Improper finalization (termination and deallocation)
TD4: Improper authentication and validation
TD5: Naming flaws, confusions, and aliases
TD6: Improper encapsulation, such as accessible internals
TD7: Asynchronous flaws, such as atomicity anomalies
TD8: Other logic errors

M1: System data in the user area
M2: Nonunique identification of system resources
M3: System violation of storage protection
M4: User data passed as system data
M5: User-supplied address of protected control block
M6: Concurrent use of serial resources
M7: Uncontrolled sensitive system resources

Chapter 5

VERDICT

This chapter describes the new taxonomy called VERDICT, including its categories and characteristics. In brief, VERDICT is an acronym of the four causes of computer security errors: **V**alidation, **E**xposure, **R**andomness, and **D**eallocation. The final three letters of the acronym spell out **I**mproper **C**onditions **T**axonomy. Thus, the entire acronym is the following:

Validation
Exposure
Randomness
Deallocation
Improper
Conditions
Taxonomy

This chapter overviews the characteristics of VERDICT in Section 5.1, the evolution of the taxonomy in Section 5.2, and the categories in detail in Section 5.3.

5.1 VERDICT Characteristics

As specified in Section 2.5 on page 37, a taxonomy needs to be composed of mutually exclusive categories that are exhaustive and unambiguous. Any attempt to classify an item into a

taxonomy needs to yield repeatable equivalent results. Two things need to be considered in the development of a taxonomy. What the *fundamentum divisionis* is and secondly, what the abstraction level is. These are discussed in Sections 5.1.1 and 5.1.2 respectively. Finally, cause and effect are described in Section 5.1.3.

5.1.1 Fundamentum Divisionis

The *fundamentum divisionis*, "grounds for a distinction," or the fundamental divide is one of the most important aspects of a taxonomy.[1] The structure of a taxonomy can be defined as either a tree structure, where a decision is made about the item to be classified at each level; or as a flat structure, where one or more characteristics combine to form the classification. These two types of taxonomies are described below in Sections 5.1.1.1 and 5.1.1.2.

5.1.1.1 Tree Structure

When a taxonomy is represented as a tree structure, it is shown as an n-ary trees.[2] Most classical taxonomies are represented this way, such as the biological taxonomy of living entities.[3]

Past security taxonomies such as those discussed in Chapter 3 are usually in a tree structure such as the SRI Computer Abuse Methods Model; or they are in a list of flaws that are not necessarily mutually exclusive, such as the Protection Analysis (PA) or the Research in Secured Operating Systems (RISOS). These lists of flaws can be considered a one-level n-ary tree with n being the number of categories in the taxonomy.

The problem with a security taxonomy in a tree structure is that a security vulnerability is often composed of multiple flaws. There is no feature or question that can decide which category the security vulnerability falls. For example, as stated in Section 2.5.2 on page 40, a vulnerability could be either a race condition, a configuration problem, or both. To

[1] "A *fundamentum divisionis* is a term from scholastic Logic and Ontology that means "grounds for a distinction" [Audi, R., Ed. 1995. *The Cambridge Dictionary of Philosophy.* Cambridge University Press.] [Krsu1998], pp. 23, 126.

[2] For n = 2 for all decision levels, the tree is called a binary tree.

[3] The taxonomy is seven levels consisting of the description of the following: Kingdom, Phylum, Class, Order, Family, Genus, and Species.

overcome this deficiency, a taxonomy with a characteristics structure, as shown in Section 5.1.1.2 is needed.

5.1.1.2 Characteristics Structure

A taxonomy with a characteristics structure is defined as a taxonomy with a set of categories consisting of different types of characteristics of that which is being defined. Like the nucleotides of DNA,[4] one or more of the characteristics of the taxonomy can be linked together to describe the item that is being placed in a taxonomy. As stated in Section 2.5.2, Bishop [Bish1999] and Krsul [Krsu1998] assert that taxonomies should classify properties of vulnerabilities and not by the vulnerability itself. These **characteristics**, also called **features** or **attributes** are the building blocks that form the description of the vulnerability.

VERDICT consists of four characteristics: **V**alidation, **E**xposure, **R**andomness, and **D**eallocation. These **I**mproper **C**onditions form the **T**axonomy. A vulnerability can be the result of one or more of the four characteristics. (For more information about Cause and Effect, see Section 5.1.3.)

5.1.2 Abstraction Levels

In addition to determining what the *fundamentum divisionis* is to be, it is also necessary to determine at what level VERDICT is to be placed. Past work in taxonomies such as the PA and RISOS studies concentrated mainly on operating systems. VERDICT takes a broader approach, being able to apply the taxonomy to all aspects of computer security, beyond only the operating system.

5.1.3 Cause and Effect

In order to determine the root problems in computers, one needs to know the cause of the error. In some previous taxonomies, categories can be described as either a cause of a vulnerability or the result (i.e., effect) of a vulnerability. In some cases, a category could be

[4] A,C,T, and G stand for Adenine, Cytosine, Thymine, and Guanine.

considered *both* a cause and an effect, depending on what conditions are applied and how one views the error.

For example, exposed representations (P8), is an effect of some other type of error. The fact that some aspect of the system is visible when it should not be is an error. However, the result that the aspect is visible may be the cause of another error, such as allowing an improper sequencing, causing an improper domain access. Causes and effects can be chained together, as shown in a later section.

5.2 Evolution of VERDICT

This section will show the evolution of VERDICT to its present state. Section 5.2.1 reviews the summary of operating system attacks derived in Chapter 3. The randomness category was added to the original nine categories (Section 5.2.2). Because of cause and effect shown in 5.1.3, a combination and reduction of categories began that is outlined in Section 5.2.3. Section 5.2.4 describes how five final categories were developed into the SERVR taxonomy and why one final category was cut to develop VERDICT. VERDICT's categories are more fully described in Section 5.3.

5.2.1 Summary of Operating System Attacks

In Section 4.6.7, a table of operating system (OS) flaws is shown that is a combination of those flaws from the taxonomies listed by Bisbey et al.'s Protection Analysis (PA) [Bisb1978], Neumann [Neum1978], Bishop [Bish1995], Abbott et al.'s Research in Secured Operating Systems [Abbo1976], Neumann [Neum1995], and McPhee [McPh1974]. For convenience, the table is reprinted in this chapter in Table 5.1.

The last error, "Improper operation/operand selection; logic error" is a catch-all error. Also known as an all-inclusive "human error," it is not at the same level as errors caused by computers. Since computers are made by humans, all errors can eventually be traced back to a "human error." Because of this, it needs to be dropped from the taxonomy.

Table 5.1: Summary of Operating System Integrity Flaws

SUMMARY OF TAXONOMY CATEGORIES	[BISB 1978]	[NEUM 1978]	[BISH 1995]	[ABBO 1976]	[NEUM 1995]	[MCPH 1974]
1. Consistency of Data Over Time (Integrity must be maintained); Improper change; TOCTTOU	**P1**	**N3**	**B1C**	**R4** **R5**	**TD7**	**M6**
2. Validation of operands; Improper validation; Boundary condition, overflowing bounds; Inadequate authentication / identification / authority (weak passwords)	**P2** **P9**	**N6**	**B2**	**R1** **R2** **R5** **R6**	**TD1,** **TD4**	**M3** **M4** **M5**
3. Residuals; improper deallocation	**P3**	**N5**	**B1E**	**R3**	**TD3**	**M1**
4. Naming (must have resolution in objects – no ambiguity)	**P4**	**N4**	**B1D**	**R5**	**TD5**	**M2**
5. Domain – covert channels (security boundaries must be maintained)	**P5**	**N1**	**B1A**	**R3**	**TD2**	**M1** **M7**
6. Serialization; improper sequencing / Atomic operations; improper indivisibility, race conditions	**P6 /** **P7**	**N8 /** **N7**	**B3B /** **B3A**	**R4**	**TD7**	**M6**
7. Exposed representations (data hiding must be maintained); improper isolation of implementation detail	**P8**	**N2**	**B1B**	**R3**	**TD6**	**M1** **M7**
8. Improper operation / operand selection; logic error	**P10**	**N9**	**B4**	**R7**	**TD8**	**—**

R1: Incomplete parameter validation
R2: Inconsistent parameter validation
R3: Implicit sharing of privileged / confidential data
R4: Asynchronous-validation / inadequate-serialization
R5: Inadequate identification / authentication / authorization
R6: Violable prohibition / limit
R7: Exploitable logic error

P1: Consistency of data over time
P2: Validation of operands
P3: Residuals
P4: Naming
P5: Domain
P6: Serialization
P7: Interrupted atomic operations
P8: Exposed representations
P9: Queue management dependencies
P10: Critical operator selection errors

N1/B1A: Improper choice of initial protection domain
N2/B1B: Improper isolation of implementation detail
N3/B1C: Improper change
N4/B1D: Improper naming
N5/B1E: Improper (incomplete) deallocation of deletion validation
N6/B2: Improper validation
N7/B3A: Improper indivisibility
N8/B3B: Improper sequencing
N9/B4: Improper operation or operand selection

TD1: Improper identification and authentication
TD2: Improper initialization and allocation
TD3: Improper finalization (termination and deallocation)
TD4: Improper authentication and validation
TD5: Naming flaws, confusions, and aliases
TD6: Improper encapsulation, such as accessible internals
TD7: Asynchronous flaws, such as atomicity anomalies
TD8: Other logic errors

M1: System data in the user area
M2: Nonunique identification of system resources
M3: System violation of storage protection
M4: User data passed as system data
M5: User-supplied address of protected control block
M6: Concurrent use of serial resources
M7: Uncontrolled sensitive system resources

5.2.2 Addition of Randomness Category

In papers such as Venema [Vene1996], the lack of randomness is shown to be a cause of security errors. Appendices A.1.13 and A.1.14 on pages 244 and 244 show and explain that some results occur without adequate randomness. Because of these observations, I added to the summary of operating system attacks shown in Section 5.2.1 a category, "Improper Randomness".

5.2.3 Combination of Categories

Having removed the last error of "Improper operation/operand selection; logic error" in Table 4.14 and replaced it with "Improper Randomness," the total number of operating system integrity flaws is still eight. Recall that Bisbey [Bisb1978]; Neumann [Neum1978]; and Sections 4.2.2.2 and 4.2.2.9 state that P2 (*Validation of Operands*) and P9 (*Queue Management Dependencies*) can be combined into one category. In addition, as Section 4.2.3 states, the categories similar to Protection Analysis' P6 (*Serialization*) and P7 (*Interrupted Atomic Operations*) can be combined into one category, since P7 is a "special manifestation" of P6 [Bisb1978]. Finally, all the errors can be thought of as an improper condition and hence labeled with a prefix "Improper...." The result is shown below:

- P1: Improper change
- P2/P9: Improper validation
- P3: Improper deallocation/residuals
- P4: Improper naming
- P5: Improper domain
- P6/P7: Improper sequencing
- P8: Improper exposure
- —: Improper randomness

P1, *Improper change*, is the condition when some value or object is changed when it should not be. One needs to be able to "see" or access the object in order to change it. Once the object is able to be accessed, only a lack of proper validation will cause it to be changed. Therefore, P1, *Improper change*, is caused by improper validation and improper exposure, P2/P9 and P8. Because the taxonomy needs to show the causes of the error, P1 *Improper change* can be deleted from the taxonomy.

P4, *Improper naming*, is the result of improper validation, exposure, and randomness. For example, the improper naming of NFS file handles as described by Venema [Vene1996] is the result of improper randomness (failing to initialize the time of day variable in the file handle number computation), improper exposure (any NFS client who could guess the file handle could have access to the NFS system), and improper validation (there were no checks to see if it was a valid file handle). Because improper naming is the result of improper validation, exposure, and randomness, it can be dropped from the final taxonomy.

P5, *Improper domain*, is an effect, not a cause; it is an effect of other improper operations. When a vulnerability causes a domain to be accessed that should not be, it is the result (or effect) of that vulnerability. For example, because of improper validation on an input queue, one may be able to overflow a buffer and have the CPU execute instructions that are not in the user's authorized domain.

5.2.4 The SERVR Taxonomy

After the elimination of P1 (*Improper Change*), P4 (*Improper Naming*), and P5 (*Improper Domain*) from the eight categories shown in Table 4.14, only five remain. They form what is called the SERVR taxonomy. SERVR, pronounced "server," is the following:

Improper...

Sequencing

Exposure

Residuals

Validation

Randomness

These five categories are the same categories that are seen in other taxonomies, such as the Protection Analysis (PA) (Section 4.2.2). More detailed explanations of each will be covered in Sections 5.3.1 – 5.3.4.

5.3 VERDICT Categories

The SERVR taxonomy is a nearly complete computer attack taxonomy. However, *Incorrect Sequencing* is caused by *Improper Validation* and may also include *Improper Exposure*. Incorrect sequencing is often manifested by an error in the Time-Of-Check-To-Time-Of-Use (TOCTTOU). TOCTTOU was described in Section 4.1.6.4 as an object that should not be changed is incorrectly exposed allowing a change between the time of check and the time of use. Incorrect validation occurs to prevent this item from being accessed (or changed).

For example, McPhee [McPh1974] describes the error as *User data passed as system data*, listed as M4 in Table 5.1. This occurs when a user can call one service routine (SVC) that in turn calls a second service routine (SVC). The user could have directly called the second SVC but, by going through the first SVC, the second SVC may bypass some of the security checks because it has been called by a "trusted routine." If one couples this with the TOCTTOU error and one is able to modify the data in the first SVC, an integrity error can occur:

> An integrity exposure occurs if SVC routine B bypasses some or all validity checking based solely on the fact that it was called by another SVC routine (routine A), and if user-supplied data passed to routine B by routine A either is not *validity* checked by routine A or is *exposed* to user modification after it was validated by routine A (the TOCTTOU problem).[5]

Because *Incorrect Sequencing* is caused by *Improper Validation* and *Improper Exposure*, it too should be dropped from the final taxonomy. What remains are four categories. With *Residuals (Deallocation)* renamed to *Deallocation (Residuals)*, the taxonomy is complete. As stated above, the final result is the following:

[5]Emphasis added.

Validation

Exposure

Randomness

Deallocation

Improper

Conditions

Taxonomy

While some of the past work in computer attack taxonomies covered just operating system taxonomies, VERDICT covers all aspects of the security process, from physical security to hardware and software systems.

5.3.1 Validation

Validation is an overarching problem. In addition to the validation in operating systems, it also includes physical security. Validation was covered in greater detail in Sections 4.2.2.2, 4.1.6.1, and 4.1.6.2.

5.3.2 Exposure

Improper exposure was covered in greater detail in Section 4.2.2.8, so it is not repeated in this section.

5.3.3 Randomness

Randomness is one of the fundamental pillars of cryptography [East1994]. Without having a random source, certain aspects of cryptography such as nonces will not work. It is very difficult to generate a truly random number on a computer; thus, pseudo-random numbers are used. Simple (but breakable) generators are calculated using a number as a seed and reiterating a polynomial formula to generate the next number in the sequence. These numbers will repeat because of the modulus in the generator formula. Schneier lists other types of generators and qualities needed in a pseudo-random number generator in [Schn1996].

Venema [Vene1996] discusses non-randomness in selecting Kerberos keys (version 4) and in the X Window system's of authenticating using predictable random numbers. Both Kerberos version 4 and XDM (X Windows graphical login tool) use the non-random values of the time of day or the system's process id as their "random" value. He says, "In order to generate a secret password you need a secret to begin with." By using the non-random items, the "security" of the key is negated. When he and Dan Farmer created SATAN [Farm1995], they used the UNIX kernel to generate random events [Vene1996]. Some examples of random events they like are "keystroke timings, mouse event timings, or disk seek times."

Bruce Schneier, in a private e-mail on 6 December 1999 when asked whether Venema's candidates were random enough for security, wrote:

> They are random enough for security (see also my own Yarrow, which is a pseudo-random number generator) if they are properly implemented.
>
> The issue is that you cannot get _ true_ randomness from human key entry or computer timing events, but only from real-world events like radioactive decay. You might be interested in taking a look at the Yarrow paper, which discusses ways that PRNGs[6] fail....

The Yarrow paper is found at [Kels1999]; other papers on random numbers include Gifford [Giff1988] and Park and Miller [Park1988].

5.3.4 Deallocation

Also known as residuals, deallocation is described in more detail in Section 4.2.2.3. Besides covering the traditional residuals of data, it also includes "dumpster diving." Dumpster diving is when someone literally goes through the trash in a dumpster or similar trash-holding device to find out information about a target individual from discarded information. This discarded information could include discarded letters, credit card receipts, or anything else that could be used to get information on the target.

Bruce Schneier's section on destroying information (10.9) in [Schn1996] is worth repeating as it tells of how much effort it takes to *really* delete anything:

[6] Pseudo Random Number Generators

When you delete a file on most computers, the file isn't really deleted. The only thing deleted is an entry in the disk's index file, telling the machine that the file is there. Many software vendors have made a fortune selling file-recovery software that recovers files after they have been deleted.

And there's yet another worry: Virtual memory means your computer can read and write memory to disk any time. Even if you don't save it, you never know when a sensitive document you are working on is shipped off to disk. This means that even if you never save your plaintext data, your computer might do it for you. And driver-level compression programs like Stacker and DoubleSpace can make it harder to predict how and where information is stored on a disk.

To erase a file so that file-recovery software cannot read it, you have to physically write over all of the file's bits on the disk. According to the National Computer Security Center [National Computer Security Center, "A Guide to Understanding Data Rememberance in Automated Information Systems," NCSC-TG-025 Version 2, Sep 1991.]:

> Overwriting is a process by which unclassified data are written to storage locations that previously held sensitive data.... To purge the... storage media, the DoD requires overwriting with a pattern, then its complement, and finally with another pattern; e.g., overwrite first with 0011 0101 followed by 1100 1010, then 1001 0111. The number of times an overwrite must be accomplished depends on the storage media, sometimes on its sensitivity, and sometimes on different DoD component requirements. In any case, a purge is not complete until a final overwrite is made using unclassified data.

You may have to erase files or you may have to erase entire drives. You should also erase all unused space on your hard disk.

Most commercial programs that claim to implement the DoD standard overwrite three times: first with all ones, then with all zeros, and finally with a repeating one-zero pattern. Given my general level of paranoia, I recommend

> overwriting a deleted file seven times: the first time with all ones, the second time with all zeros, and five times with a cryptographically secure pseudo-random sequence. Recent developments at the National Institute of Standards and Technology with electron-tunneling microscopes suggest even that might not be enough. Honestly, if your data is sufficiently valuable, assume that it is *impossible* to erase data completely off magnetic media. Burn or shred the media; it's cheaper to buy media new than to lose your secrets.

If one does not burn media, data can often be recovered. Gutmann illustrates some of these methods of recovering data in magnetic and solid-state memory in [Gutm1996].

5.4 Summary

This chapter is key in this dissertation. In the first part of the chapter, ways a taxonomy can be constructed is discussed. It is determined that a computer attack taxonomy should have a structure consisting of characteristics, as opposed to a tree structure. Abstraction levels are discussed, and VERDICT is shown to exist at all levels of computing, not just at the operating system level. In order to determine root problems in computers, it is necessary to find the cause of the problems; however, some computer security problems have an effect on other systems, and cause computer security problems elsewhere in the system. A complete computer attack taxonomy needs to address these concerns.

This chapter presents a new comprehensive taxonomy of computer attacks, VERDICT: **V**alidation **E**xposure **R**andomness **D**eallocation **I**mproper **C**onditions **T**axonomy. It is shown how VERDICT's categories are derived from the summary of operating system attacks and extended to include all levels of security, not just to operating systems. The next chapter will show how previous computer attack taxonomies match with the VERDICT in order to show that VERDICT is a valid computer security taxonomy.

Chapter 6

Verification of VERDICT

If VERDICT (**V**alidation **E**xposure **R**andomness **D**eallocation **I**mproper **C**onditions **T**axonomy) covers all of the different categories of past computer attack taxonomies and if there are more categories that are in VERDICT but not in other taxonomies, it can be shown to be superior to other computer attack taxonomies. This chapter will show that VERDICT covers more topics in a more complete way. In order to verify VERDICT, the categories of VERDICT are applied to five taxonomies and one listing of "Top 10 Attacks" in the following sections: Protection Analysis (PA), Section 6.1; Research In Secured Operating Systems (RISOS), Section 6.2; Neumann and Parker's Computer Misuse Categories, Section 6.3; Neumann and Parker's Trapdoor Attacks, Section 6.4; McPhee's Integrity Flaws, Section 6.5; and SANS Top 10 Attacks, Section 6.6. A summary is presented afterwards.

6.1 VERDICT Applied to PA

As seen in Chapter 5, VERDICT was derived from the Protection Analysis (PA) taxonomy. Table 6.1 outlines the application of VERDICT to PA.

6.1.1 P1: Consistency of Data Over Time

Consistency of data over time (TOCTTOU)[1] is the result of improper validation and exposure.

[1] Time-Of-Check-To-Time-Of-Use.

Table 6.1: VERDICT Applied to PA

PA: PROTECTION ANALYSIS [BISB1978]	IMPROPER VALIDATION	IMPROPER EXPOSURE	IMPROPER RANDOMNESS	IMPROPER DEALLOCATION
P1: Consistency of Data Over Time	X	X		
P2: Validation of operands	X			
P3: Residuals				X
P4: Naming	X	X	X	
P5: Domain	X	X	X	X
P6: Serialization	X	X		
P7: Interrupted Atomic Operations	X	X		
P8: Exposed Representations		X		
P9: Queue Management Dependencies	X			
P10: Critical Operator Selection Errors	X	X	X	X

Validation: The Operating System (OS) or the Trusted Computing Base (TCB) does not prevent an invalid change from occurring.

Exposure: Some aspect of the system is improperly exposed in order for the item to be changed between the time that it was checked to the time it was used. Sometimes a permission that is set incorrectly, yielding possible the change. This is often coupled with improper validation to yield the inconsistency of data over time.

6.1.2 P2: Validation of Operands

Validation: This is an obvious match with Validation. In VERDICT, improper validation can be more than just the validation of operands (see Section 4.2.2.2 on page 105).

6.1.3 P3: Residuals

Deallocation: This is a one-to-one matching with Deallocation. In VERDICT, Deallocation is broader than those residuals resulting from actual deallocation, but residuals from all

operations (see Section 4.2.2.3 on page 105).

6.1.4 P4: Naming

Naming flaws can occur because of improper validation, exposure, or randomness.

Validation: Incorrect validation could allow a naming error to occur. For example, the OS or TCB should have checked that a filename submitted is correct and not conflicting or duplicating with another in-use name.

Exposure: Errors can result if a malicious program with the same name as an OS program is placed in the directory search path before the legitimate system program. Because the directory search path is searched sequentially, the malicious program will be executed first.

Randomness: If there is not enough resolution in the objects, such as NFS handles (see [Vene1996]), a naming error can occur.

6.1.5 P5: Domain

Domain errors are a very general class of errors. They are the *result* of any of the four VERDICT categories:

Validation: Buffer overflow can cause (or result in) *root* or *Administrator* privileges. Improper validation is also known as "incorrect enforcement at a domain crossing" in the PA report.

Exposure: The PA report gives the following as an explanation of *Exposed Representations*:

> To each user, an operating system presents an abstract machine consisting of the hardware user instruction set plus the pseudo-instructions provided through the supervisor call/invocation mechanism. The pseudo-instructions, in general, allow the user to manipulate abstract objects for which representations and operations are not provided in the basic hardware instruction set. *Inadvertent exposure by the system of the representation of the abstract object, the primitive instructions*

> *which implement the pseudo-instructions or the data structures involved in the manipulation of the abstract object can sometimes result in protected information being made accessible to the user, thereby resulting in a protection error.* (Emphasis added.)

Any improper exposure can cause a domain breach; domain is the result of improper exposure.

Randomness: Because NFS handles were not randomized properly [Vene1996], someone could send an illegitimate NFS handle and potentially cross domains. This would not be validated, because in the current protocol, all handles are of the same form and equivalent to each other; randomness, therefore, was the cause of the error.

Deallocation: When some piece of data has been left over in deallocated memory such as would yield a attribute or composition error, it causes an exposure which causes a domain error. The final PA report classify Attribute Residual Errors and Composition Residual Errors under the heading "Domain Errors" [Bisb1978].

6.1.6 P6: Serialization

Serialization errors occur when there is a lack of validation and improper exposure during some operation of the computer that had to be completed in a certain order, with no interruptions.

Validation: Some aspect of the the computer (OS, TCB), should validate that whatever caused the atomic operation to be interrupted (Section 6.1.7) or the serialization of processes to fail.

Exposure: Exposure must occur for something to interrupt something it should not.

6.1.7 P7: Interrupted Atomic Operations

Interrupted atomic operations is a subset of P6: Serialization [Bisb1978]. Because of this, the previous section, Section 6.1.6, gives the reasons why Validation and Exposure are the

root causes of this flaw.

Validation: See Section 6.1.6.

Exposure: See Section 6.1.6.

6.1.8 P8: Exposed Representations

Exposure: The VERDICT category exposure is a one-to-one (equivalent) matching of the Protection Analysis (PA) category, P8: Exposed Representations.

6.1.9 P9: Queue Management Dependencies

Queue Management Dependencies, or buffer overflows, are caused by improper validation. No or little bounds checking causes the buffer to overflow, corrupting other memory.

Validation: See Section 6.1.2 for a broader generalization of validation.

6.1.10 P10: Critical Operator Selection Errors

Critical Operator Selection Errors are another way of saying human error. This is a catch-all error. Human error can cause *all* of the VERDICT errors. Computer errors do not cause human error; human error causes computer errors. Although a human can propagate a computer error into another human error, the original computer error was caused by a human. Humans are the only thing on earth that can create. Computers cannot create; animals cannot create.[2]

If an addition operation was mistakenly substituted instead of a subtraction operation, it could cause another error of Validation, Exposure, Randomness, or Deallocation (Residuals). Therefore, this matching of VERDICT to this category (P10) is not because the VERDICT categories cause human error; human error causes the VERDICT categories.

Validation: This can be caused by human error.

[2]Beavers may build dams, bees build honeycombs, but these acts are instinctive. I believe only humans have the ability to create because they (not like animals) are created in the image of God.

Table 6.2: VERDICT Applied to RISOS

RISOS [ABBO1976]	IMPROPER VALIDATION	IMPROPER EXPOSURE	IMPROPER RANDOMNESS	IMPROPER DEALLOCATION
R1: Incomplete Parameter Validation	X			
R2: Inconsistent Parameter Validation	X			
R3: Implicit Sharing of Privileged / Confidential Data		X		X
R4: Asynchronous Validation/ Inadequate Serialization	X	X		
R5: Inadequate Identification / Authentication / Authorization	X	X	X	
R6: Violable Prohibition / Limit	X			
R7: Exploitable Logic Error	X	X	X	X

Exposure: This can be caused by human error.

Randomness: This can be caused by human error.

Deallocation: This can be caused by human error.

6.1.11 Summary

This section has shown that VERDICT can be applied to the Protection Analysis (PA) [Bisb1978] taxonomy. The VERDICT categories are the basis of all the PA categories.

6.2 VERDICT Applied to RISOS

Bishop [Bish1996c] compares the Protection Analysis (PA) [Bisb1978] and the Research in Secured Operating System (RISOS) [Abbo1976] studies and shows that the two taxonomies are equivalent. Because of this, VERDICT is applied to the RISOS operating system taxonomy categories to show that all the RISOS categories are caused by one or more of the VERDICT categories. Table 6.2 outlines the application of VERDICT to RISOS.

6.2.1 R1: Incomplete Parameter Validation

Validation: Whether incomplete or inconsistent (Section 6.2.2) parameter validation, validation is the cause.

6.2.2 R2: Inconsistent Parameter Validation

Validation: Whether incomplete (Section 6.2.1) or inconsistent parameter validation, validation is the cause.

6.2.3 R3: Implicit Sharing of Privileged/Confidential Data

What causes the sharing of data? It is privileged information located in storage accessible to an inferior process.

Exposure: As stated above, in order to facilitate sharing of data, the data must be "located in storage accessible...."

Deallocation: Bisbey notes that, "Sometimes work files and workspace are not erased when a user releases them, and another user can scavenge this "unerased blackboard" when the uncleared file space or buffer space is next assigned" [Bisb1978, Koni1976].

6.2.4 R4: Asynchronous Validation / Inadequate Sharing

Similar to P1, P6, and P7 (Sections 6.1.1, 6.1.6, and 6.1.7), the sharing of a piece of memory between an inferior and superior process requires improper validation and exposure.

Validation: Because of improper validation, data is able to be changed. This invalid validation could be in the form of a buffer overflow or other error.

Exposure: However, in order to change some data asynchronously that should not be changed, an improper exposure must occur.

6.2.5 R5: Inadequate Identification / Authentication / Authorization

Inadequate identification, authentication, and authorization are basic problems in today's security environment. They can be caused by improper validation, exposure, and randomness.

Validation: Fundamentally, improper identification, authentication, and authorization are the result of improper validation.

Exposure: This category in the RISOS taxonomy gave an example of the "bypass of controlled-access security" expounding on the ability to access the basic file access methods directly without going through validation methodologies. These file access methods are exposed where they should not be.

Randomness: "Inadequate" identification implies that there is not enough resolution, or randomness in the identification process. This is similar to the NFS handles discussed in Venema [Vene1996].

6.2.6 R6: Violable Prohibition/Limit

Violable Prohibition or Limit means that some restriction in the form of a prohibition or limit was broken or violated.

Validation: This is a one-to-one matching with the Protection Analysis (PA) P9: Queue management dependencies, as both are buffer overflows. Thus, improper validation of the set limit must occur in order for that limit to be violable.

6.2.7 R7: Exploitable Logic Error

The final error category of the RISOS taxonomy is an exploitable logic error. As in the Protection Analysis taxonomy discussed in 6.1.10, human error is the cause of all errors. Since this category is such a broad error, all VERDICT categories apply.

Validation: In the RISOS report [Bisb1978], it discusses a logic error where the user can break out of a login routine if a "NAK"[3] is pressed. Another example given is using a full-word arithmetic on a system returned half-word address. An overflow can occur, thus changing the state to the control state.

Exposure: Instruction side effects can cause flaws; coupled with validation, exposures can cause sequencing and TOCTTOU errors.

Randomness: From assigning the same tape block simultaneously to multiple users to other logic errors caused by human error, the lack of randomness can cause logic errors [Bisb1978, Koni1976].

Deallocation: The half-word address overflow discussed under the validation heading above shows that deallocation is a root cause of exploitable errors.

6.2.8 Summary

This section has shown that VERDICT can be applied to the Research In Secured Operating Systems (RISOS) [Abbo1976] taxonomy. The VERDICT categories are the basis of all the RISOS categories.

6.3 VERDICT Applied to Neumann and Parker's Computer Misuse Categories

This section shows the combination of VERDICT categories that comprise the 26 types of computer misuse (CM) attacks given by Neumann and Parker [Neum1995]. After each misuse type, a brief description of the type of computer misuse is given from the original chart in Neumann [Neum1995]; these are the same descriptions given in Table 3.4 on page 54. Following the description, a short defense is given to show how the VERDICT categories are the cause of each of the computer misuse types.

[3]Not AcKnowledge.

Table 6.3: VERDICT Applied to Computer Misuse Categories

COMPUTER MISUSE CATEGORIES	IMPROPER VALIDATION	IMPROPER EXPOSURE	IMPROPER RANDOMNESS	IMPROPER DEALLOCATION
CM1: Visual Spying		X		
CM2: Misrepresentation	X			
CM3: Physical Scavenging				X
CM4: Logical Scavenging				X
CM5: Eavesdropping		X		
CM6: Interference		X		
CM7: Physical Attack		X		
CM8: Physical Removal	X	X		
CM9: Impersonation	X			
CM10: Piggybacking Attacks	X			
CM11: Spoofing Attacks	X			
CM12: Network Weaving	X			
CM13: Trojan Horse Attacks	X	X	X	X
CM14: Logic Bombs	X	X	X	X
CM15: Malevolent Worms	X	X	X	X
CM16: Virus Attacks	X	X	X	X
CM17: Trapdoor Attacks	X	X	X	X
CM18: Authorization Attacks	X	X	X	
CM19: Basic Active Misuse	X			
CM20: Incremental Attacks				X
CM21: Denials of Service	X	X		
CM22: Browsing		X		
CM23: Inference, Aggregation		X		
CM24: Covert Channels		X		

6.3.1 CM1: Visual Spying

Observing of keystrokes or screens

Exposure: In order for visual spying to occur, that which is spied upon must be exposed.

6.3.2 CM2: Misrepresentation

Deceiving operators and users

Validation: Proper validation would prevent this problem from occurring.

6.3.3 CM3: Physical Scavenging

Dumpster-diving for printout

Deallocation: Deallocation, or residuals, is what someone dumpster-diving is looking for. Printouts may not be the only target; media may be discarded with data remaining on it.

6.3.4 CM4: Logical Scavenging

Examining discarded/stolen media

Deallocation: Once the media is obtained, either by dumpster-diving or outright theft, the data is examined to ascertain whether any data remains that can be read and is useful.

6.3.5 CM5: Eavesdropping

Intercepting electronic or other data

Exposure: If it is possible to eavesdrop on communications, that data is improperly exposed.

6.3.6 CM6: Interference

Jamming, electronic or otherwise

Exposure: Similarly, if it is possible to jam the communications, that data is improperly exposed.

6.3.7 CM7: Physical Attack

Damaging or modifying equipment, power

Exposure: If the object that is attacked was not supposed to be accessed, that object is improperly exposed.

6.3.8 CM8: Physical Removal

Removing equipment and storage media

Validation: If there was an improper checking or validation of the equipment, it could be removed.

Exposure: The equipment must be available to the thief in order to be removed.

6.3.9 CM9: Impersonation

Using false identities external to computer systems

Validation: Improper validation allows impersonation.

6.3.10 CM10: Piggybacking Attacks

Usurping communication lines, workstations

Validation: If the communication lines are not guarded against physical tampering, piggybacking can occur. If the piggyback attack is purely electronic, proper validation can prevent the attack.

6.3.11 CM11: Spoofing Attacks

Using playback, creating bogus nodes and systems

Validation: Similar to CM9: Impersonation, improper validation allows spoofing attacks.

6.3.12 CM12: Networking Weaving

Masking physical whereabouts or routing

Validation: Being able to hide oneself in the network is caused by improper validation.

6.3.13 CM13: Trojan Horse Attacks

Implanting malicious code, sending letter bombs

Validation: Validation, done with either a closed test environment (sandbox) or other form of validation can prevent a Trojan horse attack.

Exposure: Improper exposure can cause the Trojan horse to planted, and the damage of the attack to be great if the exposure is not limited.

Randomness: Improper randomness may be a cause of a Trojan horse.

Deallocation: A Trojan horse can use residuals as part of the attack.

6.3.14 CM14: Logic Bombs

Setting time or event bombs (a form of Trojan horse)

Validation: If one views a logic bomb as a form of a Trojan horse, the same four VERDICT categories apply. However, the logic bomb does not have to be disguised as a benign program like a Trojan horse does. Because of this, I do not believe that a logic bomb is just another form of a Trojan horse. However, just as a Trojan horse, a logic bomb is such a large class of possible manifestations of vulnerabilities, that all categories of VERDICT may apply.

Exposure: See Exposure under CM13: Trojan Horse Attacks.

Randomness: See Randomness under CM13: Trojan Horse Attacks.

Deallocation: See Deallocation under CM13: Trojan Horse Attacks.

6.3.15 CM15: Malevolent Worms

Acquiring distributed resources

Validation: Similar in complexity as Trojan Horse Attacks and Logic Bombs, all combinations of VERDICT categories may apply.

Exposure: See Exposure under CM13: Trojan Horse Attacks and CM14: Logic Bombs.

Randomness: See Randomness under CM13: Trojan Horse Attacks and CM14: Logic Bombs.

Deallocation: See Deallocation under CM13: Trojan Horse Attacks and CM14: Logic Bombs.

6.3.16 CM16: Virus Attacks

Attaching to programs and replicating

Validation: Similar in complexity as Trojan Horse Attacks, Logic Bombs, and Malevolent worms, all combinations of VERDICT categories may apply.

Exposure: Similar in complexity as Trojan Horse Attacks, Logic Bombs, and Malevolent Worms, all combinations of VERDICT categories may apply.

Randomness: Similar in complexity as Trojan Horse Attacks, Logic Bombs, and Malevolent Worms, all combinations of VERDICT categories may apply.

Deallocation: Similar in complexity as Trojan Horse Attacks, Logic Bombs, and Malevolent Worms, all combinations of VERDICT categories may apply.

6.3.17 CM17: Trapdoor Attacks

Utilizing existing flaws

Validation: This is such a generalized "computer misuse" that all combinations of VERDICT categories may apply.

Exposure: This is such a generalized "computer misuse" that all combinations of VERDICT categories may apply.

Randomness: This is such a generalized "computer misuse" that all combinations of VERDICT categories may apply.

Deallocation: This is such a generalized "computer misuse" that all combinations of VERDICT categories may apply.

6.3.18 CM18: Authorization Attacks

Password cracking, hacking tokens

Validation: "Authorization" in authorization attacks means validation. In addition, passwords and tokens are tools of proper authorization.

Exposure: In order to get the file containing the password or any tokens, they must be improperly exposed.

Randomness: Improper randomness in passwords can lead to the cracking of passwords.

6.3.19 CM19: Basic Active Misuse

Creating, modifying, using, denying service, entering false or misleading data

Validation: All of these fall under the category of improper validation to ensure proper operation.

6.3.20 CM20: Incremental Attacks

Using salami attacks

Deallocation: Deallocation, or residuals, are what is added together or collected in salami attacks (see Section A.1.17 on page 246 for more information about salami attacks).

6.3.21 CM21: Denials of Service

Perpetrating saturation attacks

Validation: Proper validation of communications may thwart denials of service.

Exposure: In order for any service to be denied, it must be exposed. If it is exposed incorrectly, a DoS can occur.

6.3.22 CM22: Browsing

Making random or selective searches

Exposure: If someone should not see the data, the data should not be accessible, or improperly exposed.

6.3.23 CM23: Inference, Aggregation

Exploiting database inferences and traffic analysis

Exposure: Similar to CM22: Browsing, if someone should not have information in the database or traffic to aggregate, infer, or analyze, it should not be improperly exposed.

6.3.24 CM24: Covert Channels

Exploiting covert channels or other data leakage

Exposure: Whether timing or storage covert channels, improper exposure of them allows them to be exploited.

6.3.25 CM25: Inactive Misuse

Willfully failing to perform expected duties, or committing errors of omission

Validation: When a system administrator, or someone of trusted authority, is collaborating with an end user in a nefarious operation, it is improper validation of the character of the administrator (and the end user) that is in question. However, all combinations of VERDICT categories may be applicable.

Exposure: This is such a generalized "computer misuse" that all combinations of VERDICT categories may apply.

Randomness: This is such a generalized "computer misuse" that all combinations of VERDICT categories may apply.

Deallocation: This is such a generalized "computer misuse" that all combinations of VERDICT categories may apply.

6.3.26 CM26: Indirect Misuse

Preparing for subsequent misuses, as in off-line preencryptive matching, factoring large numbers to obtain private keys, autodialer scanning

Validation: This is such a large computer misuse category. Similar to CM25: Inactive Misuse, all combinations of VERDICT categories may be applicable.

Exposure: This is such a large computer misuse category. Similar to CM25: Inactive Misuse, all combinations of VERDICT categories may be applicable.

Randomness: This is such a large computer misuse category. Similar to CM25: Inactive Misuse, all combinations of VERDICT categories may be applicable.

Deallocation: This is such a large computer misuse category. Similar to CM25: Inactive Misuse, all combinations of VERDICT categories may be applicable.

Table 6.4: VERDICT Applied to Trapdoor Attacks

NEUMANN/PARKER TRAPDOOR ATTACKS [NEUM1995]	IMPROPER VALIDATION	IMPROPER EXPOSURE	IMPROPER RANDOMNESS	IMPROPER DEALLOCATION
TD1: Improper Identification / Authentication	X	X		
TD2: Improper Initialization / Allocation	X	X	X	
TD3: Improper Finalization (Termination / Deallocation)				X
TD4: Improper Authentication / Validation	X			
TD5: Naming Flaws, Confusions, Aliases	X	X	X	
TD6: Improper Encapsulation, Such as Accessible Internals		X		
TD7: Asynchronous Flaws, Such as Atomicity Anomalies	X	X		
TD8: Other Logic Errors	X	X	X	X

6.3.27 Summary

This section has shown that Neumann and Parker's Types of Computer Misuse (CM) are combinations of the four categories of VERDICT.

6.4 VERDICT Applied to Neumann and Parker's Trapdoor Attacks

From Table 4.10 shown on page 140, Neumann and Parker's trapdoor attacks [Neum1995] are equivalent to the Protection Analysis (PA) study [Bisb1978]. The VERDICT categories are applied to Neumann and Parker as well because of the generality of the trapdoor attacks and are shown in Table 6.4.

6.4.1 TD1: Improper Identification/Authentication

Validation: Improper identification and authentication match one-to-one with improper validation.

6.4.2 TD2: Improper Initialization/Allocation

Validation: Improper initialization and allocation determines what initial domain an object resides in, and so TD2 matches P5, which has all VERDICT categories. However, since this is allocation and not deallocation, the deallocation (residual) VERDICT category more precisely matches TD3, improper finalization, as seen below.

Exposure: Initial domain can be corrupted by improper exposure.

Randomness: Allocation can be corrupted by improper randomness.

6.4.3 TD3: Improper Finalization (Termination and Deallocation)

Deallocation: As stated above in the discussion of TD2, improper initialization and allocation, improper finalization (termination and deallocation) matches one-to-one with Deallocation (Residuals), as it is also a match with P3, Residuals.

6.4.4 TD4: Improper Authentication/Validation

Validation: This is a one-to-one match with P2/P2 (Validation of operands, Queue management dependencies); so also it precisely matches the VERDICT category of improper validation.

6.4.5 TD5: Naming Flaws, Confusions, Aliases

Validation: Naming flaws, confusions, and aliases is a precise one-to-one match to P4, naming. Because of this, the reasons given for P4 to match with improper validation, exposure, and randomness are given here. Improper validation may cause a naming error to occur.

Exposure: Similar to the reason of P4, naming, improper exposure can cause naming flaws or aliasing errors.

Randomness: Improper randomness can also cause naming flaws, as seen with the NFS handles [Vene1996].

6.4.6 TD6: Improper Encapsulation, such as Accessible Internals

Exposure: This is a one-to-one mapping to the VERDICT category Improper Exposure.

6.4.7 TD7: Asynchronous Flaws, such as Atomicity Anomalies

Validation: Asynchronous flaws, such as atomicity anomalies are a one-to-one match of P6 and P7, Serialization and Interrupted Atomic Operations. In addition, P1, Consistency of Data Over Time, also is tied into this category. All of these, as stated above, have as VERDICT categories improper validation and exposure.

Exposure: In order to have asynchronous flaws, objects must be improperly exposed. See discussion above on TD7 and improper validation.

6.4.8 TD8: Other Logic Errors

Validation: This error, like P10, is a catch-all error. Because of this, any combination of VERDICT categories may apply.

Exposure: Similar to P10: Critical Operator Selection Errors Inactive Misuse, all combinations of VERDICT categories may be applicable.

Randomness: Similar to P10: Critical Operator Selection Errors Inactive Misuse, all combinations of VERDICT categories may be applicable.

Deallocation: Similar to P10: Critical Operator Selection Errors Inactive Misuse, all combinations of VERDICT categories may be applicable.

6.4.9 Summary

This section has shown that the Trapdoor Attacks of Neumann and Parker (TD1 – TD8) are derived from combinations of VERDICT categories [Neum1995].

Table 6.5: VERDICT Applied to McPhee

MCPHEE [MCPH1974]	IMPROPER VALIDATION	IMPROPER EXPOSURE	IMPROPER RANDOMNESS	IMPROPER DEALLOCATION
M1: System Data in the User Area	X			
M2: Non-Unique Identification of System Resources	X		X	
M3: System Violation of Storage Protection	X			
M4: User Data Passed As System Data	X	X		
M5: User-Supplied address of Protected Control Blocks	X	X		
M6: Concurrent User of Serial Resources	X	X		
M7: Uncontrolled Sensitive System Resources	X	X		

6.5 VERDICT Applied to McPhee's Integrity Flaws

This section shows that McPhee's Integrity Flaws (M1 – M7) can be derived from combinations of VERDICT categories. It is to be noted that Improper Validation is a cause of all of McPhee's categories; in addition, Improper Deallocation is not a cause of any of McPhee's categories. See Table 6.5.

6.5.1 M1: System Data in the User Area

Validation: If system data appears where it should not (i.e., the user area), it is the result of improper validation. Validation should have stopped the improper system data from being accessible. It is not because of improper exposure that the data is there. Because of improper validation, there is improper exposure.

6.5.2 M2: Nonunique Identification of System Resources

Validation: Improper validation must have caused the wrong identification of the said system resources.

Randomness: "Nonunique identification" is improper randomness.

6.5.3 M3: System Violation of Storage Protection

Validation: McPhee states, "*System violation of storage protection* is a problem where a system routine, operating in one of the privileged system keys (0–7), performs a store or fetch operation in behalf of a user routine *without adequately validating* that a user-specified location actually is in an area accessible to him." (Latter emphasis added) [McPh1974]. There is a system violation of storage... without validation.

6.5.4 M4: User Data Passed as System Data

Validation: The problem is one where:

> ...it is possible for an unauthorized user program to use one SVC routine (routine A) to invoke a second SVC routine (routine B) that the problem program could have invoked directly. An integrity exposure occurs if SVC routine B bypasses some of all validity checking based solely on the fact that it was called by another SVC routine (routine A), and if user-supplied data passed to routine B by routine A either is not *validity* checked by routine A or is *exposed* to user modification after it was validated by routine A (the TOCTTOU problem). This problem does not exist if the user calls SVC routine B directly because the validity checking will be performed on the basis of the caller being an unauthorized program. The confusion arises because of the various cases in the system where SVC routines operating in their own behalf invoke other SVC routines to perform operations that would not and should not, withstand the normal validity checking applied to unauthorized programs. The problem is identify the case where an SVC is

operating in a user's behalf — that is, with *unvalidated, user-supplied data that should undergo normal validity checking.* (Emphasis added) [McPh1974].

There is improper validity checking, and sometimes improper exposure.

Exposure: See the above discussion on the relationship between improper validation and improper exposure.

6.5.5 M5: User-Supplied Address of Protected Control Blocks

McPhee argues that if the user is able to supply an address of a control block, it could trigger a validity error: "Inadequate *validity* checking in this situation creates an integrity *exposure* since the user program can provide its own (counterfeit) control block in palace of the system control block and thereby cause a virtually unlimited array of integrity problems depending on exactly what sensitive data the system may be keeping in the control block involved" (Emphasis added) [McPh1974].

Validation: As McPhee states above, it is "inadequate validity checking" that causes the error.

Exposure: An improper exposure is the result of improper validation, but improper exposure can also be a cause. McPhee continues, "...In such cases, *the user must not be permitted to identify* more than one of the set of protected control blocks describing that allocation unless there exists a mechanism whereby information contained in one of the blocks..." (Emphasis added) [McPh1974]. Further, "...If the user were allowed to provide the address..." [McPh1974]. Improper exposure can be the cause of this integrity flaw.

6.5.6 M6: Concurrent Use of Serial Resources

Improper sequencing is the result of a combination of improper validation and improper exposure. Time-Of-Check-To-Time-Of-Use (TOCTTOU) is related to this problem.

Validation: Part of the protection system should have prevented the concurrent use of serial resources.

Exposure: In order to concurrently use the serial resources, improper exposure of the resources may have led to the vulnerability.

6.5.7 M7: Uncontrolled Sensitive System Resources

These "system resources" are the equivalent to the set user-id (SUID) programs of UNIX today: "Because there has been no way in the past for the control program to effectively differentiate the class of programs that require such special services from the totality of user programs, these special services have generally been made available to all user programs without restriction. The lack of restriction on such sensitive services results in system integrity problems" [McPh1974].

Validation: There is no validation to see who should be able to correctly access these programs; hence there is improper validation.

Exposure: Because of improper exposure, these sensitive system resources are accessible to objects that should not have access to them.

6.5.8 Summary

This section has shown that McPhee's seven integrity errors (M1 – M7) can be derived from combinations of VERDICT categories.

6.6 VERDICT Applied to SANS Top 10 Attacks

SANS, System Administrators and Network Security, is an organization that publishes a "working" document[4] written by those in academia, industry, and government that lists ten computer attacks that are most widely seen. Named the Top 10 Attacks, it is continually updated. The top ten attacks as of November 2000 are given in Table 6.6 outlines the application of VERDICT to the SANS Top 10 Attacks.

[4]Presently (November 2000) available on their web site at http://www.sans.org or other mirrored locations.

Table 6.6: VERDICT Applied to SANS Top 10 Attacks

SANS TOP 10 ATTACKS	IMPROPER VALIDATION	IMPROPER EXPOSURE	IMPROPER RANDOMNESS	IMPROPER DEALLOCATION
SANS1: BIND Weaknesses Allow Immediate Root Compromise	**X**			
SANS2: Vulnerable CGI Programs Installed on Web Servers	**X**	**X**		
SANS3: Remote Procedure Call (RPC) Weaknesses Yields Root	**X**	**X**		
SANS4: Remote Data Services Hole in MS Internet Info Server	**X**	**X**		
SANS5: Sendmail and MIME Buffer Overflows and Pipe Attacks	**X**	**X**		
SANS6: sadmind and mountd	**X**			
SANS7: Global File Sharing in NetBIOS, NFS, Appleshare		**X**		
SANS8: User IDs, Esp. Root / Admin with No or Weak Passwords			**X**	
SANS9: IMAP and POP Buffer Overflow or Incorrect Configuration	**X**	**X**		
SANS10: Default SNMP Community Strings “Public” and “Private”			**X**	

In Sections 6.6.1 – 6.6.10, the error is described from version 1.30 of the report[5] and then VERDICT categories applicable to the error are discussed. Full quotes are used because the document is not an archival document such as a journal article or conference report.

There is not much information in the document itself describing the technical details of the attack, but general themes of improper validation, exposure, randomness are seen. It is noted that none of the ten attacks described here have as their causes deallocation (residuals). This does not exclude deallocation as an area from which errors can be formed.

6.6.1 BIND Weaknesses Allow Immediate Root Compromise

The SANS report gives the following description of this error:

> **BIND weaknesses: nxt, qinv and in.named allow immediate root compromise.**
>
> The Berkeley Internet Name Domain (BIND) package is the most widely used implementation of Domain Name Service (DNS) — the critical means by which we all locate systems on the Internet by name (e.g., www.sans.org) without having to know specific IP addresses — and this makes it a favorite target for attack. Sadly, according to a mid-1999 survey, about 50% of all DNS servers connected to the Internet are running vulnerable versions of BIND. In a typical example of a BIND attack, intruders erased the system logs, and installed tools to gain administrative access. They then compiled and installed IRC utilities and network scanning tools, which they used to scan more than a dozen class-B networks in search of additional systems running vulnerable versions of BIND. In a matter of minutes, they had used the compromised system to attack hundreds of remote systems abroad, resulting in many additional successful compromises. This illustrates the chaos that can result from a single vulnerability in the software for ubiquitous Internet services such as DNS.

[5]This version is dated November 17, 2000.

Validation: The description of the BIND problem in the April 8, 1998 CERT Advisory CA-98.05 reads as follows:[6]

> I. Description This advisory describes three distinct problems in BIND. Topic 1 describes a vulnerability that may allow a remote intruder to gain root access on your name server or to disrupt normal operation of your name server. Topics 2 and 3 deal with vulnerabilities that can allow an intruder to disrupt your name server. Detailed descriptions of each problem and its solutions are included in the individual sections on each topic....
>
> **Topic 1: Inverse Query Buffer Overrun in BIND 4.9 and BIND 8 Releases**
>
> 1.A. Description BIND 4.9 releases prior to BIND 4.9.7 and BIND 8 releases prior to 8.1.2 do not properly bounds check a memory copy when responding to an inverse query request. An improperly or maliciously formatted inverse query on a TCP stream can crash the server or allow an attacker to gain root privileges.
>
> **Topic 2: Denial-of-Service Vulnerabilities in BIND 4.9 and BIND 8 Releases**
>
> 2.A. Description BIND 4.9 releases prior to BIND 4.9.7 and BIND 8 releases prior to 8.1.2 do not properly bounds check many memory references in the server and the resolver. An improperly or maliciously formatted DNS message can cause the server to read from invalid memory locations, yielding garbage record data or crashing the server. Many DNS utilities that process DNS messages (e.g., dig, nslookup) also fail to do proper bounds checking....
>
> **Topic 3: Denial-of-Service Vulnerability in BIND 8 Releases**
>
> 3.A. Description Assume that the following self-referential resource record is in the cache on a name server:
>
> ```
> foo.example.    IN    A    CNAME    foo.example.
> ```
>
> The actual domain name used does not matter; the important thing is that the target of the CNAME is the same name. The record could be in the cache either

[6]Presently found on the WWW at: http://www.cert.org/advisories/CA-98.05.bind_ problems.html

because the server was authoritative for it or because the server is recursive and someone asked for it. Once this record is in the cache, issuing a zone transfer request using its name (e.g., "dig@my_ nameserver foo.example. axfr") will cause the server to abort().

Most sites will not contain such a record in their configuration files. However, it is possible for an attacker to engineer such a record into the cache of a vulnerable nameserver and thus cause a denial of service.

It is seen from this document that the vulnerability is improper validation. The first two topics are caused by buffer overflows. The third is caused by a self-referential record (i.e., a record that refers to itself). Proper validation could fix these vulnerabilities.

6.6.2 Vulnerable CGI Programs Installed on Web Servers

The SANS report gives the following description of this error:

Vulnerable CGI programs and application extensions (e.g., ColdFusion) installed on web servers.

Most web servers support Common Gateway Interface (CGI) programs to provide interactivity in web pages, such as data collection and verification. Many web servers come with sample CGI programs installed by default. Unfortunately, many CGI programmers fail to consider ways in which their programs may be misused or subverted to execute malicious commands. Vulnerable CGI programs present a particularly attractive target to intruders because they are relatively easy to locate, and they operate with the privileges and power of the web server software itself. Intruders are known to have exploited vulnerable CGI programs to vandalize web pages, steal credit card information, and set up back doors to enable future intrusions, even if the CGI programs are secured. When Janet Reno's picture was replaced by that of Adolph Hitler at the Department of Justice web site, an in-depth assessment concluded that a CGI hole was the most probable avenue of compromise. Allaire's ColdFusion is a web server application

package which includes vulnerable sample programs when installed. As a general rule, sample programs should always be removed from production systems.

Validation: CGI programs suffer from the same fate as do all generalized interpretive languages that can be accessed and run externally. Because the languages are generalized, vulnerabilities can be crafted and run.

Exposure: The fact that the CGI programs are accessible is an improper exposure.

6.6.3 Remote Procedure Call (RPC) Weaknesses Yields Root

The SANS report gives the following description of this error:

> **Remote Procedure Call (RPC) weaknesses in rpc.ttdbserverd (ToolTalk), rpc.cmsd (Calendar Manager), and rpc.statd that allow immediate root compromise**
>
> Remote procedure calls (RPC) allow programs on one computer to execute programs on a second computer. They are widely-used to access network services such as shared files in NFS. Multiple vulnerabilities caused by flaws in RPC, are being actively exploited. There is compelling evidence that the vast majority of the distributed denial of service attacks launched during 1999 and early 2000 were executed by systems that had been victimized because they had the RPC vulnerabilities. The broadly successful attack on U.S. military systems during the Solar Sunrise incident also exploited an RPC flaw found on hundreds of Department of Defense systems.

Validation: According to CERT Incident Note IN-99-04,[7]

the "Vulnerabilities we have seen exploited as a part of these attacks include: CA-99-08 — Buffer Overflow Vulnerability in rpc.cmsd...." This indicates that the RPC weaknesses were in part caused by a buffer overflow, an improper vulnerability.

[7]Presently found at http://www.cert.org/incident_ notes/IN-99-04.html

Exposure: IN-99-04 continues: "CA-99-05 — Vulnerability in statd exposes vulnerability in automountd...."[8] CERT Advisory CA-99-05 describes the rpc.statd vulnerability as follows:

> The vulnerability in rpc.statd allows an intruder to call arbitrary rpc services with the privileges of the rpc.statd process. The called rpc service may be a local service on the same machine or it may be a network service on another machine. Although the form of the call is constrained by rpc.statd, if the call is acceptable to another rpc service, the other rpc service will act on the call as if it were an authentic call from the rpc.statd process.

Any valid rpc.statd request will be allowed to run. This improper exposure is coupled with the improper validation.

6.6.4 Remote Data Services Hole in MS Internet Info Server

The SANS report gives the following description of this error:

> **RDS security hole in the Microsoft Internet Information Server (IIS).**
>
> Microsoft's Internet Information Server (IIS) is the web server software found on most web sites deployed on Microsoft Windows NT and Windows 2000 servers. Programming flaws in IIS's Remote Data Services (RDS) are being employed by malicious users to run remote commands with administrator privileges. Some participants who developed the "Top Ten" list believe that exploits of other IIS flaws, such as .HTR files, are at least as common as exploits of RDS. Prudence dictates that organizations using IIS install patches or upgrades to correct all known IIS security flaws when they install patches or upgrades to fix the RDS flaw.

Validation: CVE[9] 1999-1011 gives the description as follows: "The Remote Data Service (RDS) DataFactory component of Microsoft Data Access Components (MDAC) in IIS 3.x

[8] Presently found at http://www.cert.org/advisories/CA-99-05-statd-automountd.html
[9] Common Vulnerabilities and Exposures, see http://cve.mitre.org

and 4.x exposes unsafe methods, which allows remote attackers to execute arbitrary commands." Running arbitrary programs is caused by a lack of validation.

Exposure: In order to run these programs, they must be accessible. Because they are accessible, it is another example of improper exposure.

6.6.5 Sendmail and MIME Buffer Overflows and Pipe Attacks

The SANS report gives the following description of this error:

> **Sendmail and MIME buffer overflows as well as pipe attacks that allow immediate root compromise.**
>
> Sendmail is the program that sends, receives, and forwards most electronic mail processed on UNIX and Linux computers. Sendmail's widespread use on the Internet makes it a prime target of attackers. Several flaws have been found over the years. The very first advisory issued by CERT/CC in 1988 made reference to an exploitable weakness in sendmail. In one of the most common exploits, the attacker sends a crafted mail message to the machine running Sendmail, and Sendmail reads the message as instructions requiring the victim machine to send its password file to the attacker's machine (or to another victim) where the passwords can be cracked.

Validation: A buffer overflow of sendmail and MIME are also caused by improper validation.

Exposure: Old versions of sendmail would run mail data as programs through a "debug" command. This exposure was improper.

6.6.6 sadmind and mountd

The SANS report gives the following description of this error:

sadmind and mountd

Sadmind allows remote administration access to Solaris systems, providing graphical access to system administration functions. Mountd controls and arbitrates access to NFS mounts on UNIX hosts. Buffer overflows in these applications can be exploited allowing attackers to gain control with root access.

Validation: Buffer overflows are caused by improper validation.

6.6.7 Global File Sharing in NetBIOS, NFS, Appleshare

The SANS report gives the following description of this error:

> **Global file sharing and inappropriate information sharing via NetBIOS and Windows NT ports 135-¿139 (445 in Windows2000), or UNIX NFS exports on port 2049, or Macintosh Web sharing or AppleShare/IP on ports 80, 427, and 548.**
>
> These services allow file sharing over networks. When improperly configured, they can expose critical system files or give full file system access to any hostile party connected to the network. Many computer owners and administrators use these services to make their file systems readable and writeable in an effort to improve the convenience of data access. Administrators of a government computer site used for software development for mission planning made their files world readable so people at a different government facility could get easy access. Within two days, other people had discovered the open file shares and stolen the mission planning software.
>
> When file sharing is enabled on Windows machines they become vulnerable to both information theft and certain types of quick-moving viruses. A recently released virus called the 911 Worm uses file shares on Windows 95 and 98 systems to propagate and causes the victim's computer to dial 911 on its modem. Macintosh computers are also vulnerable to file sharing exploits.

The same NetBIOS mechanisms that permit Windows File Sharing may also be used to enumerate sensitive system information from NT systems. User and Group information (usernames, last logon dates, password policy, RAS information), system information, and certain Registry keys may be accessed via a "null session" connection to the NetBIOS Session Service. This information is typically used to mount a password guessing or brute force password attack against the NT target.

Exposure: This is in pure form improper exposure of file and file systems.

6.6.8 User IDs, Esp. Root/Admin with No or Weak Passwords

The SANS report gives the following description of this error:

> **User IDs, especially root/administrator with no passwords or weak passwords.**
>
> Some systems come with "demo" or "guest" accounts with no passwords or with widely-known default passwords. Service workers often leave maintenance accounts with no passwords, and some database management systems install administration accounts with default passwords. In addition, busy system administrators often select system passwords that are easily guessable ("love," "money," "wizard" are common) or just use a blank password. Default passwords provide effortless access for attackers. Many attackers try default passwords and then try to guess passwords before resorting to more sophisticated methods. Compromised user accounts get the attackers inside the firewall and inside the target machine. Once inside, most attackers can use widely-accessible exploits to gain root or administrator access.

Randomness: Better, more random passwords ("non-guessable") is the cause of this problem.

6.6.9 IMAP and POP Buffer Overflow or Incorrect Configuration

The SANS report gives the following description of this error:

> **IMAP and POP buffer overflow vulnerabilities or incorrect configuration.**
>
> IMAP and POP are popular remote access mail protocols, allowing users to access their e-mail accounts from internal and external networks. The "open access" nature of these services makes them especially vulnerable to exploitation because openings are frequently left in firewalls to allow for external e-mail access. Attackers who exploit flaws in IMAP or POP often gain instant root-level control.

Validation: A buffer overflow is caused by improper validation.

Exposure: Incorrect configuration, caused by improper validation that a correct configuration is in place, causes improper exposures.

6.6.10 Default SNMP Community Strings "Public" and "Private"

The SANS report gives the following description of this error:

> **Default SNMP community strings set to 'public' and 'private.'**
>
> The Simple Network Management Protocol (SNMP) is widely used by network administrators to monitor and administer all types of network-connected devices ranging from routers to printers to computers. SNMP uses an unencrypted "community string" as its only authentication mechanism. Lack of encryption is bad enough, but the default community string used by the vast majority of SNMP devices is "public", with a few "clever" network equipment vendors changing the string to "private". Attackers can use this vulnerability in SNMP to reconfigure or shut down devices remotely. Sniffed SNMP traffic can reveal a great deal about the structure of your network, as well as the systems and devices attached to it. Intruders use such information to pick targets and plan attacks.

Randomness: Similar to the eighth "top ten" attacks, having better passwords, or ones with higher entropy would remove this improper randomness.

6.6.11 Summary

This section has applied the four categories of VERDICT to the present (November 17, 2000) "top ten" attacks. It has shown that VERDICT can be applied to attacks of today as easily as those attacks of days past.

6.7 Summary

In this chapter, VERDICT is applied to the past taxonomies of the Protection Analysis (PA), Research in Secured Operating Systems (RISOS), Neumann and Parker's Computer Misuse Categories and Trapdoor Attacks, McPhee's Integrity Flaws, and the SANS Top Ten Attacks. This chapter verifies that all categories in the previous taxonomies can be attributed to one or more of VERDICT categories.

VERDICT is a simpler system than previous taxonomies. It has fewer categories and is able to be applied to all abstraction levels, from operating systems to the system as a whole. Because all the categories of other taxonomies are equivalent to the categories of VERDICT, VERDICT is a superior system.

In the next chapter, methodologies and algorithms for each of the four VERDICT improper conditions are discussed to show how one can apply the four categories to all levels of abstraction over the entire system down to actual programming code.

Chapter 7

Methodologies of VERDICT

This chapter discusses and outlines methodologies and algorithms for application to systems of each of the four categories of VERDICT: Validation, Exposure, Randomness, and Deallocation (Residuals). With these algorithms and methodologies, one can determine what the weaknesses of a system are. There is a need for these methodologies in order to post-analyze attacks (see Chapter 6) and to predict weaknesses and vulnerabilities of future protocols (see Chapter 8). Methodologies consisting of algorithms to find errors and checklists of items to verify correctness are developed and presented.

7.1 Validation

Of all VERDICT categories, Validation is presently the broadest category of error. This is true both for the breadth of its coverage and the number of actual validation errors present today. It is broader than just "access control" that attempts to prevent unauthorized kernel access. In addition to the validation of objects in a program, it includes physical and site procedural security.

There are two types of levels of abstraction that one can look at to determine if any errors exists. The first is at a lower level, looking at the validation of code itself. The second is at a higher level, looking at the system as a whole. The system referred to can either be as small as a protocol or an entire system. Algorithms for validation of code is presented in Section 7.1.1, and methodologies for evaluating the security of a protocol are presented in

Section 7.1.2.

7.1.1 Validation of Code

There are two types of validation in code that can be discussed. The first is critical conditions, and the second is buffer overflows. They will be discussed in Sections 7.1.1.1 and 7.1.1.2 below. For critical conditions, the algorithms developed by Bisbey et al. [Bisb1978] are presented.

7.1.1.1 Critical Conditions

Carlstedt and Bisbey studied critical conditions in programs and how it needs to be ensured that those conditions do not cause improper validation [Carl1976, Bisb1978]. Critical conditions can be either abstract or concrete; they are the conditions that must be validated in order to assure that the routine or program is secure. Carlstedt called the procedure "validation of critical conditions" [Carl1976].

To determine the critical conditions, all variables used in programs must be determined. Next, a relationship between input and output variables must be determined; Bisbey calls this, "data dependency analysis" [Bisb1978]. A data flow graph for mappings of formal to actual parameters and global variables must be created. However, one cannot do a complete analysis with just static analysis.

Carlstedt and Bisbey give two ways to accomplish the validation of critical operands methodology: Outside-to-Inside and Inside-to-Outside [Carl1976, Bisb1978].

7.1.1.1.1 Outside-to-Inside Validation Methodology The purpose of this methodology is to prevent operators from operating on invalid operands. One starts with all data input entry points and proceeds with the control flow of the program to other operands. When the innermost operators are reached and validated, the validation ends because there will not be operators operating on invalid operands.

The outside-to-inside algorithm from Bisbey is the following [Bisb1978]:

> A purpose of validation is to prevent privileged system operators from operating

on incorrect/unvalidated operands. Externally-supplied user data constitutes such a source. They enter the system in a variety of ways. Direct or indirect parameters to supervisor subroutines constitute one large source. Others include mutually agreed upon mail boxes, communications areas, or files. The operating system is responsible for insuring that this data is properly checked before a system operator uses it.

One approach for determining the adequacy of validation is to begin at the user/system interface and calculate the validity conditions for all user-supplied data at various operators within the system. This can be done as follows:

1. Identify all data entry points into the system. (At all such points, data can enter the system that needs to be validated.)
2. For each data entry point, calculate data flow paths through the system. All operating system variables to which the entering data is directly or indirectly assigned must be recorded.
3. Examine all operators referencing a variable identified in (2) above. Verify that the validity condition enforced on each data path leading ot that operator/operand is sufficient.

Step 2 can be automated using data dependency analysis or a modified form of symbolic execution. Steps 1 and 3 must be done manually. It is important to note that without detailed semantic information describing operations being performed, any procedure, such as the above, can only tell an evaluator where to look for errors, but not what to look for.

7.1.1.1.2 Inside-to-Outside Validation Methodology Inside-to-outside validation methodology works in the opposite direction as the previously described outside-to-inside validation methodology. In this case, the purpose is to determine what data will be valid. One starts with the internal operators and passes in the reverse direction of control flow. The methodology ends when the input entry points are reached.

The outside-to-inside algorithm from Bisbey is the following [Bisb1978]:

Suppose a protection evaluator can identify all critical operators in the system and can specify for each operator the validity condition that must hold for the successful completion of that operator. The problem of finding validation errors then amounts to determining the sufficiency of validation code on all paths leading to that operator. A procedure for checking sufficiency would be as follows:

1. Identify the critical operations within the operating system and the necessary conditions associated with those operations. Record the condition with the associated operand.
2. If an operand is a local or a parameter, follow all possible control paths leading from the operation to determine the data paths leading to the critical operation. In passing in a reverse direction through code that enforces portions of the validation condition, discard the enforced condition. Eventually, one of the following will occur:
   a. All conditions are enforced for that control path.
   b. All conditions are not enforced upon reaching a user/system interface, i.e., a validation error can be caused by supplying a value outside the range of remaining unenforced condition.
   c. The control path terminates at a global variable/parameter interface within the system. Go to 3.
3. If the operand is a global or formal parameter from 2c, all operators modifying the global/parameter must contain as an output condition the validity condition associated with the respective variables. They become critical operators to be evaluated by this same algorithm.

A more detailed description of validation errors can be found in [Carl1976].

7.1.1.2 Buffer Overflows

Called the "vulnerability of the decade" by Cowan [Cowa2000], buffer overflows are discussed in detail in Section A.1.15 on page 245. Buffers are the way that operands are entered into

Table 7.1: Buffer Overflow Locations

Attack Code Location	Code Pointer Types
Resident	Activation record
Stack Buffer	Function pointer
Heap Buffer	Longjmp buffer
Static Buffer	Other variables

routines and programs.

Buffer overflows do not concern "critical conditions" per se (Section 7.1.1.1) but instead the input itself. A vulnerability occurs when there is no length check to the input and the input overflows to memory locations outside the original buffer locations. There have been many uses of buffer overflows. The Internet Worm (Section 2.3.1.2 on page 28) used buffer overflows in the *finger* daemon (*fingerd*) [Zimm1991]to overflow the filename input buffer [Cowa2000].

Cowan lists different types of attack code locations and the code pointer types that are in use; they are shown in Table 7.1. For more detailed information, see his paper [Cowa2000] and Dildog's work [Dild1998].

Inputs not necessarily constrained to the input into buffers are discussed in Section 7.1.2.1 below.

7.1.1.3 Summary

In order to insure reliable operation of programs, all critical conditions and inputs must be checked to ensure that no buffers overflow and everything is validated. Improper validation is the most common form of vulnerabilities in computers.[1]

7.1.2 Validation of Protocols

Sometimes a system that is desired to be checked for improper validation does not have code available to run the algorithms described in Section 7.1.1. Often this system is just a protocol, and for this, other methodologies and algorithms are needed. Sections 7.1.2.1 – 7.1.2.4 describe methods to determine if a protocol or system has improper validation.

[1] As the expression goes: "Garbage In, Garbage Out."

7.1.2.1 Inputs

Similar to buffer overflows described in Section 7.1.1.2, this section describes methodologies to check inputs into protocols or systems for improper validation. In general, anything that is input into a system or protocol must be validated to make sure that it is proper data and that the data does not overflow a buffer into which the input is being put. For example, all addresses must be checked to be sure that the address: has a valid structure; has a legitimate and probable source; is cryptographically signed; and does not overflow the input address buffer.

The address structure must be a valid address for where the packet claims it is coming from and where it is going to. For example, addresses are often segregated into local or global addresses. If an address comes to a border router that has the internal test address,[2] the broadcast or multicast address as its source, or other addresses that should not occur, the packet or message is probably in error. This is what ingress [Ferg2000] and egress filtering do. Ingress filtering checks packets coming into a network to verify that the source address is outside the network and the destination address is inside the network. Egress filtering checks packets going out of a network to verify that the source address is inside the network and the destination address is outside the network.

All objects should be cryptographically signed to make sure that they are from who they say they are from. Finally, the object must not overflow the input buffer that is designed to hold them. This overflow not only applies to addresses, but to the entire packet. For example, IPv6 allows a "jumbogram," which is a packet of up to four gigabytes. If a node supports jumbograms, an input packet buffer must have the room to either store that entire packet or ensure that all packets over the buffer length are automatically discarded [Borm1999].

7.1.2.2 Protocol Verifiers

Messages in protocols also need to be verified. Protocols need to make sure that the received messages do not allow a version rollback (see Section A.1.12 on page 244) or allow

[2] IPv4 has addresses that are not to be broadcast outside the local network. In addition, certain addresses are not to be source address; and certain addresses are not to be destination addresses. See Figure 3.9 in [Stev1994], p. 45.

an unverified packet overwrite some data that was previously written by another (verified or unverified) packet. If the packets are not validated as they arrive, the second packet could be spoofed and overwrite the data accepted and stored from the valid source.

Formal protocol verifiers are available for cryptographic protocols. They will not be discussed further in this dissertation, but see Meadows [Mead1996].

7.1.2.3 State Machines

Any system with an internal state machine must follow general state machine design principles in order to be designed correctly. In hardware, state machines are stored internally by a number of bits (flip-flops or other latches). Because of the binary nature of computers, a set of N storage bits can hold 2^N states. If less than 2^N states are used for valid states, the other bit combinations for unused states are designated as "don't care" states. The state machine should not get to those states. If however, it does (by a stray bit flip due to an external source such as a gamma ray, magnetic field, etc.), the correctly designed hardware should get back to a known state. Refer to a digital design hardware book such as Wakerly [Wake1990] and Nelson et al. [Nels1995].

The same is in software. The designer must consider the following:

- Unused states (i.e., don't care states)
- What happens in an unused state
- Designate a way back to a known state if the state machine is put in an "invalid" state.

Even protocols that have been tested for numerous years have problems. TCP itself has been shown to have state transition errors [Guha1995, Guha1996, Guha1997].

7.1.2.4 Past Attacks

Past attacks are a prime source of current attacks. This dissertation has shown in numerous places that present attacks are similar to past attacks. Sometimes the present attacks are the exact same attacks used in the past; at other times, the attack is the same, but in a different medium. For example, a wired attack can be launched against a wireless medium. Or, an attack on one protocol can be used against a similar protocol.

7.1.3 Summary

In summary, *everything* must be verified! Validation is the most critical aspect of determining if a system or program is secure. As shown in Section 8.2, improper validation in the IEEE 802.11 protocol produces the most errors. The next section (Section 7.2 covers the methodologies and algorithms for determining improper exposure in systems.

7.2 Exposure

The Protection Analysis study [Bisb1978] explored in more detail some of the categories: P1 (Consistency of data over time), P2 (Validation of operands), P3 (Residuals), and P6 (Serialization). Exposed Representations (P8) was not explored further, and no algorithm or methodology was developed. VERDICT category Improper Exposure is derived from P8; in this section, a methodology for finding improper exposure is developed.

7.2.1 Exposure Methodology Introduction and Observations

The problem of controlling covert channels (the confinement problem) is explored by Lampson [Lamp1973], Lipner [Lipn1975], and Kemmerer [Kemm1983]. Covert channels, that is incorrect domain crossing, generates improper exposure, but it is the *effect* of other problems. VERDICT outlines four basic errors in security that must be solved, improper validation, exposure, randomness, and deallocation.

Everything, including variables, objects, and actions, should be considered improperly exposed until proven otherwise. That is, the initial exposure domain of all objects is infinite (∞).

The domain is represented by an upper case letter and an object as a lower case letter. If an object is within a domain, it is represented by the symbol "is an element of" ($\in$). If one domain can transfer information to another domain, it is represented with a right arrow ($\rightarrow$).

Another important observation is that exposures can be chained. For example, the object x in domain A can be exposed to B if domain A can transfer information to B. That is, if x

is an element of domain A, and domain A can transfer information to domain B, then x can be seen by domain B, or x is now an element of domain B:

$$\text{if } (x \in A) \text{ and } (A \rightarrow B) \text{ then } (x \in B)$$

The logic can be continued, exposing the object x to domain C if domain B can transfer information to domain C:

$$\text{if } (x \in B) \text{ and } (B \rightarrow C) \text{ then } (x \in C)$$

In practical terms, consider a password entry by a user. When the user types in the password (x), the finger motions (and hence the password) are exposed to anyone within sight (Domain A). That is, ($x \in A$). If a malicious person (Domain B) was able to shoulder surf the user's password ($A \rightarrow B$), then the password (x) is now within the body of knowledge[3] (domain) of that malicious person (Domain B) ($x \in B$).

Extending the analogy, now that the malicious person (Domain B) knows the password (x), anywhere the malicious person goes (Domain C) could potentially know the password (x) if the malicious person (Domain B) transfers ($B \rightarrow C$) that information via voice, writing, etc. The password would be transferred to that Domain C. Exposures can be chained. The password went from being exposed to anyone watching the entry of the password to *anywhere* the shoulder surfing malicious person went and transferred the knowledge to.

7.2.2 Finding Vulnerabilities vs. Exposures

There are differences between finding vulnerabilities and finding exposures. Theo de Raadt and a team of programmers have gone through the BSD system code and corrected errors.[4] They found numerous vulnerabilities and have patched them. Because of the systematic way they went through the code, it is considered by many to be the most secure operating system in use.

[3] i.e., the brain.
[4] http://www.openbsd.org

Table 7.2: Vulnerability Logic

A	B	Vulnerability
0	0	0
0	1	0
1	0	0
1	1	1

Table 7.3: Exposure Logic

A	B	Exposure
0	0	0
0	1	1
1	0	1
1	1	1

7.2.2.1 Vulnerabilities

Each vulnerability may be composed of multiple faults or errors. The vulnerability as a whole is stopped when at least one fault is not operational. Consider a fault as an electric switch. If the switch is closed (logic 1), the fault exists; if the switch is open (logic 0), the fault does not exist. To determine if a vulnerability composed of multiple faults exists, one needs to look at the combination of the individual faults in the vulnerability. To eliminate ("open") the vulnerability "circuit," one must "open" one fault *or* another. In essence, the errors or faults are in series, and the vulnerability is the equivalent of a logic "AND." See Table 7.2 for an example of a vulnerability composed of two faults, A and B.

7.2.2.2 Exposures

Exposures are similar to vulnerabilities. As each vulnerability may have multiple faults or errors, so each exposure may effect multiple domains. As the errors of a vulnerability are in series, exposed domains are in parallel. Instead of stopping a vulnerability by eliminating (opening) one of the faults, all exposed domains must be opened. In essence, the circuit representation of exposures is an "OR" circuit. See Table 7.3 for an example of an exposure composed of two domains, A and B.

7.2.3 Exposure Methodologies

In this section, two methodologies are shown that can be used to determine if an improper exposure occurs. The first is called in-to-out, and the second is called an "exposure matrix."

7.2.3.1 In-to-Out

This methodology is similar to the in-to-out methodology for validation as discussed in Section 7.1.1.1.2. In order to assess exposure risk, all objects must be examined. The individual exposures must be coupled with a logic OR between them to determine the composite improper exposure. Once all the exposed domains are known, it can be determined if any of these domains are improperly exposed.

7.2.3.2 Exposure Matrix

Another way to determine exposure of objects to domains is to create a matrix with all the objects crossed with all the domains. One takes a listing of all objects that are elements of domains (e.g., $x \in A$), a listing of all objects that can transfer information from themselves to another object, and a listing of all domains that can transfer information from themselves to another domain. Because some objects in a domain may not be able to transfer information to another object in the same domain, it cannot be assumed that domains are reflexive ($A \rightarrow A$). With these pieces of information, an object domain exposure resolution is made.

An example of an object domain exposure matrix is given in Table 7.4. In an object domain exposure matrix, objects are listed on one axis, while domains are listed on the other. Cells within the matrix indicate whether a particular object is an element of a particular domain. In this example, it is shown that objects one and two are elements of domain A ($1 \in A$, $2 \in A$), and object two is an element of domain C ($2 \in C$). Object two can be an object of both domains A and C if it crosses boundaries between the two domains.

If it is known that object one can transfer information to object two ($1 \rightarrow 2$), then it can be shown that object one is exposed to domain C ($1 \in C$) because object 2 is an element of C. If object 1 is exposed to domain C, then domain A can potentially transfer information to domain C ($A \rightarrow C$).

Table 7.4: Object Domain Exposure Matrix

		Domains		
		A	**B**	**C**
Objects	**1**	∈		
	2	∈		∈
	3			

Table 7.5: Domain Exposure Matrix

		Domains		
		A	**B**	**C**
Domains	**1**	∈		
	2	∈		∈
	3			

Instead of objects being considered separate from domains, objects can be considered domains in and of themselves. This is similar to object oriented programming methodologies. When this consideration is made, the exposure matrix becomes a "Domain Exposure Matrix" with each axis of the matrix being a listing of all the domains (Domains $\times$ Domains). A listing is made of all domains that can transfer information to another domain (Domain $\rightarrow$ Domain), and a domain exposure resolution is made, as in Table 7.5.

A domain exposure resolution determines which domains can transfer information to other domains given a listing of initial domains that can transfer information. I have developed an algorithm that will resolve domains:

Given:

$$\begin{aligned} x(1) &\rightarrow y(1) \\ x(2) &\rightarrow y(2) \\ &\vdots \\ x(n) &\rightarrow y(n) \end{aligned}$$

The algorithm is the following:

for $j = 1$ to n

for $k = 1$ to n

if $y(j) = x(k)$ then $x(j) \rightarrow y(k)$ if d.n.e.

until no more domains are added

An example is given below. If the following domain transfers are given:

$$A \rightarrow B \tag{7.1}$$

$$B \rightarrow C \tag{7.2}$$

$$C \rightarrow D \tag{7.3}$$

After the first pass through the algorithm, the following equations are derived using the previous three equations (7.1 – 7.3):

$$(7.1 + 7.2) \qquad A \rightarrow C \tag{7.4}$$

$$(7.2 + 7.3) \qquad B \rightarrow D \tag{7.5}$$

After the second pass through the algorithm, the final equation is derived:

$$(7.1 + 7.5) \qquad A \rightarrow D \tag{7.6}$$

Thus, it is shown in Equation 7.6 that Domain A can transfer information (is exposed) to Domain D.

7.2.4 Summary

In this section, the finding of vulnerabilities is contrasted with the finding of exposures, and two methodologies are presented to find improper exposures. In the next section, methodologies for the third VERDICT category, improper randomness, are explored.

7.3 Randomness

Random numbers are used in many aspects of computer security. In cryptography, session keys and nonces are derived from random numbers [Schn1996, Kels1999]. Other uses include passwords, initialization vectors, the salt in UNIX password storage [Curr1992], initial sequence numbers in TCP [Stev1994, Wrig1995], and even in applications such as NFS file handles [Vene1996]. Some articles and reports on random numbers include [East1994], [Giff1988], [Kels1999], [Park1988], and [vonN1961]. Classic works include those by Schneier [Schn1996] and Knuth [Knut1981].

7.3.1 Types of Random Numbers

Gifford [Giff1988] outlines three types of random numbers, or random bits: perfect, natural, and pseudo-random. He classifies the three types by the generating process.

7.3.1.1 Perfect Random Bits

Perfect random bits, says Gifford, "...are generated by an unbiased Bernoulli process where trials are completely independent [Drake67].[5] Perfect random bits represent a theoretical ideal that we would like a random bit generator to achieve in practice" [Giff1988]. No further discussion will be made on perfect random bits in this document.

7.3.1.2 Natural Random Bits

Gifford describes natural random bits as those, "...generated by transducing a natural random process such as shot noise, radioactive decay, or coin flips. Natural random bits are required by pseudo-random bit generators to serve as seeds. Thus a source of natural random bits is an essential part of any random bit generator" [Giff1988].

7.3.1.3 Pseudo-random Bits

The final type of random bits is the one that is the most often used. They are computed by an algorithmic process with a "seed." The algorithm is usually a linear congruential generator, which is deterministic, of the following form:

$$X_{n-1} = (aX_n + c) \bmod m \tag{7.7}$$

7.3.2 Randomness Methodology

While pseudo-random numbers are the most widely used in security, they are deterministic, derived from the linear congruential generator and a "random seed." The seed needs to be properly randomized. For adequate security, seeds should be derived from natural random numbers. Eastlake et al. suggests in RFC 1750 that hardware random number generators

[5] [Drake67] Drake, A., *Fundamentals of Applied Probability Theory*, McGraw Hill, New York, 1967.

should be included on devices [East1994]. With the cost and size of hardware shrinking daily, these devices are not out of the question. The assumptions that one cannot have a hardware random number generator as part of the entire system is changing (see Section 2.2 about changing assumptions in computing).

Gifford suggests that noise diodes can be used as a base for a natural random seed. His 1988 technical report illustrates the validated possibility of such devices [Giff1988]. While perfect random numbers may not be possible to achieve, but natural random seeds come statistically close.

7.3.3 Summary

This section has outlined the wide use of random numbers, described the three types of random numbers (perfect, natural, and pseudo-random), and outlined a methodology for achieving "enough" randomness. Recommendations for not having improper randomness include having a hardware source of natural random numbers on each computational device. The next section (Section 7.4) in this chapter will overview the methodologies for ensuring proper deallocation, or residuals.

7.4 Deallocation

Deallocation, or the residuals that result from the deallocation, comprises more than the deletion of data in a computer. From the procedures of physical document destruction, to dumpster diving, or old backup tapes, the process of deallocation can be inside a computer system, or outside. As described in Section 4.2.2.3, there are three types of residuals: access, composition, and data. These three types along with methodologies to control improper deallocation will be discussed in Sections 7.4.1 – 7.4.3.

7.4.1 Data Residuals Methodology

Data residuals, the most common, is the most familiar to users. Residuals caused by improper deallocation result from old content in cells. Bisbey [Bisb1978] outlines an algorithm to

determine how to find and eliminate improper data residuals from occurring; Hollingworth [Holl1976] expounds on the methodology further. The algorithm presented by Bisbey for, "...finding data residuals is based on identifying the cell allocation/deallocation routine in which residual prevention code should be contained" [Bisb1978]. It is the following:

1. Identify all cell types found in the system. This can be done manually listing various storage media and cells on that media and by examining system data declarations.
2. For each cell, identify its particular freepool, i.e., the buffers for cell resources between deallocation and allocation.
3. For each freepool, identify allocation/deallocation code by finding all symbolic references to the freepool.
4. For each allocation/deallocation routine, determine if a data residual can occur.

This algorithm is straightforward, and its definition is somewhat circular. Although it lists step-by-step how to find each allocation/deallocation routine in which a data residual might occur (i.e., cell types to cell to freepool to allocation/deallocation routine), in step four of the methodology to determine if a data (content) residual occurs, Bisbey's methodology says, "For each allocation/deallocation routine, *determine if a data residual can occur*" (Emphasis added) [Bisb1978]. As of this report, the only way to determine this is by manual inspection. As discussed in Section 9.4 on page 236, automated procedures to check all allocation and deallocation routines could be designed and implemented.

7.4.2 Composition Residuals Methodology

Composition residuals yield knowledge about how deallocated cells relate to each other, often in terms of size or content. Hollingworth [Holl1976] comments on this methodology and strategy for eliminating residuals.

Important attributes to consider include cell size, inter-cell relationships, and intra-cell relationships. The location in the free-pool may provide unwanted exposure. For example,

if a stack frame of known architecture is deallocated, the return address or other parameters passed onto it may be known by knowing the offsets. Hollingworth suggests that buffers allocated of 'N' bytes may reveal that a password intended for that buffer is only 'N' bytes long.

Deallocation routines also must not preserve cell size; or along with the allocation routine, order of insertion. This is important according to Hollingworth, to prevent someone from knowing that a group of cells was for a particular type of control block, or other operating system data structure [Holl1976].

7.4.3 Access Residuals Methodology

Access residuals are the result from improper deallocation of pointers, yielding dangling references. They are similar to data residuals (Section 7.4.1), and each access residual may have multiple references. The code to remove the access pointers may not always be in a single place such as the allocation or deallocation code; for example, it may be in code that copies objects [Holl1976].

As with data residuals, one must identify and manually search each of the following for improper residuals: allocation and deallocation code; creation and destruction of access paths; tables containing offsets and pointers; and interrupted translation routines for allocation and deallocation. As discussed in 9.4, automated procedures to check all of the above potential error points could be designed and implemented.

7.4.4 Summary

There are three types of residuals: access, composition, and data. These types of residuals were discussed, and methodologies for preventing residuals in each were described. However, any object in general (including data), when it is attempted to be destroyed (deallocated), must ensure that the destruction is complete. Follow the advise of Schneier, presented in Section 5.3.4 on page 159 for how drastic a measure one may have to do in order to ensure proper deallocation.

7.5 Summary

This chapter has shown methodologies and algorithms of VERDICT. Algorithms and methods were developed to apply VERDICT to any protocol or system. By using these methods and algorithms, one can determine if there are security violations in the system under test. Some of the methods presented in this chapter have been written into actual code in the 1970s [Bisb1978]. Although that code is not available, one can implement similar programs today. Chapter 8 will apply VERDICT to the IEEE 802.11 wireless protocol to show numerous potential vulnerabilities. These methodologies present a holistic approach to the problem of finding and predicting errors in systems.

Chapter 8

Application of VERDICT to IEEE 802.11

This chapter shows the application of VERDICT to the IEEE 802.11 protocol, as proposed in the standard outlined by O'Hara and Petrick [O'Ha1999]. O'Hara and Petrick were closely involved with the standard,[1] and the book of 173 pages is a summary of the several-hundred pages standard from the IEEE.[2]

Security of wireless networks will be a fruitful area of research in the future; Snow et al. describe weaknesses in wireless infrastructures and the need for reliability and survivability [Snow2000]. By applying VERDICT to IEEE 802.11, it is shown that IEEE 802.11 as it presently stands has serious security flaws with improper validation, exposure, and randomness.

8.1 Summary of IEEE 802.11

This section will summarize the IEEE 802.11 Physical (PHY) and Medium Access Control (MAC) layers in Section 8.1.1 and the Wired Equivalent Protocol (WEP) in Section 8.1.2.

[1] "Mr. O'Hara has been involved with the development of the IEEE 802.11 WLAN standard since 1992. He is the technical editor of that standard and chairman of the revisions and regulatory extensions task group.... Mr. Petrick serves as Vice-Chair of the IEEE 802.11 WLAN standards committee" [O'Ha1999].

[2] As of 28 November 2000, the standards of 802.11-1999, 802.11a-1999, and the 802.11b-1999 were 720 pages and were $288.00 (IEEE Member $230.00). Found on the world wide web at: *http://standards.ieee.org/catalog/IEEE802.11.html*

Following an overview in this section of the protocol, the next section (Section 8.2) will overview security vulnerabilities in IEEE 802.11 based on the application of VERDICT.

8.1.1 IEEE 802.11 — PHY and MAC

This section describes the physical (PHY) and medium access control (MAC) layers of the IEEE 802.11 protocol. This section is primarily based on O'Hara and Petrick [O'Ha1999], 1996 tutorials on IEEE 802.11 found on the WWW,[3] and an non-reviewed paper written by Lough et al. in 1997 [Loug1997]. IEEE 802.11 is a wireless protocol operating at the Physical (PHY) and Medium Access Control (MAC)[4] layers. Only a brief introduction to the actual protocol, including knowledge necessary to comprehend the security vulnerabilities, will be given.

In IEEE 802.11, the proposed standard for wireless LANs, there are three different ways to configure a network: ad-hoc (IBSS), infrastructure (ESS), and a mixture of both (BSS). In the ad-hoc network, computers (or "stations") are brought together to form a network "on the fly." As shown in Figure 8.1, there is no structure to the network; there are no fixed points; and usually every node is able to communicate with every other node. This is a useful configuration for a meeting where everyone has a laptop, or other similar events. In 802.11 parlance, this is known as an *Independent Basic Service Set* (IBSS) [Loug1997, O'Ha1999].

An Access Point (AP) is a fixed station or node that allows other stations in the IBSS to communicate with each other or to access further networks through it. If an IBSS described

[3]Tutorial presentations' references are the following:

- Document IEEE P802.11-96/49A Rev.1, *Tutorial on 802.11 to 802*, Vic Hayes, Lucent Technologies, Chair IEEE P802.11;
- Document IEEE P802.11-96/49B, *802.11 Architecture*, Greg Ennis, Symbol Technologies;
- Document IEEE P802.11-96/49C, *802.11 Tutorial: 802.11 MAC Entity: MAC Basic Access Mechanism: Privacy and Access Control*, Wim Diepstraten, Lucent Technologies;
- Document IEEE P802.11-96/49D, *Frequency Hopping Spread Spectrum PHY of the 802.11 Wireless LAN Standard*, Nafali Chayat, BreezeCom; and
- Document IEEE P802.11-96/49E, *Direct Sequence Spread Spectrum Physical Layer Specification: IEEE 802.11*, Jan Boer, Chair DS PHY, Lucent Technologies WCND Utrecht (sic)

All documents found on as of November 2000 at: *http://grouper.ieee.org*.

[4]Not to be confused with the other Three Letter Acronym (TLA) Mandatory Access Control (MAC) discussed in reference to Loscocco in 2.2.3.

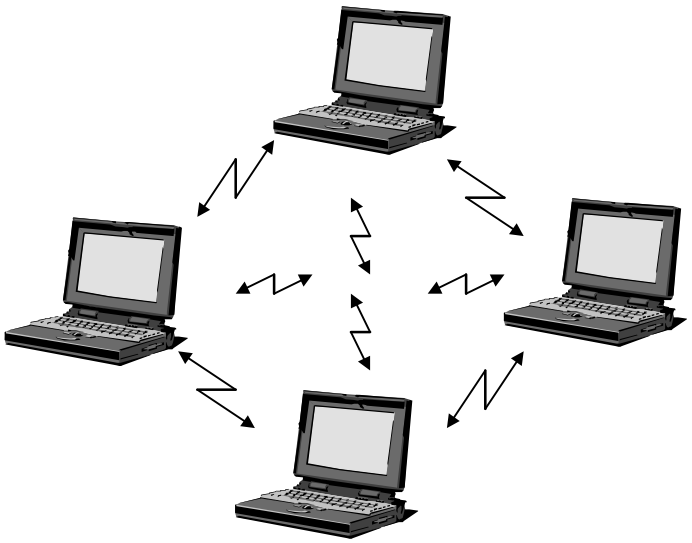

Figure 8.1: IEEE 802.11 Independent Basic Service Set (IBSS)

above has an AP as one of the nodes, it is no longer "independent" and is labeled a *Basic Service Set* (BSS). Every communication must go through an AP, even though it seems inefficient to do so. The reason for this design is so the AP can buffer packets and send them in a burst to a station that is operating in low power mode [Loug1997, O'Ha1999].

As shown in Figure 8.2, the third type of network structure used in wireless LANs is the infrastructure. This architecture uses fixed network access points with which mobile nodes can communicate. These network access points are connected to land lines to widen the LAN's capability by bridging wireless nodes to other wired nodes. If service areas overlap, handoffs can occur. This structure is very similar to the present day cellular networks around the world.

8.1.1.1 IEEE 802.11 Physical Layer (PHY)

The IEEE 802.11 standard places specifications on the parameters of both the physical (PHY) and medium access control (MAC) layers of the network. The PHY layer, which actually handles the transmission of data between nodes, can use either direct sequence spread spectrum (DSSS) , frequency-hopping spread spectrum (FHSS), or infrared (IR) pulse position modulation. IEEE 802.11 makes provisions for data rates of either 1 or 2 Mbps,[5] and

[5]Mega-Bits-Per-Second; millions of bits per second.

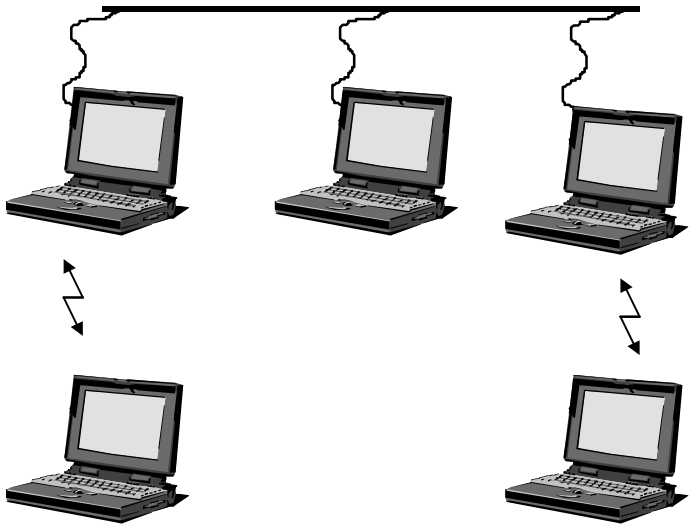

Figure 8.2: IEEE 802.11 Extended Service Set (ESS)

calls for operation in the 2.4 - 2.4835 GHz frequency band (in the case of spread-spectrum transmission), which is an unlicensed band for industrial, scientific, and medical (ISM) applications, and 300 - 428,000 GHz for IR transmission. Infrared is generally considered to be more secure to eavesdropping, because IR transmissions require absolute line-of-sight links (no transmission is possible outside any simply connected space or around corners), as opposed to radio frequency transmissions, which can penetrate walls and be intercepted by third parties unknowingly. However, infrared transmissions can be adversely affected by sunlight,[6] and the spread-spectrum protocol of 802.11 does provide some rudimentary security for typical data transfers.

There are extensions to the original IEEE 802.11 specification, IEEE 802.11a and IEEE 802.11b, that allow transmission up to 54 Mbps. It accomplishes this through RF[7] or infrared transmission and uses Orthogonal Frequency Division Multiplexing (OFDM). Orthogonal Frequency Division Multiplexing (OFDM) is, "...a means of providing power efficient signaling for a large number of users on the same channel. Each frequency... is modulated with binary data (on/off) to provide a number of parallel carriers each containing a portion of user data" [Rapp1996].

[6]Private communication with Dr. Theodore S. Rappaport, June 1997.
[7]Radio Frequency.

8.1.1.2 IEEE 802.11 Medium Access Control Layer (MAC)

The MAC layer, as the name implies, controls access to the medium, as there may be numerous conflicts to access the transmission channel. IEEE 802.11 operates as a CSMA/CA[8] MAC.

8.1.1.2.1 Backoff Factor In this protocol, when a node receives a packet to be transmitted, it first listens to ensure no other node is transmitting. If the channel is clear, it then transmits the packet. Otherwise, it chooses a random "backoff factor" which determines the amount of time the node must wait until it is allowed to transmit its packet. During periods in which the channel is clear, the transmitting node decrements its backoff counter. (When the channel is busy it does not decrement its backoff counter.) When the backoff counter reaches zero, the node transmits the packet. Since the probability that two nodes will choose the same backoff factor is small, collisions between packets are minimized. This is known as the "binary exponential backoff algorithm" [O'Ha1999].

8.1.1.2.2 RTS, CTS, and the Hidden Node Problem Collision detection, as is employed in Ethernet and IEEE 802.3, cannot be used for the radio frequency transmissions of IEEE 802.11 because when a node is transmitting it cannot hear any other node in the system that may be transmitting, since its own signal will drown out any others arriving at the node. Whenever a packet is to be transmitted, the transmitting node first sends out a short ready-to-send (RTS) packet containing information on the length of the packet. If the receiving node hears the RTS, it responds with a short clear-to-send (CTS) packet. After this exchange, the transmitting node sends its packet. When the packet is received successfully, as determined by a cyclic redundancy check (CRC), the receiving node transmits an acknowledgment (ACK) packet. This back-and-forth exchange is necessary to avoid the "hidden node" problem, illustrated in Figure 8.3. As shown, node A can communicate with node B, and node B can communicate with node C. However, node A cannot communicate node C. Thus, although node A may sense the channel to be clear, node C may in fact be

[8]Carrier Sense, Multiple Access / Collision Avoidance; as opposed to the CSMA/CD (Carrier Sense, Multiple Access / Collision Detection) of Ethernet and IEEE 802.3. See [Hals1996] or other Data Communications literature for further information.

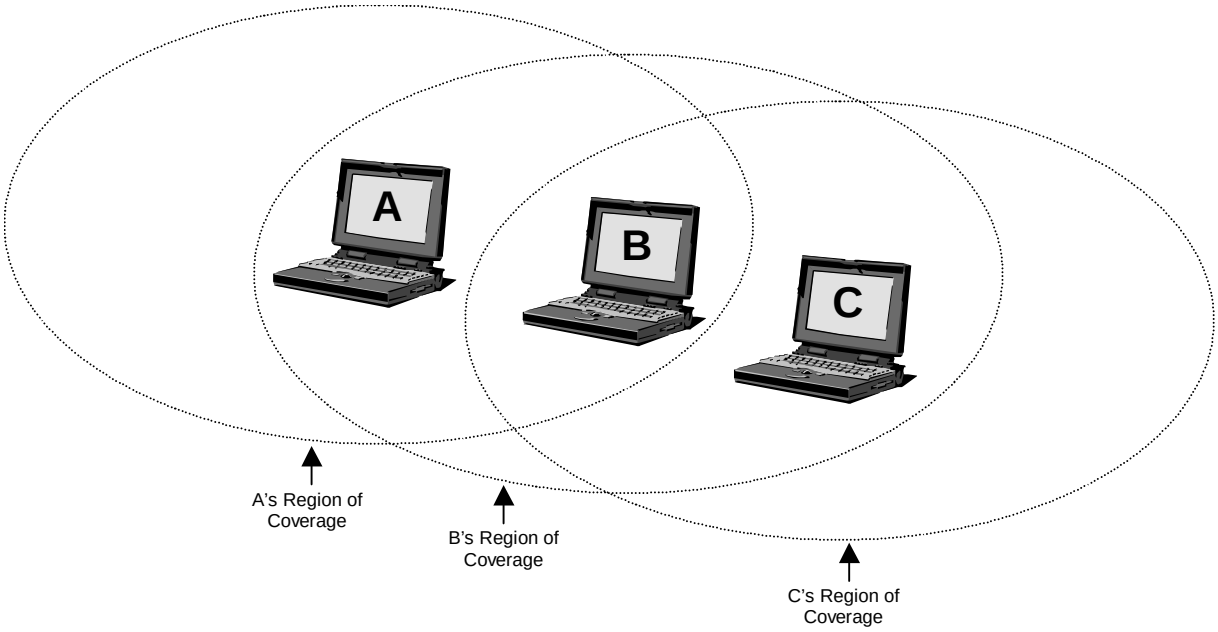

Figure 8.3: IEEE 802.11 Hidden Node Problem

transmitting to node B. The protocol described above alerts node A that node B is busy, and hence it must wait before transmitting its packet.

8.1.1.2.3 DCF and PCF The MAC layer can be operated with a Distributed Control Function (DCF) and a Point Control Function (PCF). The DCF allows any station to transmit after ensuring the medium is clear. The PCF works over the DCF and allows a Point Coordinator (PC — usually located in the Access Point (AP)) to poll each station and use a time division multiplexing in a round robin fashion to allow stations to transmit. Further details can be found in [O'Ha1999].[9]

8.1.1.2.4 Beacon and Probe Frames In order for mobile stations to determine if an Access Point (AP) exists, the station can listen for "beacon" frames; this is known as "passive scanning."[10] Beacon frames can contain timestamps, beacon intervals, supported rates, and other parameters of the BSS. Stations can also use beacons in order to save power. When stations are conserving power in their "sleep" mode, they wake up at set times and listen for beacon frames. The stations then receive the "timing" or "synchronization" of the BSS; if

[9] [O'Ha1999], pp. 27–31.
[10] IEEE P802.11-96/49C.

frames were buffered at the AP, the station can receive the data in a burst of frames. Probe frames are used to do "active scanning." As opposed to passive scanning, active scanning is initiated when a station wants to initiate the search for an AP. A probe frame is sent out, and if an AP exists and can accept traffic, a probe response is returned.[11]

8.1.1.2.5 Association and Authentication When a station wishes to join an Basic Service Set (BSS), it first has to *authenticate* itself to the BSS by a challenge-response protocol. After authentication, the station then *associates* with the BSS. This will let the station know what transmission rate(s) are available and other parameters of the BSS. When a station wants to leave a BSS, it *disassociates* from the BSS [O'Ha1999]. There are three states of a station:

1. Unauthenticated and Unassociated;
2. Authenticated and Unassociated; and
3. Authenticated and Associated.

At each stage of this "state diagram," (see Figure 8.4 there are only certain types of frames that can be transmitted.[12] For further information, refer to O'Hara and Petrick's book [O'Ha1999].

8.1.2 Wired Equivalent Privacy (WEP)

Since any transmission through a wireless media can be intercepted, the designers of IEEE 802.11 wanted a basic cryptographic protocol that would be the equivalent of a wired network. It is supposed to protect data frames against view, but not traffic analysis.

However, in a draft release of a paper, Borisov et al. find serious flaws in the WEP protocol because of the "...misapplication of cryptographic primitives" [Bori2001]. These flaws include potential keystream reuse, use of decryption dictionaries, key management, message authentication (including message modification, message injection, and message decryption), and reaction attacks.

[11] IEEE P802.11-96/49C.
[12] [O'Ha1999], pp. 15–18.

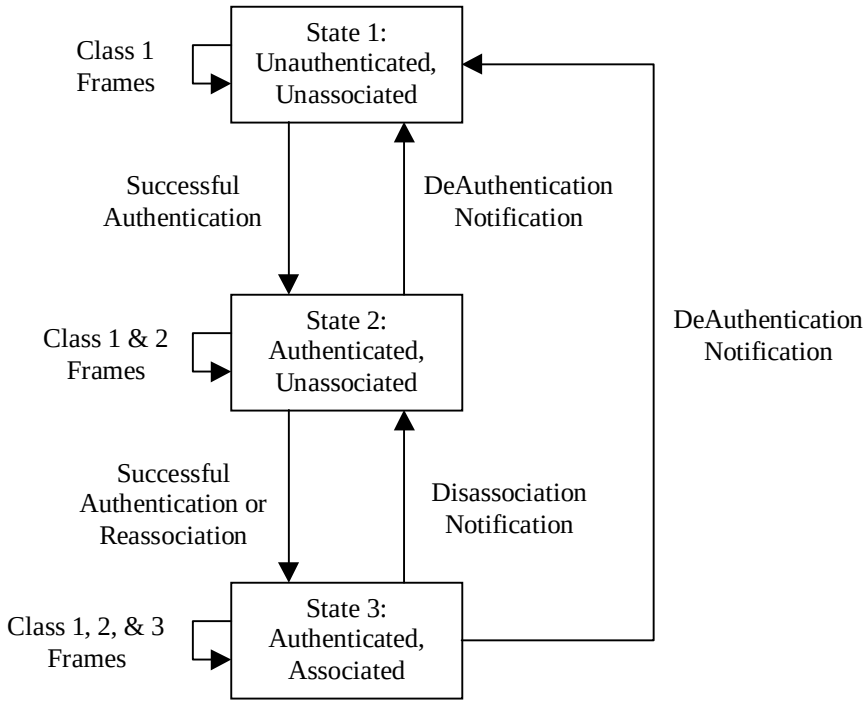

Figure 8.4: IEEE 802.11 State Diagram

Orinoco Wireless, makers of IEEE 802.11 products, released a whitepaper countering the arguments made in [Bori2001].[13] They counter that WEP is a deterrent against the majority of attacks, and the specific attacks they mention would be difficult and costly to mount. In addition, they note that changes to the standards are being made to make the standard more secure. While the standard may improve, security should never rest upon the supposed inability of an attacker to launch an attack.

The data is encrypted with RC4, which is currently a proprietary algorithm owned by RSA Data Security. Although RC4 can use up to a 256 bit key, the current WEP standard only uses 40 bits. This is due to the United States restriction on exporting certain cryptography.[14] Forty bits is about one trillion keys:

$$2^{40} = 1.0995 \times 10^{12} \text{ keys}$$

If a hardware chip could test one million keys per second,[15] a brute-force attack could be

[13] As of February 2001, it can be found at ftp.orinocowireless.com/pub/docs/ORINOco/ARTICLE/WiFiWEPSecurity.pdf.

[14] Certain cryptography is classified as a "munition." [Schn1996], pp. 610–617.

[15] This is quite doable; the DES (Data Encryption Standard) cracker from EFF searches sixty-four million

completed in 12.7 days:

$$\left(1.0995 \times 10^{12}\,\text{keys}\right)\left(\frac{1\,\text{sec}}{10^{6}\,\text{keys}}\right)\left(\frac{1\,\text{day}}{86400\,\text{sec}}\right) = 12.7\,\text{days}$$

If one were to put 1000 chips in parallel, the time to crack 40 bits would be *20 minutes*! Schneier theorizes that if one could encrypt a 64-bit (8 bytes) block of selected text and encrypt it with all 2^{40} (1.0995×10^{12}) keys, it would take:

$$\left(\frac{8\,\text{bytes}}{\text{key}}\right)\left(1.0995 \times 10^{12}\,\text{keys}\right) = 8.7961 \times 10^{12} = 8\,\text{terabytes}$$

This is not at all out of the reach of today's storage capacities; it is large, but storage capacities are increasing tremendously. Once the key is found, all the other communications can be cracked with the same key.

Another weakness in the current IEEE 802.11 WEP protocol is that the key negotiation and distribution are still open debate:

> The IEEE 802.11 standard describes the use of the RC4 algorithm and the key in WEP. However, key distribution or key negotiation is not mentioned in the standard. This leaves much of the most difficult part of secure communications to the individual manufactures of IEEE 802.11 equipment. In a secure communication system using a symmetric algorithm, such as RC4, it is imperative that the keys used by the algorithm be protected, that they remain secret. If a key is compromised, all frames encrypted with that key are also compromised. Thus, while it is likely that equipment from many manufactures will be able to interoperate and exchange encrypted frames, it is unlikely that a single mechanism will be available that will securely place the keys in the individuals stations. There is currently discussion in the IEEE 802.11 working group to address this lack of standardization [O'Ha1999].[16].

For more information about WEP keys, see [Bori2001]. The cryptographic details of all these attacks described and referenced above are not discussed further in this dissertation.

keys per second per chip; there are 64 chips per board, 12 boards per chassis, and two chassis per cracker. This yields 92,160,000,000 keys per second with an average search time of 4.524 days [EFF1998]. Note that the maximum search time, that is the time required to brute-force the entire keyspace, takes twice as long as the average search time. This is because on average, the match is found halfway through the search. In the case of the EFF DES cracker, the entire 56-bit keyspace was searched.

[16][O'Ha1999], p. 77

8.2 Security Vulnerabilities in IEEE 802.11

This section presents an application of VERDICT to IEEE 802.11. After reading [O'Ha1999], the vulnerabilities outlined in Sections 8.2.1 – 8.2.4 were theorized. Some vulnerabilities may prove not to exist; however, the potential for these flaws is noted. References to O'Hara and Petrick [O'Ha1999] cite page numbers that that part of the protocol is discussed. Of all the citations in the validation section (Section 8.2.1), O'Hara and Petrick do not discuss any security problems with the protocol; they just outline the protocol itself. All security flaws are theorized by this author.

8.2.1 Application of VERDICT: Improper Validation

There are a number of vulnerabilities resulting from potential improper validation. These are described in the following sections: 8.2.1.1 – 8.2.1.7. Although there is no guarantee that all improper validation security flaws will be found, methodologies presented in Section 7.1.2 are used.

8.2.1.1 802.11 Validation: MAC Address Validation?

Applying the methodology discussed in Section 7.1.2.1 on page 202, the MAC address of IEEE 802.11 needs to be adequately validated. IEEE 802.11 has as a 48-bit MAC address in the same format as an IEEE 802.3 address and similar to an Ethernet address.[17] The IEEE 802.11 address can indicate if the address assignment is global or individual. Global addresses are administered by IEEE in the same way current Ethernet addresses are with unique manufacturers' identification included as part of the address; with a centralized assignment database, no two addresses should be equivalent. Individual address could be non-unique.[18]

While the 802.11 address is similar to the Ethernet's in format, it is also similar in security. Since, there is no validation of addresses, one can spoof addresses. This is similar to a previous attack of IP spoofing, as described by the methodology in Section 7.1.2.4 on

[17]There is a slight difference between IEEE 802.3 [Post1988] and Ethernet [Horn1984] addresses. See the Postel and Hornig documents and pp. 21–23 in [Stev1994] for an illustrated difference in the frame encapsulation.

[18][O'Ha1999], pp. 40–41.

page 203. This is the first example of improper validation in the IEEE 802.11 standard.

8.2.1.2 802.11 Validation: Invalid State?

As described in 8.1.1.2.5, there are three states in the general IEEE 802.11 state machine that determines what relationships the station has with other stations:

1. Unauthenticated and Unassociated;
2. Authenticated and Unassociated; and
3. Authenticated and Associated.

Using the methodology described in Section 7.1.2.3 on page 203, the IEEE 802.11 state machine is examined. Three states can be represented with two bits in a hardware design. Designers must make sure that the fourth state (Unauthenticated and Associated) must never be reached. If it is reached by some unforeseen circumstance, there must be a way in the state machine to transition to one of the three valid states. Without this, the station could become unstable.

This seems to be a protocol flaw, as the designers of the protocol should make sure that all possible states of the machine are accounted for. They should have put a transition edge to point to what state the machine should "reset" itself to if a major error like the fourth state (Unauthenticated and Associated) were to occur. Since the protocol designers did not show this, the implementations should make sure that a random bit change in the state machine will not cause the implementation to go into a mode out of which it could not change.

8.2.1.3 802.11 Validation: Forced Deauthentication/Disassociation?

Using the methodologies shown in Section 7.1.2.1 on page 202, protocol messages of IEEE 802.11 are investigated. As shown in Figure 8.4, when a deauthentication or disassociation message is received at a station, that station moves to a previous state. There is improper validation in this operation; for if a deauthentication message could be coupled with a spoofed message (Section 8.2.1.1), stations could have their connections maliciously cut. Perhaps it

could also be used as part of an effort to hijack a session. Machines that do adhere to the correct protocol can be used to bump off other innocent users. Observe from [O'Ha1999]:[19]

> A station must react to frames it receives in each of the states, even those that are disallowed for a particular state. A station will send deauthentication notification to any station with which it is not authenticated if it receives frames that are not allowed in state 1. A station will send a disassociation notification to any station with which it is authenticated, but not associated, if it receives frames not allowed in state 2. These notifications will force the station that sent the disallowed frames to make a transition to the proper state in the state diagram and allow it to proceed properly toward state 3.

If a spoofed packet were to be sent claiming to be from another station that was "authenticated" saying to deauthenticate, the receiving station would automatically deauthenticate itself. "The [deauthentication/disassociation] frame includes only a single fixed field, the reason code" [O'Ha1999].[20] However, the code '1' stands for "Unspecified reason," yielding no useful information to the deauthenticating/disassociating station [O'Ha1999]![21]

8.2.1.4 802.11 Validation: RTS Flood?

Observing that the RTS/CTS combination is similar to the TCP's Synchronize (SYN) and Acknowledge (ACK) [Post1981b], we use the methodologies of Section 7.1.2.1 and Section 7.1.2.4 on pages 202 and 203 to investigate this protocol. As stated in Section 8.1.1.2.2, the Request to Send (RTS) frame is followed by a Clear to Send (CTS) frame to ensure that no hidden node can transmit when another node out of the sender's range is also transmitting. This allows other nodes in the broadcast area to suspend transmission until the current frame has been transmitted.[22]

In a SYN-Flood, a SYN packet is sent to the target, and the target returns an ACK packet. This causes a half-open connection to be established at the target. Instead of

[19] [O'Ha1999], p. 17.
[20] [O'Ha1999], pp. 57–58.
[21] [O'Ha1999], p. 65
[22] [O'Ha1999], pp. 21–24.

returning an ACK connection to the target, the attacker sends yet another SYN packet. This causes yet another half-open connection to be established at the target with an ACK sent back. Even though there are timeouts that do not allow a connection to stay half-open forever, there are only a finite set of open connections that can be maintained; hence, a Denial of Service (DoS) occurs.

Similarly, timeouts exist in IEEE 802.11 to prevent one machine from monopolizing the transmission medium. However, many RTS frames could be sent in a flood, thus tying up the medium and causing a DoS. Because a lack of (improper) validation of the senders of the packets, a RTS Flood could be developed.

8.2.1.5 802.11 Validation: Fragmentation Attacks?

IEEE 802.11 allows packets at the network layer and above to be fragmented over the wireless medium.[23] Because of this, a method of fragmenting the packets and putting them back together in the original order is needed. IPv4 uses fragmentation;[Post1981a] IPv6 allows it, but only in a special fragmentation header [Deer1998]. Since IEEE 802.11 uses a similar idea in its fragmentation protocol, the methodology from Section 7.1.2.4 is used to investigate this part of the IEEE 802.11 protocol.

In IPv4, if the fragmentation packets are spoofed and labeled incorrectly, a receiving machine's IP stack may crash trying to reconstruct the packet. Overlapping fragments cause different operating systems to potentially handle the packets in different ways [Ptac1998]. Bellovin describes potential fragmentation attacks for IPSEC in [Bell1996b]. Improper validation of fragmented frames could potentially cause similar problems with IEEE 802.11.

8.2.1.6 802.11 Validation: Retry Overwrites (Hijacked Sessions)?

As discussed in Section 7.1.2.2, IEEE 802.11 is investigated for improper protocol transactions. In the frame control field of each frame, a single-bit retry subfield that is described by O'Hara and Petrick as follows:[O'Ha1999][24]

> It is used to indicate whether a data or management frame is being transmitted

[23] [O'Ha1999], pp. 38, 72–72.
[24] [O'Ha1999], pp. 32, 38.

for the first time or if it is a retransmission. When this subfield is zero, the frame is being sent for the first time. When this subfield is one, the frame is a retransmission. The receiving MAC, to enable it to filter out duplicate received frames, uses this subfield, along with the sequence number subfield.

Using also the techniques in Section 7.1.2.1, the protocol messages are investigated. If the sequence number of the packets can be predicted, frames could be spoofed as is described (along with a proposed solution) in [Bell1996a]. The sequence control field contains a four bit fragment number and a twelve bit sequence number. With only 4096 sequence numbers, and with the numbers incrementing one at a time, the sequence number may be able to be predicted with some repeated frames. If frames could be spoofed, the rogue station could overwrite information at the receiving station [O'Ha1999].[25]

8.2.1.7 802.11 Validation: Authentication in 802.11

Using similar methodologies described in Sections 7.1.2.1 and 7.1.2.4, protocol messages are investigated. A sixteen-bit field in the management frame determines what type of cryptographic authentication is to be used. If the sixteen bit number is '0,' it is an "open system" with *no* validation! When the sixteen bit number is '1,' the system uses a "shared key" system. Numbers '2'–'65535' are undefined [O'Ha1999].[26]

The "shared key" system uses a key that both systems know. No attempt at key management is given, although IEEE 802.11 is working on a standard. For more information on problems with the "shared key" system, see [Bori2001]. A challenge-response system allows the machines to encrypt and decrypt using the WEP system (see Section 8.1.2 on page 221. The challenge text is only authenticated one way, from the mobile station to the access point. This yields to a possibility of a rogue Extended Service Set (ESS). This is discussed in more detail in Section 8.2.2.

[25] [O'Ha1999], p. 43.
[26] [O'Ha1999], pp. 59, 83–84.

8.2.1.8 Summary

This section has hypothesized about improper validation in the IEEE 802.11 protocol, and has shown where some of the weaknesses in the protocol may occur in future implementations. Most of the potential errors are caused by improper validation.

8.2.2 Application of VERDICT: Improper Exposure

In an IEEE 802.11 standard association, the mobile station must only be associated with one access point (AP). If the station is mobile and becomes associated with a second AP, the protocol dictates that the station must disassociate with the first AP [O'Ha1999]. However, there is no validation (see Section 8.2.1.3) to ensure that the mobile station is not associated with (i.e., *exposed to*) more than one AP.

Secondly, there is an improper exposure that can lead to compromise if it is not mitigated with proper validation. When an a station seeks to establish an Independent Basic Service Set (IBSS), beacon and probe frames are sent out. There is no validation to ensure that the sender is a legitimate member of the group. One machine can say that it is the central point for the IBSS, spoofing others in trusting it.[27]

In fact, the designers of IEEE 802.11 are aware of the possibility of a spoof. O'Hara and Petrick note the following in [O'Ha1999]:[28]

> It should be noted that this algorithm really only authenticates station A to station B. The IEEE 802.11 Working Group believed that the AP somehow occupied a more privileged position than the mobile stations when it came to authentication, since it is always the mobile station that initiates the authentication process. It is for this reason that it is only the mobile station that performs the encryption operation on the challenge text. This leaves the IEEE 802.11 WLAN open to some not so subtle security problems. In particular, a rogue AP could adopt

[27] An anecdotal example from Virginia Tech was when a computer was accidentally installed incorrectly, its IP address set to the lowest number on the subnet, making it the "gateway" for the rest of the subnet to access the rest of the campus. A "black hole" of sorts was created, and all packets on the subnet destined outside the subnet were sent to that "black hole" gateway machine, effectively cutting off the other machines on the subnet from communicating with the rest of campus.

[28] [O'Ha1999], p. 84.

> the SSID[29] of the ESS and announce its presence through the normal beaconing process. This would cause mobile stations to attempt to authenticate with the rogue. The rogue could always complete the authentication process with an indication of successful authentication. This would cause mobile stations to attempt to use the rogue for access to the WLAN. The rogue could then simply complete normal frame handshake procedures and the mobile stations would be the victims of a denial of service attack. A more active rogue could use more subtle means to attempt to gain access to the content of higher layer protocol frames containing user names, passwords, and other sensitive data. However, if the data is encrypted using WEP, it is highly unlikely that the rogue could successfully decrypt the information.

O'Hara and Petrick continue with their "solution:" "Fortunately for those interested in greater security for their WLANs, the IEEE 802.11 Working Group is currently discussing extensions to the authentication algorithms that will provide cryptographically secure, bidirectional authentication."[30] This is an improvement, but refer to [Bori2001] for reasons why WEP is not cryptographically secure.

Thirdly, the infrared transmissions of IEEE 802.11 is line-of-site and low powered; the range is only about fifty feet. However, the emanations of an RF IEEE 802.11 network transmitting in the ISM[31] band can transmit with about one watt of power. This translates to a range of a couple hundred of feet.[32] Since radio waves can penetrate through some walls, the IEEE 802.11 in an office building can be accessed in the parking lot. If the parking lot contains a rogue station listening for IEEE 802.11 networks, it can get some information about the network and may be able to join it. Poulsen describes a hacker named Peter Shipley who rides around the San Francisco Bay area in his car with a laptop listening for IEEE 802.11 networks [Poul2001]. By logging the locations of them with a GPS[33] receiver, he plans to demonstrate the insecurity of the protocols. Poulsen concludes with Shipley saying,

[29]Service Set Identity, [O'Ha1999], p. 56.
[30][O'Ha1999], p. 85.
[31]Industrial, Scientific, and Medical.
[32]Private e-mail from Kevin Krizman, a RF engineer.
[33]Global Positioning Satellite.

"I can give you the density of open networks an area, organized by zip code.... People don't believe there's a security problem if you don't prove it to them" [Poul2001].

8.2.3 Application of VERDICT: Improper Randomness

There exists two instances of improper randomness in the IEEE 802.11 protocol that could be potential vulnerabilities. The first is with the binary exponential backoff algorithm (Section 8.2.3.1), and the second is with the Orthogonal Frequency Division Multiplexing (OFDM) used in IEEE 802.11a (Section 8.2.3.2).

8.2.3.1 Binary Exponential Backoff Algorithm

When a collision occurs during an attempt to transmit, each station chooses a random number of time units to wait before retransmission. The number of time units to wait is chosen over a determined range. If a collision occurs a second time, the range over which the random number is chosen is doubled. If a collision occurs the third time, the range is doubled yet again [O'Ha1999].[34] If the random numbers are deterministic by a linear congruential generator (Equation 7.7, p. 210), the random number is based off of a seed, and the seed is found, a station could potentially be prevented from transmitting. That is, if the pseudo-random time period is known, an adversary can jam that particular time unit, making the target try again and again. However, if the goal is to prevent transmission, a denial of service (DoS) attack could be launched to effectively jam *all* transmissions, without knowing exactly what time slot the target unit will be transmitting.

8.2.3.2 Orthogonal Frequency Division Multiplexing (OFDM)

Due to potentially long strings of 1s and 0s using OFDM, the data is scrambled to randomize the data. Since the initial state of the scrambler is randomly chosen, care must be taken to ensure proper randomness [O'Ha1999].[35] Although this is probably not designed as a security measure, it is an aspect of randomness that could be subverted.

[34] [O'Ha1999], pp. 25–26.
[35] [O'Ha1999], pp. 142–143.

8.2.3.3 Summary

This section has shown that there exist places in the IEEE 802.11 protocol that could be vulnerable due to improper randomness. The next section (Section 8.2.4) applies the Improper Deallocation category to IEEE 802.11.

8.2.4 Application of VERDICT: Improper Deallocation (Residuals)

Upon review of the designer's handbook [O'Ha1999], no improper deallocation or residuals were found.

8.2.5 Summary of Application of VERDICT to IEEE 802.11

This section applies VERDICT to IEEE 802.11 and theorizes vulnerabilities in the protocol. Improper validation, exposure, and randomness are shown to exist in the protocol.

8.3 Summary

This chapter gives an overview of the IEEE 802.11 wireless LAN protocol and applies VERDICT to the protocol. Numerous vulnerabilities are found in regard to improper validation, exposure, and randomness. If these errors are verified in actual implementations of the IEEE 802.11 protocol, automated attack scripts can be developed to cause a potentially massive assault on the infrastructure of an IEEE 802.11 network. The next and final chapter (Chapter 9) presents conclusions about the dissertation, and appendices follow outlining various computer attacks in the literature (Appendix A) and computer security wisdom in the art of computer security (Appendix B).

Chapter 9

Conclusions

This dissertation contributes to the field of computer security by showing all computer attack taxonomies are similar to each other; constructing a new holistic taxonomy called **VERDICT** (**V**alidation, **E**xposure, **R**andomness, **D**eallocation **I**mproper **C**onditions **T**axonomy); creating methodologies for applying VERDICT; and applying VERDICT to IEEE 802.11 showing numerous potential security breaches.

9.1 Dissertation Hypothesis

Realizing that computer attacks today were the same types of attacks occurring in the 1970s (see Appendix A), an investigation of computer attacks and computer attack taxonomies began. Because computer attacks were similar to attacks in years past, a research hypothesis was formulated:

> ***A finite number of types of computer attacks and vulnerabilities can be classified into a taxonomy, and the taxonomy along with applicable methodologies can be used to predict future attacks.***

9.2 Discussion of Dissertation Research

The work presented in this dissertation concludes that there was a finite number of types of computer attacks and vulnerabilities. In the past those attacks have been classified into computer attack taxonomies; however, only Bishop's work [Bish1995] noted that the categories of two taxonomies developed in the 1970s (Protection Analysis (PA) [Bisb1978] and Research in Secured Operating Systems (RISOS) [Abbo1976]) were similar and could be matched. This dissertation greatly expanded the findings of Bishop and compared over twenty taxonomies in the literature. The resulting discovery is that almost all the categories in the taxonomies can be matched with each other; hence, the taxonomies were similar and there was a finite number of computer attacks and vulnerabilities.

Reviewing the attacks in the literature, it was found that some attacks caused other attacks. These effects (i.e., other attacks) were listed alongside the attacks causing the effects. Hence, it was found that the common taxonomy needed to find the *causes* of vulnerabilities and not just list possible effects or side effects of the original vulnerabilities. In fact, the reason for division of one vulnerability from another (*Fundamentum Divisionis*) needed to be described. After a background and literature survey of problems in computer security, assumptions made in computer security, why computers are not secure, penetration testing, computer attack taxonomies, and wireless networking security was presented in Chapter 2, a study of computer attack taxonomies was described in detail in Chapter 3. More specific to attacks on computers themselves, operating system integrity flaws taxonomies were investigated in Chapter 4.

Seeing that all operating system integrity flaw taxonomies were similar, the most complete of the taxonomies in breadth (PA) was expanded to include improper randomness and similar categories in PA were collapsed to one. The resulting taxonomy is called **VERDICT**. VERDICT shows that all computer attacks have as the root cause of them an improper condition of validation, exposure, randomness, or deallocation. This is key. If one can ensure correct validation, correct (limited) exposure, proper randomness, and proper deallocation, most security errors can be eliminated. The derivation of the PA taxonomy, the creation of VERDICT, and the individual categories of VERDICT were discussed in Chapter 5.

In order to show that VERDICT was superior to other taxonomies, I compared it to five operating system taxonomies and showed that the five taxonomies (Protection Analysis (PA) [Bisb1978]; Research in Secured Operating Systems (RISOS) [Abbo1976]; Neumann and Parker's Computer Misuse Categories [Neum1995]; Neumann and Parker's Trapdoor Attacks [Neum1995]; and McPhee's Integrity Flaws [McPh1974]) can be described by the four VERDICT categories; I also applied VERDICT to the top ten present Internet attacks and showed how those attacks can be categorized by VERDICT. This work was presented in Chapter 6.

In Chapter 7, I developed algorithms and methodologies for finding improper conditions for the four categories of VERDICT. Some of the methodologies applied to actual code, while others applied to the system or protocol in general. In the section on validation, algorithms from Bisbey and the Protection Analysis project [Bisb1978] were taken and supplemented with general methodologies to ensure validation in systems. In the exposure section, a similar algorithm to the in-to-out algorithm presented by Bisbey [Bisb1978] was developed along with the concept of an exposure matrix to ensure that objects in systems were not improperly exposed. The randomness section described different types of random numbers and what standard is needed to ensure adequate entropy. Finally, the three types of residuals were discussed in the deallocation section and the three algorithms presented by Bisbey [Bisb1978] were presented.

Finally, in Chapter 8, I utilized the methodologies outlined in Chapter 7 to discern how the up-and-coming IEEE 802.11 wireless LAN protocol has vulnerabilities that can be exploited. Problems with improper validation, exposure, and randomness are discovered and presented. IEEE 802.11 is a wireless physical (PHY) and medium access control (MAC) layer protocol. It is similar to the IEEE 802.3 "Ethernet" protocol. Because it is wireless, connections need to be made in order to transmit at the data link layer. After reviewing attacks on TCP/IP (Appendix A and elsewhere), improper validation attacks can be made on IEEE 802.11 that are very similar to present attacks. These attacks will be implemented and deployed in the future.

After this conclusion chapter, two appendices are included that outline the following: various computer attacks in the literature (Appendix A) and general wisdom on the art of

computer security (Appendix B). A thorough annotated bibliography of works cited follows the appendices.

9.3 Contributions to the Field of Computer Engineering

The research presented in this dissertation contributions to the field of computer engineering and the field of computer security. It has contributed the following:

1. There exists a finite number of types of computer attacks and vulnerabilities;
2. Computer attack taxonomies presented in the past have a common set of categories;
3. Those categories can be classified into a common unified taxonomy called **VERDICT**: **V**alidation, **E**xposure, **R**andomness, **D**eallocation **I**mproper **C**onditions **T**axonomy;
4. Methodologies and algorithms are developed for each of the four VERDICT categories;
5. Developed methodologies and algorithms are applied to predict future attacks in IEEE 802.11.

These contributions are very important because the dissertation has shown the four root causes of computer security. System administrators can search for the types of potential errors of VERDICT in present systems. More importantly, protocol and system designers can use VERDICT in the development phase to have a greater sense of security assurance. Finally, IEEE 802.11 is just beginning to establish itself as *the* basic Wireless LAN (WLAN) protocol for computers. Security vulnerabilities are shown in this dissertation that can be corrected if action is taken *now*; else, the world will see these attacks come to fruition.

9.4 Future Work

Since security is a broad field, much work can be studied in the future. In particular, more attacks can be classified with VERDICT. In addition, the methodologies presented in Chapter 7 can be expanded.

The Protection Analysis [Bisb1978] study wished to develop an ideal tool of a "protection evaluator" that could use patterns of raw errors to automatically notify users of potential vulnerabilities in the programs. That ideal was not accomplished, but tools may be written using the methodologies presented in this dissertation to find errors of improper validation, exposure, randomness, and deallocation (residuals).

Finally, validation of the vulnerabilities of IEEE 802.11 theorized in Chapter 8 could be completed using the actual IEEE 802.11 standard and equipment. VERDICT could be used to further expansion of Borisov's et al. paper [Bori2001].

9.5 Summary

This dissertation has reviewed the past literature of computer attack taxonomies and has found that there are a common set of attacks that occur on computers. I have derived from the past taxonomies a new comprehensive taxonomy called VERDICT: Validation, Exposure, Randomness, Deallocation Improper Conditions Taxonomy. I have verified that it is comprehensive and have developed methodologies and algorithms for using it. Finally, I have applied the taxonomy categories to IEEE 802.11, finding numerous vulnerabilities.

Accidents and human failures will occur. However, the weakest link in the security chain is the human. It is only when morals of the human heart are set straight that malicious attacks on innocent systems will truly cease. Research cannot overcome the evils of the soul.

Appendix A

Computer Attacks

This appendix is a listing of various types of computer attacks that are documented in the literature. This appendix is in no way comprehensive, but it is meant as a listing of examples of the various attacks.

A.1 Types of Computer Attacks

This section describes some specific computer attacks discussed in the literature. Many books and reports have been written on computer crime, including crime done with the help of computers and crime (attacks) on computers themselves in the 1970s [Park1973, Park1975a, Park1976, Whit1978], in the 1980s [Bequ1987, Perr1984], and the 1990s [Bloo1990, Icov1995]. In addition, much has been written on dealing with the crimes: Bequai [Bequ1978]; Cornwall [Corn1987]; Icove [Icov1995]; Krauss [Krau1979]; Parker [Park1975b, Park1976, Park1983, Park1989]; Sieber [Sieb1986]; and VanDuyn [VanD1985]. Certainly there has been a history of misuse. Indeed, academic articles outline their results of penetration attacks (See Section 2.4 for more information on penetration testing).

Computer attacks have traditionally attacked one or more of the three "legs" of security: something a user knows, something a user has, or something intrinsic to the user. The process of removing money from an ATM machine involves the first two: the bank card (something the user has) and the PIN[1] (something the user knows). The third leg of security (something

[1]Personal Identification Number

intrinsic to the user) is presently not used as much as the other two except in high security areas, but that is changing with the advent of biometrics. If the user had to submit to a thumbprint scan in addition to the card and the password (or PIN), the three legs of security would all be in use.

Attacks that were seen the 1970s are seen today, thirty years later. Overviewing papers covering specific attacks and papers covering past penetration results [Hebb1980, Karg1974, McPh1974, Wilk1981, Wood1990], we can see classes of attacks emerge. The following sections outline the specific attacks seen throughout the past.

A.1.1 Input/Output

This was a fruitful area of penetration in the 1970s. [Hebb1980] found that the Michigan Terminal System (MTS) did not have problems in with its I/O system, rendering little fruit in the penetration testing. The I/O system on a computer has not been a problem in the recent years, unless one counts remote network attacks. Network attacks are a fruitful area today, as will be seen in later sections. However, if one counts the input and output of parameters into a program, there are security problems today. See Section A.1.10 for a discussion on parameter checking and how programs can be compromised with parameters. In addition, if shell escape characters can be sent into a program via configuration files or mailed, compromise can happen.

A.1.2 Design Oversights

Design oversights is a broad category of attacks that exploit the fact that some aspect of the system was designed incorrectly. [Hebb1980] found that the system could be made to store data in an unprotected segment that a user could alter. By doing this, data could subsequently be loaded into the system segment. This overarching category was the second of two types of flaws that Hebbard found. The first, parameter checking, is covered in Section A.1.10.

A.1.3 File Security

One of the four areas of penetration testing on the Burroughs system was file security [Wilk1981]. In this penetration test, direct manipulation of tapes could be done to make a valid compiler; see [Thom1984] for the seminal paper on compiler modifications, and [Boyl1999] for a more up to date survey. From there, similar to [Thom1984], almost anything could be done.

A.1.4 Resource Limits

The second of the four areas of penetration testing in [Wilk1981] was the investigation of resource limits. If one could use all of the available resources, certain aspects of whatever resources one was using would fail and could cause holes of exploitation. Even today, one can fill up a disk, and the program writing audit logs to a file cannot write out the errors.

If we consider the space in a fixed size buffer and we place more characters in the buffer than it can hold, the limits of the resource (the buffer) will be reached and overflowed. Buffer and numeric overflows (Section A.1.15) are subsets of this type of attack.

A.1.5 Accounting Methods

In the past, if one had access to the punched cards that a program was on, one could change the methods on which the usage was accounted [Wilk1981]. This attack is much less likely today, as much of the world does not use punch cards.

A.1.6 Disruption/Denial-of-Service (DoS) Attacks

This was the final area of penetration testing that Wilkinson did in his penetration testing [Wilk1981]. However, today, disruptions include the popular Denial-of-Service (DoS) attacks. Mudge cites them in [Mudg1997] with attacks on Windows NT, such as WinNuke[2], a program to send an out of band (OOB) [Stev1994] data to the NetBIOS port causing a computer running Windows to degrade by dropping carrier or turning the screen white. This area

[2]WinNuke has been written in multiple lines of C or Perl. A one line Perl script even exists.

was studied extensively at Iowa State [Rich1999, Rich2001]. The outcome of many other computer attacks listed in Section A.1 can result in a DoS attack.

A.1.7 Object Reuse / Residuals

When memory is allocated for a particular object, used, and then returned to the main heap, one must make sure that the next usage of the object will not have access to the old bits of data stored in it. The user or the operating system must make sure to clean it out before object reuse. This error has been seen in [Wood1990] and is mentioned in [Vene1996]. Venema wishes that a modified *malloc()*[3] be used to automatically wipe the memory upon release, lest other processes can allocate large chunks of memory and search it for useful pieces of information, such as secret data. For this reason, Venema discourages secret data in memory of unprivileged programs.

A.1.8 Noncaptive Environments

FTP servers have been in use for many years. Administrators gave access to anonymous users via anonymous FTP. When the World Wide Web (WWW) was developed, the same principle applied. Administrators gave access to the http tree. When Microsoft first put out their FTP server, one could go from the root of the FTP access tree "\" and go up another level so that one was out of the FTP server tree [Mudg1997].

Even when a user is able to write to a piece of memory that they should not be able to (such as system memory or memory that should be system memory), the operating system is giving the user an environment that is not captive. This causes vulnerabilities for privilege escalation attacks (See Section A.1.9) [Mudg1997].

Another example of this is the Stop-L1 keyboard combination on Sun computers. When the stop key was held down while the L1 key was pushed, a system monitor[4] could be accessed. By noting where in memory the user's ordinary non-privileged shell's data structure began, it is possible to change the bits in the structure of the shell that represent the user

[3] *malloc()* is a standard C function for **m**emory **alloc**ate that returns a pointer to a block of memory dynamically allocated from the heap.

[4] This is a program that allows direct manipulation of any piece of memory. It was used to debug operating systems, but has great misuse potential if ordinary users can access this.

who owns it to 0. This effectively changes the ownership of the shell to root, and on exiting the monitor, the shell had root privileges.[5] Thus a privilege escalation attack could be done.

A.1.9 Privilege Escalation

Privilege escalation is the end result of other types of attacks, including object reuse, file security, etc. This concept of privilege escalation is to make the machine change one user's privilege levels to a higher one. It is most commonly exploited in the UNIX world by the Set User ID (SUID) set of programs.

In UNIX, files are accessed by referencing a set of bits that tell who can access the file, be it the user, a group that the user is in, or the world. But beyond that, there are still only two levels of privilege, root and non-root. One either has the power to access any file and do anything as root, or not. Once root is obtained, the user is now said to have obtained or gotten root. In fact, since root can do anything on the system (box), it is said to be "owned."

SUID programs allow a program to have the user id (that user who created or owns the program) to have root level privileges. This is to enable the mailer program (sendmail) to be able to write to all users' mailboxes with incoming mail. The password program in UNIX (passwd) needs to access the password file to update users' passwords. The problem with SUID programs is if the program can be made to run arbitrary code, it would run that code as root and anything can be done. Too many times, programmers need a little more privilege and make their programs SUID without totally checking to make sure that the program needs all that untapped power.

How can one make a SUID program to run arbitrary code? If one were able to cause a UNIX program to crash by means of escape sequences or other interprocess signals, a core file may be generated that would be writable and marked as SUID [Gram1984]. Over a decade later, [Mudg1997] describes privilege escalation attacks in Windows NT that allowed a user to modify memory that enabled an account to be added to the Administrator group.

[5]This type of shell is known as a rootshell.

A.1.10 Parameter Checking

In an analysis of the Michigan Terminal System (MTS), [Hebb1980] describes how a flaw in parameter checking could allow a user to store arbitrary bitstrings into a system segment. He names parameter checking as one of the types of errors found in their penetration testing. The other, design oversights, are covered in Section A.1.2. In [Vene1992], Venema described his TCP WRAPPERS software that, if probed, would send a finger request back to the probing host; this request is known as a reverse finger. This was part of his described "booby trap" methods in the host system. The shell command that would execute this reverse finger was:

finger -l @%h | /usr/ucb/mail root

where %h was the name of the probing host. The problem with this shell sequence, as described in his later paper [Vene1996], was that it substituted host names received from the Domain Name System (DNS) into the finger command. Since DNS is easily able to be forged, almost any sequence of characters could be inserted into a command running with root privileges. With that, unfortunate consequences could happen.

Parameter checking is a major type of attack today when applied to passing environment variables into executable programs. In [Bish1999], Bishop shows numerous examples of using the **PATH** and the **IFS**[6] environment variables to have a SUID program run a program of their own choosing. The moral is, when parsing parameters is one should not look for the "bad" characters or sequence of characters that may cause a vulnerability to be exploited, but to look for a set of "good" characters that one knows is good. In essence, "do not look for the bad; look for the good." A similar problem is described as the fourth in McPhee's general classes of integrity problems, known as, "user data passed as system data" [McPh1974].

A.1.11 Weak Passwords

This attack has been around for a long time. This is one of the fundamental problems with security today. Users want security, but do not want to be hassled with passwords. Passwords

[6]IFS is the environment variable that determines what characters are considered white space.

are written down (sometimes on notes attached to the terminal itself) so that the security of the password is compromised. Venema [Vene1996] starts off his paper of lessons learned in computer security with a discussion of weak passwords and the effects that they cause. Not only does this affect login security but also any tokens or keys that are generated. Kerberos keys and X Window cookies that are predictable (not random) will most likely eventually fail. See Section A.1.13 for more information on picking random numbers. To generate a secret password, you need a secret to begin with.

Mudge describes a problem in Windows NT's recent password scheme [Mudg1997]. Microsoft's LANMAN password scheme (only slightly weaker than the NT password scheme), "hashes the passwords in a predictable way and does not use salting (the process of inserting several random bits into the hash)" [Mudg1997].

A.1.12 Version Rollback Attack

Almost every time that software is upgraded, it seeks to be backwards compatible. The problem is that while the security features may have been updated on the new version, the features are not implemented on the old version. A Version Rollback Attack makes the new system believe that it is talking to an older system, thus making the new system run with the insecurities of the old version's program.

A.1.13 Lack of Randomness

See Section 5.3.3 for a description of randomness.

A.1.14 Nonunique Identification of System Resources

McPhee [McPh1974] identifies seven classes of integrity problems. The second class listed is Nonunique identification of system resources. This particular problem and his other six classes are compared in Section 4.4. He gives an example of having an object in use by the system and a user. The operating system must make sure that the two objects do not refer to the same item, lest the user delete the object while the system still uses it.

A more current example in [Vene1996] is the case of identical Network File System (NFS) file handles. NFS sets the handle of the file system with a pseudo-random number initialized by the time of day and the process ID. Some systems did not initialize the time of day, so some machines had the same NFS handles. This problem is a case of nonunique identification of system resources (NFS handles), but it also falls under the problem of not having enough randomness (see Section A.1.13).

A.1.15 Overflowing Bounds

Overflowing bounds is arguably the most used attack in the 1990s [Cowa2000]. The most common form is known as the buffer overflow. This is where too much data is entered in a fixed length character buffer. The buffer fills up, and the rest of the characters must go beyond the end of the buffer. If one knows what is beyond the buffer (stack return addresses, etc.), one can cause the flow of the program to be diverted by placing a user defined address in the return address register [Bish1999, Dild1998]. There are many ways to counter the classic buffer overflow problem. Cowan's StackGuard [Cowa1998] is one example that watches the stack for changes in the return address through the use of a "canary"[7] word which is a checked before and after the function is run. The canary word is put in when the source code links with the StackGuard library.

Bishop reviews "numeric overflow" attacks in [Bish1999], one which the numeric ID are overflowed. For example, a 32-bit NFS[8] ID can be passed to the UNIX kernel, which only has UIDs[9] ($2^{16} - 1$) or less. The UNIX kernel strips the top 16 bits off, making an NFS ID of 2^{17} ('1' followed by sixteen '0's) into 0 (only the lower 16 bits pass through). Since the UID is zero, root access occurs[10]. These numeric overflows are similar to buffer overflows, because too many bits are put into a fixed sized buffer.

[7] A direct descendent of the Welsh miner's canary. [Cowa1998]
[8] Network File System
[9] User ID
[10] Zero is not allowed as a valid NFS ID to begin with, but 2^{17} is.

A.1.16 Race Conditions

Race conditions exist in both hardware and software. Venema [Vene1996] describes a race condition to read the core dump generated by a signal between the time the login process switches to the user and the user's shell is run. Bishop [Bish1996b] shows a race condition in file access that is also outlined in [Bish1996a].

A specific type of race condition is called a TOCTTOU (Time-Of-Check-to-Time-Of-Use) [McPh1974, Bish1999]. As seen in Section 2.4.2.2.4 on page 36, TOCTTOU is an attack that results from an object changing from the time of the security check to the time of the object's use. TOCTTOU is described in [McPh1974] and is a subclass of race conditions.

A.1.17 Salami Attacks

As salami is composed of little bits of meat and other mysterious things, so a salami attack takes bits of information to generate a whole attack. It is most famously noted in banking, where a theoretical account checking computer program could take the fractions of a penny generated each time a remainder is generated and save them to a "salami" account. The owners of the legitimate accounts would not notice fractions of pennies of interest being siphoned away each day interest is calculated, but the result would certainly add up.

A.1.18 DoS SYN Flood Attacks

TCP/IP's SYN Flood attack is described in Section 8.2.1.4 on page 226. By accepting numerous SYN packets and keeping the half-open connections, the amount of connection buffers run out, causing a Denial of Service (DoS) attack and preventing legitimate users from connecting. For a detailed account on the attack and a review of solutions including a solution of their own *synkill*, see [Schu1997].

A.1.19 Electromagnetic Eavesdropping and TEMPEST

In 1985, Wim van Eck published an article on the eavesdropping on the emissions of video display units that introduced an entirely new concept to the field of computer security.[11] It was stated that with a small amount of commercial off the shelf (COTS) equipment, emanations from computer monitors can be captured and displayed on other video screens from up to one kilometer away [vanE1985]. Some technical information was intentionally left out of the article, but van Eck's reply to an inquiry and a block diagram of a receiver is found in [High1988].

According to [High1988], "Information about this type of eavesdropping has been classified for about 20 years. The Tempest (**T**ransient **E**lectro**M**agnetic **P**ulse **E**manation **ST**andard) project has been a joint research and development effort of the U.S. National Security Agency (NSA) and the Department of Defense (DoD). Even the program's name had been classified for most of that period" [High1988]. Schwartau devotes a chapter to Van Eck and follow ups of his and other related work [Schw1996]. Its effects can be partially prevented by architecture techniques as described in [GR1995]. Anything with a wire can act as an antenna; hence, information on such devices as printers, etc., can be captured. Pipes and conduits can also be used as emanation points. In summary, this is a very fruitful area attack and research.

A.2 Further Information

This section lists other papers that may not have been mentioned in the body of this dissertation that have studied attacks on computers and the attacks themselves therein. The specifics of the papers are not enumerated, but they have much information about computer attacks. Many of these are seminal papers; consult the annotated bibliography for more information.

- *Security Problems in the TCP/IP Protocol Suite* [Bell1989]

[11] Other supplementary articles are in the June 1986 and September 1986 of the journal Computers & Security. In addition, there is an article, "The Tempest over Leaking Computers" published in the Winter 1988 issue of *Abacus* [High1988].

- *There Be Dragons* [Bell1992]
- *Packets Found on an Internet* [Bell1993]
- *Problem Areas for IP Security Protocols* [Bell1996b]
- *Detecting Disruptive Routers: A Distributed Network Monitoring Approach* [Brad1998]
- *An Evening with Berferd in which a Cracker is Lured, Endured, and Studied* [Ches1992]
- *Computer Viruses: Theory and Experiments* [Cohe1987]
- *Internet Holes: 50 Ways to Attack Your Web Systems* [Cohe1995]
- *Internet Holes — Eliminating IP Address Forgery* [Cohe1996]
- *Information System Attacks: A Preliminary Classification Scheme* [Cohe1997a]
- *ARPANET Disruptions: Insight into Future Catastrophes* [Croc1989]
- *Internet Security Attacks at the Basic Levels* [deVi1998]
- *Internet Vulnerabilities Related to TCP/IP and T/TCP* [deVi1999]
- *The COPS Security Checker System* [Farm1990]
- *Web Spoofing: An Internet Con Game* [Felt1997]
- *Reducing the Vulnerability of Dynamic Computer Networks* [Finn1988]
- *Attack Class: Address Spoofing* [Hebe1996]
- *A Simple Active Attack Against TCP* [Jonc1995]
- *Penetrating Computer Systems and Networks* [Kaba1995]
- *ARPANET Lessons* [Klei1976]
- *Protocol Traps in Computer Networks — A Catalog* [Lai1982]
- *A Weakness in the 4.2BSD Unix TCP/IP Software* [Morr1985a]

- *Déjà Vu All Over Again* [Mudg1997]
- *Subversion: The Neglected Aspect of Computer Security* [Myer1980]
- *Techniques Adopted By 'System Crackers' When Attempting To Break Into Corporate or Sensitive Private Networks* [NSS1998]
- *TCP Congestion Control with a Misbehaving Receiver* [Sava1999]
- *Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP)* [Schn1998a]
- *Internet Sniffer Attacks* [Schu1995]
- *Reflections on Trusting Trust* [Thom1984]

Appendix B

Art of Security

There seems to be a distinction and a desire to classify principles, characteristics, and tenets of security. I distinguish these the following ways: **Principles** are those ideas that encompass the goals of security. That is, confidentiality, integrity, authority, and availability. Howard [Howa1997] has a category "Results" that contains these principles of security: *Corruption of Information* (integrity); *Disclosure of Information* (confidentiality); *Theft of Service* (authority);[1] and *Denial-of-Service* (availability).

But acceptability encompasses all. For example, one must know, when speaking of integrity, whether that means integrity of data or integrity of policy. That is, when one browsed (CM22 — see Section 6.3.22 on page 177), the integrity of the data itself would not be violated (nothing would change except for a few log files). However, the integrity (soundness) of the policy would be violated (perhaps the data should not be read at all). One needs to know whether it is acceptable to read the data. And integrity may not even be the start of the problem. Subversion [Myer1980] may *cause* the loss of integrity over time.

When one refers to availability, one must differentiate between availability of data and the availability of service. One needs to look at the security policy and determine what the acceptability parameters are. For example, suppose the power is turned off to a computer holding data, and that data has no backup. The data itself is not available; this would seem to violate the availability requirement of security. If the power outage was caused as

[1] I use authority because Howard defines *Theft of Service* as "...unauthorized use of computer or network services without degrading the service to other users."

part of a plan to physically attack the system, it would violate the requirement. But if the computer was routinely turned off at night, the policy is not violated. The data is not presently available, but it *can be made* available.

It is the same with getting root. If someone gets root, the policy *may* be violated. But if the "admin" (administrator) gets root, the policy is not violated — unless someone is spoofing the "admin" account. The policy of acceptability needs to be defined; however, it may be difficult to take into account situations like this to generate a solid policy.[2]

Characteristics of security are those flaws that taken together form a vulnerability. Bishop [Bish1999] expands biological theory (cited in Krsul [Krsu1998]) that taxonomies should be divided along characteristics of an object to be categorized (see Section 2.5.2 on page 40.

Tenets of security is a collection of wisdom, a collection of ideas of how to make programs and systems secure. Neumann's collection of tenets is outlined in Section B.1 while Hoffman's collection is in Section B.2. Dennis Director's four "Laws" of the computer are outlined in Section B.3. My collection of wisdom is outlined in Section B.4.

B.1 The "Eggs"hortations of Neumann

Neumann gives a pun laden lesson from his experiences in computer attacks [Neum1995]:

- *Do not put all your eggs in one basket.* Centralized solutions are inherently risky if a weak-link failure can knock out the entire system. Even if the centralized mechanism is designed to be highly robust, it can still fail. If everything depends on it, the design is a poor one.
- *Do not have too many baskets.* A risk of the distributed systems... is that physical dispersion, intercommunications, distributed control, configuration control, and redundancy management may all become so complex that the

[2]Take the statement: "This statement is false." Was the previous statement in quotations true or false? The answer is it neither true or false. But does it have to be either? Consider: if one assumes the statement in question is true, then by the reading of the statement, one would conclude that the statement is false. On the other hand, if one assumes the statement in question is false, then by the reading of the statement, one would conclude that the statement is false (double negative), making the statement true....

new global fault modes cannot be accommodated adequately, as exemplified by the 1980 ARPAnet and 1990 AT&T collapses... There is also a risk of overdependence on individual baskets, as suggested by the Lamport quote... [in Section B.4].

- *Do not have too many eggs.* There are saturation effects of trying to manage too many objects all at the same time, irrespective of whether the eggs are all in one basket or are dispersed widely.
- *Do not have too few eggs.* There are problems relating to multiplexing among inadequate resources, recovering from malfunctions, and providing timely alternatives.
- *Avoid baskets and eggs of poor quality.* Poor quality may result from improper care in the production process and shabby quality control One rotten egg can spoil the whole crate, just as entire networks have had great falls because of uncovered fault nodes. Furthermore, a rotten egg in one basket rather surprisingly may be able to spoil the other baskets as well.
- *Know in advance what you are trying to do.* Look at the big picture. Are you trying to make a cake, an omelette, an operating system, or a worldwide network?
- *Be careful how you choose and organize your roosters.* Chief programmers and chief designers can be very effective, but only if they are genuinely competent. Otherwise, they can lead to more problems than they can solve.
- *Be careful how you choose and organize your hens.* Structuring your system into layers can help, or am I egging you on too much?

B.2 Tenets of Hoffman

Reprinted in [Hoff1990], Rochlis and Eichin [Roch1989] outline their conclusions at the end of their paper, *With Microscope & Tweezers: The Worm from MIT's Perspective*:

- *Least privilege.* Those ignoring this face consequences.

- *"We have met the enemy and he is us."* Insiders like Morris do damage, sometimes the most damage.

- *Diversity is good.* Similar to biological diversity, one disease will not kill everyone. However, with Microsoft Windows on 85% of the computers in the world, this principle is not adhered to.

- *"The cure shouldn't be worse than the disease."* Sometimes restoring from backups is cheaper than trying to figure out damage that occurred.

- *Defenses must be made at the host level, not the network level.* Application programs are the problems, not the network. The author disagrees with this assessment, since the network can be taken down and *no one* would be able to communicate.

- *Logging information is important.* This helps in the recovery effort and to determine what was damaged.

- *Denial of service attacks are easy.* Again, one should also protect the network.

- *A central security fix repository may be a good idea.* This would be to collect information and patches for all to use. Repositories are in use today, but there is not one central one.

- *Knee-jerk reactions should be avoided.* Sharing information helps in the end.

B.3 Law and Order for the Personal Computer

An article of this section's name is included as Article 38 in [Denn1990b]. A "philosophy" of computing is presented by Dennis Director as four "laws". They are the following:

- ***The First Law: Do Not Accept That the Newest and the Latest Is the Best*** Although fixes and patches are very important to the system administrator, more bugs are often introduced into the system. One should be careful as to what exactly gets fixed with the introduction of the patch.

- ***The Second Law: Do Not Byte off More Features than You Can Swallow***[3] Most users only use a core set of features in a software package. With more features added come the possibility that errors could occur in those features. The feature of KISS[4] needs to be observed and maintained.

- ***The Third Law: Do Not Automatically Assume That Automatic Systems Are Automatically Better*** Failures can ripple through networked systems that automatically try and "fix" themselves from neighbors. Remote controls, if not secured, can open up more holes and potentially cause more damage and downside than the upside of the benefits of the remote control.

- ***The Fourth Law: Do Not Overlook the Danger from Within*** Either from accidents or on purpose, the insider often causes the most damage.

B.4 Tao of Security Tenets

The following is a collection of tenets about security. Similar to Tzu's The Art of War [Tzu1963], the collection is meant as a philosophy of security, or the art of security.

- Trust nothing.
- Verify everything. As Ronald Reagan said about the Soviets: "Trust but verify."
- Think like an attacker.
- Be explicit in programs.
- Do not look for the bad; look for the good. [Bish1999]
- Do not trust or depend upon programs not designed for security. [Vene1996]
- "In order to generate a secret password[5] you need a secret to begin with." [Vene1996]
- Clear memory before releasing it.

[3] (sic)
[4] Keep It Simple, Stupid.
[5] Or any other secret for that matter

- Do not go for a ride with a stranger [Gram1984]. From using "cu," a program for "call UNIX" that enabled one machine to call another. These machines were untrusted. Just as parents give the advise to their children, so must we not trust a stranger's computer.

- How do we stop these errors from occurring? Theo de Raadt and a team of computer security experts went through each line of code in their OpenBSD operating system finding programming bugs. They looked at parameter passing, which could have the potential for causing buffer overflows, and other security programming flaws. Matt Bishop gives a seminar on secure programming [Bish1999] that all programmers in education should be teaching.

- Sometimes, many vulnerabilities are necessary for an attack. Sometimes, many attacks are needed for a penetration. Sometimes, many penetrations are needed for a compromise? Sometimes, many compromise are needed for a system compromise? Sometimes, just one compromise, one penetration, one attack, or one vulnerability is needed. Sometimes, just one.

- Read the manual or RFC. If it says, "you MUST not do this," try it; it will probably break because overflowing bounds (Section A.1.15) will probably occur. The manual is one of the greatest assets to a penetrator.

- Leslie Lamport is quoted in [Neum1995] as saying, "A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable." See Neumann's quotes about too many "eggs" in Section B.1.

- While the defender is holding down all fronts, an attacker only needs to breach one. From [Tzu1963]: "For if he prepares to the front his rear will be weak, and if to the rear, his front will be fragile. If he prepares to the left, his right will be vulnerable and if to the right, there will be few on his left. And if there is no place he does not make preparations there is no place he not vulnerable."

- Judgement comes from experience; experience comes from poor judgement.[6]

[6]This is a quote entitled Robert E. Lee's Truce on a "Murphy's Computer Law" poster, SP 155 Copyright

- "Misplaced confidence in the security of a system is worse than having no confidence at all in its security." [Brin1995]

- It has been said, "Those who cannot remember the past are condemned to repeat it." Brinkley [Brin1995] continues this thought with: "Do not entrust security to technology unless that technology is demonstrably trustworthy, and the absence of demonstrated compromise is absolutely not a demonstration of security."

- Return plaintext with plaintext; return ciphertext with ciphertext. Bellovin [Bell1996b] concludes that this will defend against some chosen plaintext attacks.

- A listing of general defense ideas is found in [Cohe1997b]. It is not "classification scheme" as Cohen suggests, but it is a good list of security practices.

- Active languages can cause problems. Postscript and macros in Microsoft Word can cause damage as viruses. Does Active Networks pose a problem? Anytime data is treated as instructions, problems can occur.

- Just as in biology, homogeneity is bad. Diversity (heterogeneity) is good. A weakness in one machine can bring down the entire network if all of the machines on the network are the same machine, the same configuration, etc.

- Randy Marchany suggests a shift in the fruitful areas of attack. In a server/client model, one used to attack the server or the services offered by the server. Once the server became "fairly" stable, attacks were made against the means of transport between the client and the server such as sniffers. Once means to secure the transport were implemented (such as SSH[7]), attacks are now being made against the client itself. This is becoming more prevalent with the advent of small handheld devices such as the Palm Pilot. Will the client become the next fruitful area of attacks? Time will tell.

1984 Celestial Arts P.O. Box 7327, Berkeley, CA 94707. It is unknown whether this quote is truly from the famed Civil War general, Robert E. Lee. Incidentally, a famous quote Robert E. Lee did say at the Battle of Fredericksburg, "It is well that war is so terrible, lest man become too fond of it."

[7]Secure SHell

B.5 Summary

The philosophy of the art of security is essential meditation for the serious student of this field. These tenets should be taken to heart and utilized in the design of present and future systems. If not, the same vulnerabilities will arise continually until fixed. As the saying goes: "Those who ignore the past are doomed to repeat it."

Annotated Bibliography

[Abbo1976] R.P. Abbott, J.S. Chin, J.E. Donnelley, W.L. Konigsford, S. Tokubo, and D.A. Webb. Security Analysis and Enhancements of Computer Operating Systems. Technical Report NBSIR 76-1041, Lawrence Livermore Laboratory, Institute for Computer Sciences and Technology / National Bureau of Standards / Washington, DC 20234, April 1976. T.A. Linden, Editor; The RISOS Project.

The protection of computer resources, data of value, and individual privacy has motivated a concern for security of EDP installations, especially of the operating systems. In this report, three commercial operating systems are analyzed and security enhancements suggested. Because of the similarity of operating systems and their security problems, specific security flaws are formally classified according to a taxonomy developed here. This classification leads to a clearer understanding of security flaws and aids in analyzing new systems. The discussions of security flaws and the security enhancements offer a starting reference for planning a security investigation of an EDP installation's operating system.

[Abra1970] N. Abramson. The Aloha System — Another Alternative for Computer Communications. In *Proceedings Fall Joint Computing Conference, AFIPS Conference*, volume 37, page 37, 1970. November 17–19.

In September 1968 the University of Hawaii began work on a research program to investigate the use of radio communications for computer-computer and console-computer links. In this report we describe a remote-access computer system — THE ALOHA SYSTEM — under development as part of that research program[8] and discuss some advantages of radio communications for interactive users of a large computer system. Although THE ALOHA SYSTEM research program is composed of a large number of research projects, in this report we shall be concerned primarily with a novel form of random-access radio communications developed for use within THE ALOHA SYSTEM....

[Abra1995a] Marshall D. Abrams, Sushil Jajodia, and Harold J. Podell, editors. *Information Security: An Integrated Collection of Essays.* IEEE Computer Society Press, 10662 Los Vaqueros Circle P.O. Box 3014 Los Alamitos, CA 90720-1264, 1995.

This collection of essays provides a comprehensive summary of practice and research. The essays provide an overview of the vulnerabilities and threats to information security and introduce the important concepts and terms. Inaddition, the essays summarize the definitions and controls of the trusted computer system evaluation

[8] N ABRAMSON et al 1969 *annual report THE ALOHA SYSTEM* University of Hawaii Honolulu Hawaii January 1970.

criteria and discuss information security policy focusing on information control and dissemination. Recommendations are presented based on practical experience. Other essays explore the architectures used in the development of trusted relational database management systems, discuss the effects that multilevel DBMS security requirements can have on the system's data intergrity, and compare three research DBMS prototypes.

Additional essays identify the motivation for using formal methods across different development stages of a trusted computer system, feature a new approach to formal modeling of a trusted computer system, and present a new security model for mandatory access controls in object-oriented database systems. The book concludes with a list of acronyms, a glossary offering multiple definitions of terms, and a list of references from the text.

[Abra1995b] Marshall D. Abrams and Harold J. Podell. Local Area Networks. In Marshall D. Abrams, Sushil Jajodia, and Harold J. Podell, editors, *Information Security: An Integrated Collection of Essays*, chapter 16, pages 385–404. IEEE Computer Society Press, 1995.

Local area network (LAN) communications security is addressed in this essay. LANs are introduced as providing: (1) a private communications facility, (2) services over a relatively limited geographic area, (3) a high data rate for computer communications, and (4) common access to a wide range of devices and services. Security issues pertinent to LANs are discussed. For example, LANs share many security problems and approaches for their solutions with point-to-point conventional communications systems. In addition, LANs have some unique problems of their own: (1) universal data availability, (2) passive and active wiretap threats, (3) end-to-end access control, and (4) security group control.

Countermeasures include physical protection, and separation by physical, logical, and encryption methods. Trusted Network Interface Units, encryption, and key distribution are also discussed.

Examples are discussed to illustrate the different approaches to LAN security. The examples in this essay are a composite of several existing product features, selected to demonstrate the use of encryption for confidentiality, and trusted system technology for a local area network.

[Ahuj1996] Vijay Ahuja. *Network & Internet Security.* AP Professional, 1300 Boylston Street, Chestnut Hill, MA 02167, 1996.

This book undertakes the problems of network and Internet security by teaching appropriate ways to combat intrusions and viruses. It begins with a background of client/server networks and an overview of security risks, exposures, and threats to both the single workstation and the network. It then describes, more technically, the many different security elements and their uses, including user authentication, virus protection, and encryption. The book not only covers network and workstation security, but Internet security as well. Ahuja covers such important Internet issues as secure e-mail, electronic commerce, and data transfer.

This book should serve the needs of networking professionals who are faced with network security problems and require information on how to solve them. It also provides a broad understanding of data security topics and technologies.

[Amor1994] Edward G. Amoroso. *Fundamentals of Computer Security Technology.* Prentice-Hall PTR, 1994.

The primary goal of this book is to introduce critical issues in computer security technology to individuals who rely on computer and network systems in their work and need to protect information and resources from malicious tampering.

In his forward, Leonard LaPadula writes, "Students, teachers, engineers, and scientists interested in computer security have a new assistant in this book. Basing his approach and much of the material on extensive notes from his teaching experience, Dr. Amoroso has produced a book that sympathetically teaches and effectively summarizes computer security."

Students and professionals will benefit from the thorough coverage of fundamental topics including:

- Threat and vulnerability assessment,

- Security policy modeling,
- Safeguard and countermeasure selection,
- Network and database security, and
- Security Evaluation.

This book also includes an extensive annotated bibliography describing over 250 papers, reports, and texts dealing with computer security.

[Ande1972a] James P. Anderson. Computer Security Technology Planning Study. Technical Report ESD-TR-73-51, Vol. I, James P. Anderson & Co., Box 42, Fort Washington, PA 19034, October 1972. Contract No. F19628-72-C-0198 for Deputy for Command and Management Systems HQ Electronic Systems Division (AFSC) L.G. Hanscom Field, Bedford, MA 01730.

The results of a planning study for USAF multilevel computer security requirements are presented. The study recommends research and development urgently needed to provide secure information processing systems for command and control and support systems for the Air Force....

The principal unsolved technical problem found by the working group was that of how to provide multilevel resource and information sharing systems secure against the threat from a malicious user. This problem is neither hopeless nor solved. It is, however, perfectly clear to the panel that solutions to the problem will not occur spontaneously, nor will they come from the various well-intentioned attempts to provide security as an add-on to existing systems.

The reason that an add-on approach, which looks so appealing, will not suffice is that in order to provide defense against a malicious user, one must design the security controls into the operating system of a machine so as to not only control the actions of each user, but of the many parts of the operating system itself when it is acting on a user's behalf. It is this latter requirement that invalidates the concept of providing only those controls required by the security level of the information being processed on a system. The issue of computer security is one of completeness rather than degree, and a complete system will provide all of the controls necessary for a mixture of all security levels on a single system. It is the notion of completeness that compels one to take the position that security must be designed into systems at their inception.

The approach recommended in the development plan is to start with a statement of an ideal system, a model, and to refine and move the statement through various levels of design into the mechanisms that implement the model system. Other elements of the plan address ancillary developments needed to reduce costs or to support common applications.

The plan described in this report represents a coherent approach to attacking these problems. It is our opinion that attempting to solve the problem by piecemeal application of parts of this plan will not produce the desired results.

[Ande1972b] James P. Anderson. Computer Security Technology Planning Study. Technical Report ESD-TR-73-51, Vol. II, James P. Anderson & Co., Box 42, Fort Washington, PA 19034, October 1972. Contract No. F19628-72-C-0198 for Deputy for Command and Management Systems HQ Electronic Systems Division (AFSC) L.G. Hanscom Field, Bedford, MA 01730.

Details of a planning study for USAF computer security requirements are presented. An Advanced development and Engineering program to obtain an open-use, multilevel secure computing capability is described. Plans are also presented for the related developments of communications security products and the interim solution to present secure computing problems. Finally a Exploratory development plan complementary to the recommended Advanced and Engineering development plans is also included.

[Ande1980] James P. Anderson. Computer Security Threat Monitoring and Surveillance. Technical Report Contract 79F296400, James P. Anderson Co., Box 42 Fort Washington, PA 19034 (215) 646-4706, April 1980.

This is the final report of a study, the purpose of which was to improve the computer security auditing and surveillance capability of the customer's systems [Ande1980]. In this paper, Anderson introduces an alternate taxonomy of threats to computers. Those interested in threat analysis and taxonomies are directed toward this work [Amor1994].

[Ande1995] Ross Anderson and Roger Needham. Programming Satan's Computer. In Jan van Leeumen, editor, *Computer Science Today: Recent Trends and Developments*, number 1000 in Lecture Notes in Computer Science, pages 426–440. Springer-Verlag, Berlin, Heidelberg, New York, 1995.

Cryptographic protocols are used in distributed systems to identify users and authenticate transactions. They may involve the exchange of about 2–5 messages, and one might think that a program of this size would be fairly easy to get right. However, this is absolutely not the case: bugs are routinely found in well known protocols, and years after they were first published. The problem is the presence of a hostile opponent, who can alter messages at will. In effect, our task is to program a computer which gives answers which are subtly and maliciously wrong at the most inconvenient possible moment. This is a fascinating problem; and we hope that the lessons learned from programming Satan's computer may be helpful in tackling the more common problem of programming Murphy's.

[Ande2001] Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, 2001.

"Many people are anxious about Internet security for PCs and servers," says leading expert Ross Anderson, "as if that's all there is when in reality security problems have just begun. By 2003, there may be more mobile phones on the Net than PCs, and they will be quickly followed by network-connected devices from refrigerators to burglar alarms to heart monitors. How will we manage the risks?"

Dense with anecdotes and war stories, readable, up-to-date and full of pointers to recent research, this book will be invaluable to you if you have to design systems to be resilient in the face of malice as well as error. Anderson provides the tools and techniques you'll need, discusses what's gone wrong in the past, and shows you how to get your design right the first time around.

You don't need to be a security expert to understand Anderson's truly accessible discussion of:

- Security engineering basics, from protocols, cryptography, and access controls to the nuts and bolts of distributed systems
- The lowdown on biometrics, tamper resistance, security seals, copyright marking, and many other protection technologies — for many of them, this is the first detailed information in an accessible textbook
- What sort of attacks are done on a wide range of systems — from banking and medical records through burglar alarms and smart cards to mobile phones and e-commerce — and how to stop them
- Management and policy issues — how computer security interacts with the law and with corporate culture

[Anon1997] Anonymous. *Maximum Security: A Hacker's Guide to Protecting Your Internet Site and Network*. Sams.Net, 201 W. 103rd St., Indianapolis, IN 46290, 1997.

With increasing frequency, there are reports of crackers breaking into systems and stealing data or maliciously altering Web sites. *Maximum Security: A Hacker's Guide to Protecting Your Internet Site and Network* is designed for system administrators and managers who need to find out how to protect their computers, networks, and

Internet sites from these kinds of unauthorized intrusions. Written by an experienced hacker, this unique guide to Internet and network security identifies the security holes and faults inherent in a wide variety of computer systems and networks, and then describes how to go about fixing them.

[Arqu1993] John Arquilla and David Ronfeldt. Cyberwar is Coming! *Comparative Strategy*, 12(2):141–165, April–June 1993.

The information revolution and related organizational innovations are altering the nature of conflict and the kinds of military structures, doctrines, and strategies that will be needed. This study introduces two concepts for thinking about these issues: *cyberwar* and *netwar*.

Industrialization led to attritional warfare by massive armies (e.g., World War I). Mechanization led to maneuver predominated by tanks (e.g., World War II). The information revolution implies the rise of cyberwar, in which neither mass nor mobility will decide outcomes; instead, the side that knows more, that can disperse the fog of war yet enshroud an adversary in it, will enjoy decisive advantages.

Communications and intelligence have always been important. At a minimum, cyberwar implies that they will grow more so and will develop as adjuncts to overall military strategy. In this sense, it resembles existing notions of "information war" that emphasize C3I. However, the information revolution may imply overarching effects that necessitate substantial modifications to military organization and force posture. Cyberwar may be to the twenty first century what blitzkrieg was to the twentieth. It may also provide a way for the U.S. military to increase "punch" with less "paunch."

Whereas cyberwar refers to knowledge-related conflict at the military level, netwar applies to societal struggles most often associated with low intensity conflict by non-state actors, such as terrorist, drug cartels, or black market proliferators of weapons of mass destruction. Both concepts imply that future conflicts will be fought more by "networks" than by "hierarchies," and that whoever masters the network form will gain major advantages.

[Arqu1998] John Arquilla. The Great Cyberwar of 2002. *Wired*, pages 122–127, 160–170, February 1998.

NATO expands eastward, Taiwan declares independence, Russia and China form an alliance, North Korea violates its nuclear disarmament agreement, Iran and Iraq make peace, and Liddy Dole faces the biggest crisis of her presidency: The first global cyberwar, where the enemy is invisible, the battles virtual, and the casualties all too real.

[Ashi1993] D. Ashitey, A. Sheikh, and K.M.S. Murthy. Intelligent Personal Communication System. In *43rd IEEE Vehicular Technology Conference*, pages 696–699. IEEE, 1993.

This paper presents an architecture for personal communication system (PCS) based on an Intelligent Network (IN) infrastructure. Personal communication service is realized by identifying functions such as radio access, authentication, and location registration an incorporating them as service logics in the intelligent network. Call models based on the set of Trigger Check Points (TCPs) and Functional Entity Actions (FEAs) defines in CCITT Capability Set 1 are also presented to show that IN capabilities can be used to support most PCS features.

[Asla1995] Taimur Aslam. A Taxonomy of Security Faults in the UNIX Operating System. Master's thesis, Purdue University, August 1995.

Security in computer systems in important to ensure reliable operation and protect the integrity of stored information. Faults in the implementation can be exploited to breech security and penetrate an operating system. These faults must be identified, detected, and corrected to ensure reliability and safe-guard against denial of service, unauthorized modification of data, or disclosure of data.

We define a classification of security fault in the Unix operating system. We state the criteria used to categorize the faults and present examples of the different fault types.

We present the design and implementation details of a database to store vulnerability information collected from different sources. The data is organized according to our fault categories. The information in the database can be applied in static audit analysis of systems, intrusion detection, and fault detection. We also identify and describe software testing methods that should be effective in detecting different faults in our classification scheme.

[Asla1996] Taimur Aslam, Ivan Krsul, and Eugene H. Spafford. Use of A Taxonomy of Security Faults. Technical Report TR-96-051, Purdue University, West Lafayette, IN 47909-1398, September 1996. This paper to be presented at the 19th National Information Systems Security Conference, October 22–25, 1996, Baltimore, Maryland.

Security in computer systems is important so as to ensure reliable operation and to protect the integrity of stored information. Faults in the implementation of critical components can be exploited to breach security and penetrate a system. These faults must be identified, detected, and corrected to ensure reliability and safeguard against denial of service, unauthorized modification of data, or disclosure of information.

We define a classification of security faults in the Unix operating system. We state the criteria used to categorize the faults and present examples of the different fault types.

We present the design and implementation details of a prototype database to store vulnerability information collected from different sources. The data is organized according to our fault categories. The information in the database can be applied in static audit analysis of systems, intrusion detection, and fault detection. We also identify and describe software testing methods that should be effective in detecting different faults in our classification scheme.

[Atta1976] C. R. Attanasio, P. W. Markstein, and R. J. Phillips. Penetrating an operating system: a study of VM/370 integrity. *IBM System Journal*, 15(1):102–116, 1976.

Discussed is a methodology for discovering operating system design flaws as an approach to learning system design techniques that may make possible greater data security.

Input/output has been found to be involved in most of the weaknesses discovered by a study team in a particular version of the system.

Relative design simplicity was found to be the source of greatest protection against penetration efforts.

[Bace1995] Rebecca G. Bace and Marvin Schaefer. 'TSUPDOOD? Repackaged Problems for You and MMI. In *New Security Paradigms Workshop*, pages 2–10, National Security Agency and Arca Systems Inc., 1995. ACM SIGSAC.

Changes in computer usage have significantly changed the so-called computer security, network security and information security problems. The changes are largely due to rapid proliferation and interconnection of computers and the associated distribution of software. Of concern is the uncontrolled nature of this activity: systems and workstations are often interconnected without notice being given to all of the affected parties. The result has been increased user-perception of breaches in "security", especially in the form of computer takeover, data destruction, or service denial by virus, worm or trapdoor. It is expected that consciousness of these problems, and of confidentiality compromised, will increase in the coming months. It is posited that a principal cause of the problem is willful promiscuity and a pronounced lack of mutual suspicion. The separation kernel concept is revisited as a potential practical means of improving security protections consistent with preserving the use of legacy systems and of commercial products.

[Bake1996] Dixie B. Baker. Fortresses Built Upon Sand. In *New Security Paradigms Workshop*, pages 148–153, September 1996.

The current "trusted system" paradigm is built upon the notion of a Reference Monitor that assumes the existence of a well-defined security policy, a bounded system entity, and a centralized reference validation mechanism with knowledge of and control over the system entity. The "trusted system" paradigm is hierarchical: management defines the policy, the hardware and system software that comprise the trusted computing base enforce the policy, and applications must conform to the policy. This paradigm acknowledges that applications depend upon the hardware and operating system on which they run, and that assurance *assurance that they will execute safely is derived form the strength of this "trusted computing base."*....

The "obvious conclusion seems to be "The Emperor has no clothes!" The "trusted system" paradigm must not be working — what we need is a totally different paradigm!

It's obvious to even the casual observer that what we're doing now to make our systems safe and secure is not working. But is the "trusted system" paradigm at fault? Or are we just attempting to build our fortresses upon sand? Let's examine the perceived problems with the existing paradigm and one of the proposed solutions.

[Beha1997] Richard Behar. Who's reading your e-mail? *Fortune*, pages 56–61, 64–70, February 3 1997.

As the world gets networked, spies, rogue employees, and bored teens are invading companies' computers to make mischief, steal trade secrets — even sabotage careers.

[Beiz1990] Boris Beizer. *Software Testing Techniques*. Van Nostrand Reinhold, 115 Fifth Avenue, New York, New York 10003, second edition, 1990.

This book concerns testing techniques that are applied to individual routines. The companion volume, *Software System Testing and Quality Assurance* [BEIZ84: Beizer, B. *Software System Testing and Quality Assurance*. New York: Van Nostrand Reinhold, 1984], is concerned with integration testing, development of system test plans, software quality management, test teams, and software reliability. Most software is produced by the cooperative effort of many designers and programmers working over a period of years. The resulting product cannot be fully understood by any one person. Consequently, quality standards can only be achieved by emplacing effective managment and control methods. However, no matter how elegant the methods used to test a system, how complete the documentation, how structured the architecture, the development plans, the project reviews, the walkthroughs, the data-base management, the configuration control — no matter how advanced the entire panoply of techniques — all will come to nothing, and the project will fail, if the unit-level software, the individual routines, have not been properly tested. Quality assurance that ignores unit-level testing issues is a construct built on a foundation of sand.

[Bell1989] S.M. Bellovin. Security Problems in the TCP/IP Protocol Suite. *ACM Computer Communications Review*, 19(2):32–48, April 1989.

The TCP/IP protocol suite, which is very widely used today, was developed under the sponsorship of the Department of Defense. Despite that, there are a number of serious security flaws inherent in the protocols, regardless of the correctness of any implementations. We describe a variety of attacks based on these flaws, including sequence number spoofing, routing attacks, source address spoofing, and authentication attacks. We also present defenses against these attacks, and conclude with a discussion of broad-spectrum defenses such as encryption.

[Bell1992] Steven M. Bellovin. There Be Dragons. In *UNIX Security Symposium*. USENIX Association, July 30, 1992. smb@ulysses.att.com; paper found at http://csrc.nist.gov/secpubs/.

Our security gateway to the Internet, **research.att.com** provides only a limited set of services. Most of the standard services have been replaced by a variety of trap programs that look for attacks. Using these, we have detected a wide variety of pokes, ranging from simple doorknob-twisting to determined assaults. The attacks range from simple attempts to log in as **guest** to forged NFS packets. We believe that many other sites are being probed but are unaware of it: the standard network daemons do not provide administrators with either appropriate controls and filters or with the logging necessary to detect attacks.

[Bell1993] Steven M. Bellovin. Packets Found on an Internet. *Computer Communications Review*, 23(3):26–31, July 1993.

As part of our security measures, we spend a fair amount of time and effort looking for things that might otherwise be ignored. Apart from assorted attempted penetrations, we have also discovered many examples of anomalous behavior. These range from excessive ICMP messages to nominally-local broadcast packets that have reached us from around the world.

[Bell1994] Steven M. Bellovin and William R. Cheswick. Network Firewalls. *IEEE Communications Magazine*, 32(9):50–57, September 1994.

Computer security is a hard problem. Security on networked computers is much harder. Firewalls (barriers between two networks), when used properly can provide a significant increase in computer security.

[Bell1996a] S. Bellovin. Defending Against Sequence Number Attacks. Request for Comments (RFC) 1948, May 1996.

IP spoofing attacks based on sequence number spoofing have become a serious threat on the Internet (CERT Advisory CA-95:01). While ubiquitous cryptographic authentication is the right answer, we propose a simple modification to TCP implementations that should be a very substantial block to the current wave of attacks.

[Bell1996b] Steven M. Bellovin. Problem Areas for IP Security Protocols. In *Proceedings of the 6th USENIX Security Symposium: Focusing on Applications of Cryptography*, pages 205–214. USENIX, July 22–25 1996.

The Internet Engineering Task Force (IETF) is in the process of adopting standards for IP-layer encryption and authentication (IPSEC). We describe a number of attacks against various versions of these protocols, including confidentiality failures and authentication failures. The implications of these attacks are troubling for the utility of this entire effort.

[Bell1999] Steven M. Bellovin. Distributed Firewalls. *;login:*, pages 39–47, November 1999. USENIX Association Magazine.

Convential firewalls [5][9] rely on the notions of restricted topology and controlled entry points to function. More precisely, they rely on the assumption that everyone on one side of the entry point — the firewall — is to be trusted, and that everyone on the other side is, at least potentially, an enemy. The vastly expanded Internet connectivity in recent years has called that assumption into question. So-called "extranets" can allow outsiders to reach the "inside" of the firewall; on the other hand, telecommuters' machines that use the Internet for connectivity need protection when encrypted tunnels are not in place.

[Bequ1978] August Bequai. *Computer Crime*. D.C. Heath and Company, 1978.

[9] [Ches1994].

...Fewer than 1 percent of all computer crimes are uncovered. When finally discovered, the felon escapes justice by simply taking advantage of the legal maze we have created.

This book addresses the history and present dilemma posed by this felon. Chapter 1 deals with the criminology of computer crime. Chapters 2 and 3 deal with the problem of computer vulnerability and recommendations for improved security. Chapters 4 and 5 address the issue of present laws, both federal and local, to deal with the problem. Chapter 6 reviews the prosecutorial machinery and its shortcomings. Chapters 7 and 8 deal with the available investigatory machinery, both at the local and federal level. Without an adequate prosecutorial and investigatory apparatus, even the best of laws have their limitations.

Chapters 9 through 13 deal with evidentiary problems in the prosecution and conviction of computer felons. At present, both the prosecutor and litigant in computer-related litigation face serious obstacles. Chapters 14 and 15 deal with presently litigated cases involving computers; chapter 16, the last chapter, deals with the Electronic Funds Transfer System (EFTS) to adapt our legal system to the needs of an ever-growing technology, we and the problems it will bring about.

The computer is a marvel in its own right. It is the workhorse of the twentieth century, found in all facets of our economy. However, we must learn to safeguard it. If it is to be our "magic Genie," we must learn to harness it properly. This book is meant to offend no one, other than the computer felon, but it is meant to awaken us to a serious and growing problem, a problem aggravated by an antiquated and overbureaucraticized legal apparatus. At stake is our very form of government, for if we fail to adapt our legal system to the needs of an ever-growing technology, we will lose.

[Bequ1987] August Bequai. *Technocrimes*. D.C. Heath and Company, 1987.

While acknowledging that no modern society could stay intact for long without the tools of the high-tech revolution, this book identifies the potential for abuse of computer technology. This is a technology that can be easily corrupted by unethical people — criminals, political malcontents, and others who may use it to rob and manipulate society with impunity. In a sense, this book is a travelogue into our high-tech future, where all-too-realistic phantoms may haunt us. Contemplate a world in which new and more frightening methods of crime and mass destruction emerge. Ponder a cashless and paperless society, where the police track down the politically "undesirable" in a matter of microseconds, where terrorists and criminals murder by computer, and where industrial spies and saboteurs armed with portable computers threaten the West's entire financial foundation. *Technocrimes* journeys to the dark side of the high-tech revolution.

The unprecedented and accelerated changes brought about by the high-tech revolution constitute an awesome challenge to our political, social, and economic institutions. This book depicts what awaits us if we fail to understand and address the challenges of the postindustrial society. Starkly reminding us that even great civilizations can fall victim to their creations, *Technocrimes* raises the specter of a highly evolved society: a brave new world lacking in ethics, where humanity finds itself at the mercy of machines.

[Bert1992] Dimitri Bertsekas and Robert Gallager. *Data Networks*. Prentice-Hall Inc., Englewood Cliffs, New Jersey 07632, second edition, 1992.

Bertsekas and Gallager's definitive best-seller maintains its edge with a thorough revision and topical update. The authors present a clearly written but conceptually sound treatment of the basic principles of data networks. These principles are used to explain both existing networks and the evolution toward high-speed integrated networks.

CONTENT HIGHLIGHTS:

- NEW — High-speed networks with integrated voice, data, and video (BISDN and ATM switching), and high-speed local and metropolitan area networks (FDDI and DQDB).
- NEW — Internetworking and transport layer issues (TCP/IP, gateways, and bridges).
- Gives expanded coverage of queueing. The easily understandable style of the first edition is maintained, but many new results and applications have been added, providing insight and analytical tools for understanding data networks.
- Uses the principles of layering throughout while explaining its many variations in existing networks.
- Presents simplified and improved treatment of data link control, with many examples and insights into distributed algorithms and protocols.
- Discusses in-depth theoretical and practical aspects of routing and topological design.
- Covers the theory and practice of multiaccess communication, including collision resolution, carrier sensing, reservations, and local area networks.

- Provides expanded coverage of flow control emphasizing problems of congestion and delay requirements in integrated high-speed networks.

[Bisb1975] Richard Bisbey, II, Gerald Popek, and Jim Carlstedt. Protection Errors in Operating Systems: Inconsistency of a Single Data Value Over Time. Technical Report ISI/SR-75-4, Information Sciences Institute / University of Southern California, 4674 Admiralty Way / Marina del Rey / California 90291, December 1975. Reproduced by NTIS, U.S. Dept of Commerce, National Technical Information Service, Springfield, VA 22161.

This report describes a pattern-based approach for finding a general class of computer operating system errors characterized by the inconsistency of a data value between pairs of references. A formal description of the error class is given, both as a protection policy being enforced and as a violation of that policy, i.e., an error statement. A particular subclass of the general error class is then examined, i.e., those errors in which the data type is a parameter. A formal specification of a procedure for finding instances of the subclass is given with examples of errors found using the procedure.

This work has been performed under Advanced Research Projects Agency Contract DAHC15 72 C 0308. It is part of a larger effort to provide secureable operating systems in DOD environments.

[Bisb1976] Richard Bisbey, II, Jim Carlstedt, Dale Chase, and Dennis Hollingworth. Data Dependency Analysis. Technical Report ISI/RR-76-45, Information Sciences Institute / University of Southern California, 4676 Admiralty Way / Marina del Rey / California 90291, February 1976.

In order to understand the structure of computer programs and to detect certain types of protection errors in computer operating systems, it is often necessary to determine the flow of data both within single programs and among programs. The report describes a simple technique, data dependency analysis, for automatically generating this information from the static source representation of programs. The report also describes an experimental implementation used to determine the data flow of PL/1 programs taken from the Multics operating system.

[Bisb1978] Richard Bisbey and Dennis Hollingworth. Protection Analysis: Final Report. Research ISI/SR-78-13, University of Southern California Information Sciences Institute, 4676 Admiralty Way / Marina del Rey / California 90291, May 1978. ARPA Order No. 2223.

The Protection Analysis project was initiated at ISI by ARPA IPTO to further understand operating system security vulnerabilities and, where possible, identify automatable techniques for detecting such vulnerabilities in existing system software. The primary goal of the project was to make protection evaluation both more effective and more economical by decomposing it into more manageable and methodical subtasks so as to drastically reduce the requirement for protection expertise and make it as independent as possible of the skills and motivation of the actual individuals involved. The project focused on near-term solutions to the problem of improving th security of existing and future operating systems in an attempt to have some impact on the security of the systems which would be in use over the next ten years.

A general strategy was identified, referred to as "pattern-direct protection evaluation" and tailored to the problem of evaluating existing systems. The approach provided a basis for categorizing protection errors according to their security-relevant properties; it was successfully applied for one such category to the MULTICS operating system, resulting in the detection of previously unknown security vulnerabilities.

[Bish1995] Matt Bishop. A Taxonomy of UNIX System and Network Vulnerabilities. Technical Report CSE-95-10, The University of California, Davis, May 1995.

...In this paper, we shall build on prior work to present another taxonomy, and argue that this classification scheme highlights characteristics of the vulnerabilities it classifies in a more useful way than other work. We shall then examine vulnerabilities in the UNIX operating system, its system and ancillary software, and classify the security-related problems along several axes, after which we shall examine the earlier work to see if this taxonomy holds for other systems. The unique contribution of this work is an analysis of how to use the Protection Analysis work to improve security of existing systems, and how to write programs with minimal exploitable security flaws....

[Bish1996a] Matt Bishop and David Bailey. A Critical Analysis of Vulnerability Taxonomies. Technical Report CSE-96-11, The University of California, Davis, September 1996.

...In the 1970s, two major studies attempted to taxonomize security flaws. One, the RISOS study, focused on flaws in operating systems; the other, the Program Analysis (PA) study, included both operating systems and programs. Interesting enough, the taxonomies both presented were similar, in that the classes of flaws could be mapped to one another. Since then, other studies have based their taxonomies upon these results. However, the classifications defined in these studies are not taxonomies in the sense that we have used the word, for they fail to define classification schemes that identify a unique category for each vulnerability.

Aslam's recent study approached classification slightly differently, through software fault analysis. A decision procedure determines into which class a software fault is placed. Even so, it suffers from flaws similar to those of the PA and RISOS studies.

The next section contains a precise definition of *taxonomy*, as well as a review of the PA, RISOS, and Aslam classification schema. The third section shows that two security flaws may be taxonomized in multiple ways under all of these schemes. The paper concludes with some observations on taxonomies and some ideas on how to develop a more precise taxonomy.

[Bish1996b] Matt Bishop and Michael Dilger. Checking for Race Conditions in File Accesses. *Computing Systems*, 9(2):131–152, Spring 1996.

Flaws due to race conditions in which the binding of a name to an object changes between repeated references occur in many programs. We examine one type of this flaw in the UNIX operating system, and describe a semantic method for detecting possible instances of this problem. We present the results of one such analysis in which a previously undiscovered race condition flaw was found.

[Bish1996c] Matt Bishop. Classifiying Vulnerabilities. NISSC Panel on Vulnerabilities Data: The UC Davis Vulnerabilities Project, October 23 1996.

This is a presentation given at the NISSC Panel about developing a VCS (**V**ulnerabilities **C**lassification **S**cheme. He argues the need for an agreed-upon vocabulary and some method of organizing data. Classification scheme must be flexible, extensible, and useful. He defines *authorized* and *unauthorized* states, a *vulnerable state*, a *compromised state*, an *attack*, and a *vulnerability*. His approach is to decompose vulnerabilities into characteristics at any level of abstraction. He notes differences in design-oriented characteristics and implementation-oriented characteristics. He wonders what point of view is this looked at; that is, from the process(es) being attacked, the process(es) doing the attacking, the operating system, and others. His answer is to do a thesaurus of vulnerability terms.

[Bish1999] Matt Bishop. How Attackers Break Programs, and How To Write Programs More Securely. 8th USENIX Security Symposium, Technical Tutorial Session T1, University of California, Davis, August 24, 1999. bishop@cs.ucdavis.edu.

The goals of this talk is to show how attackers look at programs for potential vulnerabilities. It also aims to show how to write programs which are: to be run by *root* (or some other user), are setuid or setgid, or can't be tricked into doing what they are not intended to do. Topics include: environment, buffer overflow, numeric overflow, race conditions, and network programs.

[Blak1996] Bob Blakley. The Emperor's Old Armor. In *ACM New Security Paradigm Workshop*, pages 2–16, September 1996.

The traditional model of computer security was formulated in the 1970's, when computers were expensive, solitary, heavy, and rare. It rests on three fundamental foundations: management of security *policy* describing the set of actions each user is entitled to perform, *integrity* of the physical system, its software, and especially its security-enforcing mechanisms, and *secrecy* of cryptographic keys and sensitive data.

The modern computing environment, with its rapidly accelerating complexity, connectivity, and miniaturization, is undermining all three of these foundations. Nevertheless, the newest "secure" computer systems continue to be built on them. This paper argues that the traditional model of computer security is no longer viable, and that new definitions of the security problem are needed before the industry can begin to work toward effective security in the new environment.

[Bloo1990] Buck Bloombecker. *Spectacular Computer Crimes: What they are and how they cost American business half a billion dollars a year.* Dow Jones-Irwin, 1990.

This book is an attempt to bring focus to my 10 years work at the National Center for Computer Crime Data, collecting information about computer crime — both the spectacular and the relatively ordinary. Since the Center is a clearinghouse for such information, I have been able to draw on case studies as well as conversations with criminals, victims, and security professionals. I have also drawn on more than a few frustrations experienced in taking this organization from its birth to its current stage of adolescent struggle for identity.

Computer crime is changing. To protect our computers — and ourselves — we need to replace yesterday's myths with today's realities. The worst myth, the basis of most others, is that computer crime is a technologists' problem. In reality, computer crime victimizes us all. The first five chapters of this book offer the National Center's perspective on the realities of computer crime, as well as offering some insights into the history of the leading myths and the mischief associated with them....

Looking to the future, the final section of the book considers growing use of computers in crimes related to politics, and the struggle over software rights. Chapter 18 focuses on the "Internet Worm," which interfered with thousands of computers. This chapter argues the need for a community of computer users who take responsibility for making their use secure....

[Bori2001] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting Mobile Communications: The Insecurity of 802.11. DRAFT; found at www.isaac.cs.berkeley.edu/isaac/wep-draft.[ps/pdf], February 2001.

The 802.11 standard for wireless networks includes a Wired Equivalent Privacy (WEP) protocol, used to protect link-layer communications from eavesdropping and other attacks. We have discovered several serious security flaws in the protocol, stemming from misapplication of cryptographic primitives. The flaws lead to a number of practical attacks that demonstrate that WEP fails to achieve its security goals. In this paper, we discuss in detail each of the flaws, the underlying security principle violations, and the ensuing attacks.

[Borm1999] D. Borman, S. Deering, and R. Hinden. IPv6 Jumbograms. RFC 2675, August 1999. Internet Engineering Task Force (IETF) Request for Comments; http://www.ietf.org.

A "jumbogram" is an IPv6 packet containing a payload longer than 65,535 octets. This document describes the IPv6 Jumbo Payload option, which provides the means of specifying such large payload lengths. It also describes the changes needed to TCP and UDP to make use of jumbograms.

Jumbograms are relevant only to IPv6 nodes that may be attached to links with a link MTU greater than 65,575 octets, and need not be implemented or understood by IPv6 nodes that do not support attachment to links with such large MTUs.

[Boyl1999] James M. Boyle, R. Daniel Resler, and Victor L. Winter. Do You Trust Your Compiler? *Computer*, 32(5):65–73, May 1999.

Correctness-preserving transformations can guarantee that a program continues to do what it should when it is converted from specification to assembly code. Constructing a trusted compiler is one of many potential applications.

[Brad1998] Kirk A. Bradley, Steven Cheung, Nicholas Puketza, Biswanath Mukherjee, and Ronald A. Olsson. Detecting Disruptive Routers: A Distributed Network Monitoring Approach. *IEEE Network*, 12(5):115–124, September/October 1998.

An attractive target for a computer system attacker is the router. An attacker in control of a router can disrupt communication by dropping or misrouting packets passing through the router. We present a protocol called WATCHERS which detects and reacts to routers that drop or misroute packets. WATCHERS is based on the principle of conservation of flow in a network: all data bytes sent into a node, and not destined for that node, are expected to exit the node. WATCHERS tracks this flow, and detects routers that violate the conservation principle. We show that WATCHERS has several advantages over existing network monitoring techniques. We discuss WATCHERS response to several different types of bad router behavior. We demonstrate that in ideal conditions WATCHERS makes no false positive diagnoses, and we describe how WATCHERS can be tuned to perform nearly as well in realistic conditions. Also, we argue that WATCHERS impact on router performance and WATCHERS memory requirements are reasonable for many environments.

[Bran1973] Dennis K. Branstan. Privacy and Protection in Operating Systems. *Computer*, pages 43–46, January 1973. In [Salt1975], the article is referenced as Volume 6 (II-E).

A Workshop Report by Dr.,Dennis K.,Branstan, National Security Agency

The IEEE Committee on Operating Systems sponsored a workshop on privacy and protection in operating system in Princeton, New Jersey, from June 12–14, 1972. Thirty-two people interested in operating system protection met at the Nassau Inn to discuss various problems and their possible solutions. The workshop was organized by Dr.,R.,Stockton Gaines of the Institute for Defense Analysis, Princeton. He and Professor Peter Denning, Princeton University, acted as session chairmen.

The sessions were held informally with topics being generated by the participants. These included designing a secure operating systems on present hardware, designing new hardware protection facilities, weaknesses of current systems' protection features, and methods of continually monitoring a secure system. Informal presentations were given by various people on their own work with questions and related discussion added by the rest of the participants. This paper summarizes the information presented.

[Brin1995] Donald L. Brinkley and Roger R. Schell. What is There to Worry About? An Introduction to the Computer Security Problem. In Marshall D. Abrams, Sushil Jajodia, and Harold J. Podell, editors, *Information Security: An Integrated Collection of Essays*, chapter 1, pages 11–39. IEEE Computer Society Press, 10662 Los Vaqueros Circle P.O. Box 3014 Los Alamitos, CA 90720-1264, 1995.

This essay provides an overview of the vulnerabilities and threats to information security in computer systems. It begins with a historical presentation of past experiences with vulnerabilities in communication security along with present and future computer security experiences. The historical perspective demonstrates that misplaced confidence in the security of a system is worse than having no confidence at all in its security.

Next, the essay describes four broad areas of computer misuse: (1) theft of computational resource, (2) disruption of computational services, (3) unauthorized disclosure of information in a computer, and (4) unauthorized modification of information in a computer. Classes of techniques whereby computer misuse results in the unauthorized disclosure and modification of information are then described and examples are provided. These classes are (1) human error, (2) user abuse of authority, (3) direct probing, (4) probing with malicious software, (5) direct penetration, and (6) subversion of security mechanism. The roles of Trojan horses, viruses, worms, bombs, and other kinds of malicious software are described and examples provided.

[Burs1994] Harvey Burstein. *Introduction to Security*. Prentice Hall, Englewood Cliffs, NJ 07632, 1994.

This book was prompted by the fact that far too many criminal justice and business administration students in particular, and business and police executives, as well as some academians in general, do not seem to appreciate fully what security is. Some tend to think of it as a component of policing; others as little more than locks on doors, alarms, and guards monitoring access or shipping/receiving dock activities, or making periodic patrol rounds. In truth, security it a great deal more....

This book is intended to serve a dual purpose. First, it is hoped that it will provide its readers with a much better understanding of how security must be integrated and of the many types of organization and activities that need effective loss prevention programs. Second, it attempts to give its readers a look at the variety of career opportunities available to students who seek careers that offer a challenge, mental stimulation, a chance to learn much about the operations of businesses and institutions, and no less important, the satisfaction of seeing their ideas for improved security implemented and the hoped for results achieved....

[Calo2000a] Michael A. Caloyannides. Encryption Wars: Early Battles. *IEEE Spectrum*, 37(4):37–43, April 2000.

With the rise of hard-to-crack encryption, sensitive data is easier to protect — and criminal activity tougher to monitor, says part 1 of this two-part article.

The rise of the so-called information economy, borne along by proliferating computers, sprawling telecommunications, and the Internet, has radically transformed how people do business, govern, entertain themselves, and converse with friends and family. Private documents that in the past would have been committed to paper and hand-delivered or stowed under lock and key are now routinely created, sent, and stored electronically.

But the very things that allow such speed and ease of communication have also made it far more difficult to ensure one's privacy. In an electronic age, an interloper can intercept and alter messages far more easily now than when face-to-face exchanges were the norm.

[Calo2000b] Michael A. Caloyannides. Encryption Wars: Shifting Tactics. *IEEE Spectrum*, 37(5):46–51, May 2000.

Law enforcement's new weapons for electronic detection spur privacy proponents to strike back, says part 2 of this two-part article.

The growing availability of powerful encryption has in effect rewritten the rule book for creating, storing, and transmitting computer data. People everywhere rightly regard confidentiality as essential for conducting business and protecting personal privacy. But governments worldwide have been sent into a spin, for fear secret encryption keys will add to the weapons of terrorists and other criminals. Some nations have even attempted to control the technology by constructing a maze of regulations and laws aimed at blocking its import, export, and/or use. Such bans have largely failed, though. [See Part 1 of this article, *IEEE Spectrum*, April, pp.,39–43].[10]

[Carl1975] Jim Carlstedt, Richard Bisbey II, and Gerald Popek. Pattern-Directed Protection Evaluation. Research report ISI/RR-75-31, USC Information Sciences Institute, 4676 Admiralty Way / Marina del Rey, CA 90291, June 1975. ARPA Order #2223 / Program Code 3D30 & 3P10 / Contract DAHC 15 72 C 0308 / Defense Advanced Research Project Agency / 1400 Wilson Blvd. / Arlington, VA 22209.

Because of the urgent security requirements in many existing general-purpose operating systems, the large investment committed to such systems, and the large number of protection errors embedded in them, the problem of finding such errors is one of major importance. This report presents an approach to this task, based on the premise that the effectiveness of error searches can be greatly increased by techniques that utilize "patterns", i.e., formalized descriptions of error types. It gives a conceptual overview of the pattern-directed evaluation process and reports the authors' initial experience in formulating patterns from the analysis of protection errors previously detected in various systems, as well as in applying the pattern-directed technique.

[Carl1976] Jim Carlstedt. Protection Errors in Operating Systems: Validation of Critical Conditions. Technical Report ARPA Order No. 2223; ISI/SR-76-5, Informatoin Sciences Institute, University of Southern California, 4676 Admiralty Way / Marina del Rey / Calfiornia 90291, May 1976. (213) 822-1511.

This report describes a class of operating system protection errors known as insufficient validation of critical conditions, or simply "validation errors," and outlines a scheme for finding them. This class of errors is recognized as a very broad one, lying outside the scope of the basic protection mechanisms of existing systems; the extent of the problem is illustrated by a set of validation errors taken from current systems. Considerations for validity conditions and their attachment to variables and to various types of control points in procedures are explored, and categories of validation methods noted. The notion of criticality itself is analyzed, and criteria suggested for determining which variables and control points are most critical in the protection sense. Because a search for validation errors can involve substantial information processing, the report references existing or developing tools and techniques applicable to this task.

[Carl1978a] Jim Carlstedt. Protection Errors in Operating Systems: A Selected Annotated Bibliography and Index to Terminology. Technical Report ARPA Order No. 2223; ISI/SR-78-10, Information Sciences Institute / University of Southern California, 4676 Admiralty Way / Marina del Rey / California 90291, March 1978. (213) 822-1511.

[10] [Calo2000a].

This report represents the current state of a bibliography on the subject of protection in computer operating system. Current state means that the bibliography is incomplete; it is a byproduct of a research project in the field of protection, recently completed. The bibliography is being published in the belief that it may be useful as is, and that it might serve as the basis of a continuing effort to collect, annotate and index the more significant documents (reports, papers, articles, books, etc.) in the field. Ideally (especially in these days of computerized information bases and communication networks) workers in a research field will collaborate in developing and sharing their bibliographies — not only with simple annotations like this one but with ore extensive comments and reviews. Perhaps this document can be a contribution in that direction and will stimulate owners of other "working" bibliographies to publish theirs. As noted below, this bibliography is online and may be accessed via the ARPANET[11].

[Carl1978b] Jim Carlstedt. Protection Errors in Operating Systems: Serialization. Technical Report ARPA Order No. 2223; ISI/SR-78-9, Information Sciences Institute / University of Southern California, 4676 Admiralty Way / Marina del Rey / California 90291, April 1978. (213) 822-1511.

This document describes a class of protection errors found in current computer operating systems. It is intended (1) for persons responsible for improving the security aspects of existing operating system software and (2) for designers and students of operating systems. The purpose is to help protection evaluators find such errors in current systems and to help designers and implementers avoid them in future systems, by analysis and methodical approach.

This report deals with a class of errors, initially identified empirically, that formed itself around a group of protection errors (within a larger collection) having the common characteristic of involving operations or accesses occurring in the wrong order or at the wrong times; hence the name "serialization". In its broadest sense, it includes a large proportion of all programming errors which may have improper order or scheduling, and, in a narrower sense includes only those errors resulting form improper ordering of accesses to objects accessible by potentially concurrent operations.

This study is neither a full analysis of the subject of the ordering of operations nor only a discussion of process synchronization, but rather an attempt to give perspective to several closely-related subclasses of problems in this area.

[CCEB1994] CCEB. Common Criteria for Information Technology Security Evaluations. Common Criteria Editorial Board, April 1994. Version 0.6. Reference found in [Pfle1997]. Annotation quoted from [Pfle1997].

The Common Criteria approach closely resembles the U.S. *Federal Criteria* (which, of course, was heavily influenced by the ITSEC and Canadian efforts). It preserves the concepts of security targets and protections profiles.... The *Common Criteria* defined topics of interest of security [such as]... **Functionality** (Identification and authentication, Trusted path, Security audit, Invocation of security functions, User data protection, Resource utilization, Protection of the trusted security functions, Privacy, and Communication) and **Assurance** (Development, Testing, Vulnerability assessment, Configuration management, Life-cycle support, Guidance documents, and Delivery and operation).

Under each of these classes, they defined families of functions or assurance needs, and from the families they defined individual components....

Individual components were then combined into packages of components that met some comprehensive requirement (for functionality) or some level of trust (for assurance)....

Finally, the packages were combined into requirements sets or assertions for specific applications or products....

[11]Handwritten note at end of abstract: "(Defense Advanced Research Projects Agency, network of interconnected scientific computers)."

[Chan2001] Naveen Chandran and Matthew C. Valenti. Three Generations of Cellular Wireless Systems. *IEEE Potentials*, 20(1):32–35, February/March 2001.

Over the past decade, wireless technology has undergone enormous growth. Surveys have shown that a new wireless subscriber signs up every 2.5 seconds. The number of cellular and personal communication system (PCS) users in the US has surpassed 100 million. However, wireless is not a recent technology. As early as 1793, wireless messages were transmitted in France using the optical telegraph. Stations consisting of a telescope and a set of semaphore flags capable of encoding multiple messages were placed on adjacent hills. France was entirely linked by 566 such stations. A message could be sent from Paris to the border in about an hour....

[Chap1995] D. Brent Chapman and Elizabeth D. Zwicky. *Building Internet Firewalls*. O'Reilly & Associates, Inc., 1995.

Everyone's jumping on the Internet bandwagon today, but with the explosive growth of the Internet has come a corresponding explosion in attacks on connected computer systems. These range from familiar attacks (e.g., cracking passwords and exploiting security holes in operating systems) to newer and more technically sophisticated ones (e.g., forging IP source addresses, packet sniffing, and hijacking terminal or login sessions). How can you protect your site from these threats? How can you help your users get what they need from the World Wide Web and other Internet services, while protecting your systems and networks from compromise? Internet firewalls are currently the most effective defense.

Building Internet Firewalls is a practical guide to designing, building, and maintaining firewalls. It isn't a theoretical tome on security concepts; it's a down-to-earth, highly detailed handbook for real-life system administrators and managers — and for anyone who wants to learn what firewalls can (and cannot) do to make a site secure. If you're planning to build your own firewall, this book will tell you how to do it. If you're planning to buy one, this book will give you the background information you need to understand the protocols, technologies, and features of the products you'll be considering. It contains:

1. Detailed descriptions of how to build packet filtering and proxying firewalls, and how to configure Internet services (e.g., electronic mail, FTP, DNS, Telnet, WWW, and many more) to work with firewalls.
2. Chapters on overall Internet threats, firewall architectures, security policies and strategies, types of user authentication, firewall maintenance, and how to respond to break-ins.
3. Summaries of information resources and publicly available tools to help you build an effective and affordable firewall.

[Ches1992] Bill Cheswick. An Evening with Berferd in which a Cracker is Lured, Endured, and Studied. In *Proceedings of the Winter 1992 USENIX Conference*, pages 163–173, San Francisco, CA, January 20–24 1992. USENIX.

On 7 January 1991 a cracker, believing he had discovered the famous sendmail DEBUG hole in our Internet gateway machine, attempted to obtain a copy of our password file. I sent him one.

For several months we led this cracker on a merry chase in order to trace his location and learn his techniques. This paper is a chronicle of the cracker's "successes" and disappointments, the bait and traps used to lure and detect him, and the chroot "Jail" we built to watch his activities.

We concluded that our cracker had a lot of time and persistence, and a good list of security holes to use once he obtained a login on a machine. With these hole he could often subvert the *uucp* and *bin* accounts in short order, and then *root*. Our cracker was interested in military targets and new machines to help launder his connections.

This is a draft of a paper accepted for the January 1992 San Francisco Usenix.[12]

[Ches1994] William R. Cheswick and Steven M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley Professional Computing Series, Brian W. Kernighan, Consulting Editor. Addison-Wesley, 1994.

[12] I believe this is *the* copy of the paper, not just a draft.

As a user of the Internet, you are fortunate to be tied into the world's greatest communication and information exchange — but not without a price. As a result of this connection, your computer, your organization's network, and everywhere that network reaches are all vulnerable to potentially disastrous infiltration by hackers.

Written by the AT&T Bell Labs researchers who tracked the infamous "Berferd" hacker and also built the firewall gateway at Bell Labs, Firewalls and Internet Security gives you invaluable advice and practical tools for protecting your organization's computers from the very real threat of a hacker attack through the Internet. You will learn how to plan and execute a security strategy that will thwart the most determined and sophisticated of hackers — while still allowing you easy access to Internet services.

In particular, the authors show you a step-by-step plan for setting up a "firewall" gateway — a dedicated computer equipped with safeguards that acts as a single, more easily defended, Internet connection. They even include a description of their most recent gateway, the tools they used to build it, and the hacker attacks they devised to test it.

You will be fascinated by their first-hand account of one of the first documented hacker attacks, the "Berferd" case, in which Internet hackers created havoc for computer networks worldwide. In addition, you will find vital information on cryptography, a description of the tools used by hackers, and the legal implications of computer security.

With this book in hand, you will be well equipped to provide your organization with effective protection from the wily Internet hacker.

[Chri1999] Jim Christy. Cyber Threat & Legal Issues. Presentation, 26 October 1999. Given at Shadowcon, Dahlgren, VA; Jim Christy (Air Force Office of Special Investigations.

This presentation overviewed the cyber threats to the United States government and the legal issues that must be resolved to counter them. It included the Sandia National Laboratory Taxonomy, which is an extension to John Howard's [Howa1997] Taxonomy.

[Clan1997] Tom Clancy. *Airborne: A Guided Tour of an Airborne Task Force*. Berkley Publishing Group, 200 Maison Avenue, New York, New York 10016, 1997.

They *are* America's front lines — serving proudly in forward areas around the world. Representing the very best from the Army and Air Force, the Airborne Task Force is a formidable combination of manpower and firepower. Now, Tom Clancy examines this elite branch of our nation's armed forces. With pinpoint accuracy and a style more compelling than any fiction, the acclaimed author of *Executive Orders* delivers a fascinating account of the Airborne juggernaut — the people, the technology, and Airborne's mission in an ever-changing world...

[Clar1990] David D. Clark and System Security Committee. Computers at Risk: Safe Computing in the Information Age. Technical report, System Security Study Committee, Computer Science and Telecommunications Board, Commission on Physical Sciences, Mathematics, and Applications, National Research Council, 2101 Constitution Avenue, N.W. Washington, D.C. 20418, December 1990.

The nation is on the threshold of achieving a powerful information infrastructure that promises many benefits. But without adequate safeguards, we risk, intrusions into personal privacy (given the growing electronic storage of personal information) and potential disasters that can cause economic and even human losses. For example, new vulnerabilities are emerging as computers become more common as components of medical and transportation equipment or more interconnected as components of domestic and international financial systems. Many disasters may result from intentional attacks on systems, which can be prevented, detected, or recovered from through better security. *The nation needs computer technology that supports substantially increased*

safety, reliability, and, in particular, security.... The committee urges that its recommendations be considered together as integral to a coherent national effort to encourage the widespread development and deployment of security features in computer systems, increase public awareness of the risks that accompany the benefits of computer systems, and promote responsible use and management of computer systems. Toward the end of increasing the levels of security in new and existing computer and communications systems, the committee developed recommendations in six areas. These are outlined below and developed further in the full report....

1. **Promulgation of a comprehensive set of Generally Accepted System Security Principles, referred to as GSSP, which would provide a clear articulation of essential security features, assurances, and practices....**
2. **A set of short-term actions for system vendors and users that build on readily available capabilities and would yield immediate benefits,...**
3. **Establishment of a system-incident data repository and appropriate education and training programs to promote public awareness.**
4. **Clarification of export control criteria and procedures for secure of trusted systems and review for possible relaxation of controls on the export of implementations of the Data Encryption Standard (DES).**
5. **Funding and directions for a comprehensive program of research.**
6. **Establishment of a new organization to nurture the development, commercialization, and proper use of trust technology, referred to as the Information Security Foundation, or ISF....**

With this report, the committee underscores the need to launch now a process that will unfold over a period of years, and that, by limiting the incidence and impact of disruptions, will help society to make the most of computer and communications systems.

[Clau1993] Carl Von Clausewitz. *On War*. Everyman's Library. Alfred A. Knopf, Inc., New York, 1993. Originally written in four notes from 1816–1830.

What is War? I propose to consider first the various *elements* of the subject, next its *various parts* or *sections*, and finally *the whole* in its internal structure. In other words, I shall proceed from the simple to the complex. But in war more than in any other subject we must begin by looking at the nature of the whole; for here more than elsewhere the part and the whole must always be thought of together....

War is thus an act of force to compel our enemy to do our will....

[Cohe1987] Fred Cohen. Computer Viruses: Theory and Experiments. *Computers & Security*, 6:22–35, 1987.

This paper introduces "computer viruses" and examines their potential for causing widespread damage to computer systems. Basic theoretical results are presented, and the infeasibility of viral defense in large classes of systems is shown. Defensive schemes are presented and several experiments are described.

[Cohe1995] Fred Cohen. Internet Holes: 50 Ways to Attack Your Web Systems. World Wide Web (WWW), 1995. Unknown address. Refer to www.all.net.

The Internet is now the world's most popular network and it is full of potential vulnerabilities. In this series of articles, we explore the vulnerabilities of the Internet and what you can do to mitigate them....

These example attacks come in three types. Attacks marked with a $*$ have been demonstrated. Attacks marked with a $*+$ have caused real-world incidents. Unmarked attacks are theoretical but are very likely to work. Since the goal is 50 attacks and some of the theoretical attacks may not be active today, we have provided 60 attacks under the assumption that this redundancy will cover any attacks that are never demonstrated....

[Cohe1996] Fred Cohen. Internet Holes — Eliminating IP Address Forgery. http://www.all.net/journal/netsec/9606.html, 1996. Copyright (C) 1996, Management Analytics.

The Internet is now the world's most popular network and it is full of potential vulnerabilities. In this series of articles, we explore the vulnerabilities of the Internet and what you can do to mitigate them....

At its root, IP address forgery is a method of deception, and thus it can be used in much the same way as other forms of deception [Dunn1995];. More specifically, and using Dunnigan and Nofi's classification scheme, here are some quick ideas about how IP address forgery might be used:....

[Cohe1997a] Fred Cohen. Information System Attacks: A Preliminary Classification Scheme. *Computers & Security*, 16(1):29–46, 1997.

This paper describes almost a hundred different classes of attack methods gathered from many different sources. Where a single source for a single item is unavailable, it is cited in the text. The most comprehensive sources are not cited throughout the text but rather listed here (Cohen, 1995 and Neumann, 1995).[13][14] Other major sources not identified by specific citation are listed here....

[Cohe1997b] Fred Cohen. Information System Defences: A Preliminary Classification Scheme. *Computers & Security*, 16(2):94–114, 1997.

This paper describes 140 different classes of protective methods gathered from many different sources. Where a single source for a single item is available, it is cited in the text. The most comprehensive sources are not cited throughout the text but rather listed here (Cohen, 1995 and Neumann, 1995).[15][16] Other major sources not identified by specific citation are listed here....

[Cont1998] Alex Conta and Stephen Deering. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. Request for Comments (RFC) 1885, December 1998. Internet Engineering Task Force (IETF); http://www.ietf.org.

This document specifies a set of Internet Control Message Protocol (ICMP) messages for use with version 6 of the Internet Protocol (IPv6).

[Corn1987] Hugo Cornwall. *Datatheft: Computer Fraud, Industrial Espionage and Information Crime*. Ponting-Green, London, 1987.

This book, therefore, is not going to be a mere collection of dreadful tales of computer crime. I want, among other things, to place datacrime in the unambiguous historical and social context of change within corporations and businesses because it is only in this way that the subject can be rescued from the Scylla of sensationalism and the Charybdis of management ignorance. In fact, we will be concerned with one of the least anticipated by-products of the the ongoing information revolution.

The simple equation — more computers means more computer crime — appears to summarise all that can usefully be offered to explain the modern phenomenon of datacrime: information crimes of all types and in particular: datafraud — computer assisted fraud; dataspying — theft of computer-based information; datatheft — theft of the computer's coporeal and calculating resources; and physical attack on computer facilities. I propose to demonstrate that datacrime is now an all prevalent hazard for every organisation that makes use

[13] Cohen, F., 1995. *Protection and Security on the Information Superhighway*, Wiley and Sons, New York, 1995.

[14] [Neum1995], [Neum1989].

[15] Cohen, F., 1995. *Protection and Security on the Information Superhighway*, Wiley and Sons, New York, 1995.

[16] [Neum1995], [Neum1989].

of computers. I will be showing how nearly all present accounts of 'computer crime', however you define and measure the term, are guilty of considerable understatement in terms of the risks that commercial undertakings, public authorities and private individuals face. These crimes are now as commonplace and banal as house-breaking is in the suburbs, pilfering is in retail outlets, warehouses and workshops, and petty embezzlement is in offices; the inevitable press attention on exotic manifestations of computer crime has, alas, encouraged manages to protect their information systems against unusual risks whilst leaving themselves open to the everyday predators they should really fear. The problem of information crime — and its solution — is not in essence 'high-tech', but one of understanding, acceptance of managerial responsibility and the exercise of appropriate administrative controls. The 'answer' to datacrime lies only partly in the introduction of ever more sophisticated computer security facilities.

[Cors1999] S. Corson and J. Macker. Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. Request for Comments (RFC) 2501, January 1999. Network Working Group.

This memo first describes the characteristics of Mobile Ad hoc Networks (MANETs), and their idiosyncrasies with respect to traditional, hardwired packet networks. It then discusses the effect these differences have on the design and evaluation of network control protocols with an emphasis on routing performance evaluation considerations.

[Cowa1998] Crispin Cowan, Calton Pu, Dave Maier, Heather Hinton, Jonathan Walpole, Peat Bakke, Steve Beattie, Aaron Grier, Perry Wagle, and Qian Zhang. StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks. In *Seventh USENIX Security Symposium*, pages 63–77, January 26–29 1998.

This paper presents a systematic solution to the persistent problem of buffer overflow attacks. Buffer overflow attacks gained notoriety in 1988 as part of the Morris Worm incident on the Internet. While it is fairly simple to fix individual buffer overflow vulnerabilities, buffer overflow attacks continue to this day. Hundreds of attacks have been discovered, and while most of the obvious vulnerabilities have now been patched, more sophisticated buffer overflow attacks continue to emerge.

We describe StackGuard: a simple compiler technique that virtually eliminates buffer overflow vulnerabilities with only modest performance penalties. Privileged programs that are recompiled with the StackGuard compiler extension no longer yield control to the attacker, but rather enter a fail-safe state. These programs require *no* source code changes at all, and are binary-compatible with existing operating systems and libraries. We describe the compiler technique (a simple patch to **gcc**), as well as a set of variations on the technique that tradeoff between penetration resistance and performance. We present experimental results of both the penetration resistance and the performance impact of this technique.

[Cowa2000] Crispin Cowan, Perry Wagle, Calton Pu, Steve Beattie, and Jonathan Walpole. Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade. *Proceedings of the DARPA Information Survivability Conference & Exposition*, II:119–129, January 25–27 2000. Published by the IEEE Computer Society; www.cse.orgi.edu/DISC/projects/immunix.

Buffer overflows have been the most common form of security vulnerability for the last ten years. More over, buffer overflow vulnerabilities dominate the area of remote network penetration vulnerabilities, where an anonymous Internet user seeks to gain partial or total control of a host. If buffer overflow vulnerabilities could be

effectively eliminated, a very large portion of the most serious security threats would also be eliminated. In this paper, we survey the various types of buffer overflow vulnerabilities and attacks, and survey the various defensive measures that mitigate buffer overflow vulnerabilities, including our own StackGuard method. We then consider which combinations of techniques can eliminate the problem of buffer overflow vulnerabilities, while preserving the functionality and performance of existing systems.

[Croc1989] Stephen D. Crocker and Mary M. Bernstein. ARPANET Disruptions: Insight into Future Catastrophes. Technical Report TIS Report #247, Trusted Information Systems, Inc., 11340 Olympic Blvd., Suite 265, Los Angeles, CA 90064, August 24 1989.

The Strategic Defense System (SDS) will be a large, complex, loosely-coupled realtime systems built for the purpose of defending the United States against a missile attack. The SDS assets (weapons and sensors) will be controlled by the Battle Management System (BMS), which includes Command, Control and Communications (C^3). The BMS coordinates strategic and tactical military forces as well as intelligence operations under control of the Strategic Defense System Operational Commander at the SDS Command Center. Battle management functions will be co-located with the resources they are controlling in a distributed, global network.

The SDS communications networks must support continuous command and control. The networks must pass sensor data and status to the battle managers, and guidance commands to the inflight interceptors. The network must operate with high availability, integrity, and confidentiality to ensure reliable and timely delivery of critical data.

Opening statement in *SDS Battle Management Security Study, 28 Month Status Report*
ARPANET Disruptions: Insight Into Future Catastrophes

This report is part of a three year SDS BMS security study, funded by RADC. The overall study is examining the security architecture, the accreditation methodology, the design methodology, and the software development methodology for a secure BMS....

[Curr1992] David A. Curry. *UNIX System Security.* Addison Wesley, One Jacob Way Reading, MA 01867, 1992.

Because the UNIX system was originally designed by programmers for use by other programmers, it was used in an environment of open cooperation where security was of minimal concern. Now that its use has spread to universities, businesses, and government, the confidential and sensitive nature of the data stored on UNIX systems has made the security of these systems of paramount importance.

Despite all the technical papers and workshops on UNIX security, this book is unique. *UNIX System Security* is the first up-to-date source to provide the UNIX system user or administrator with the information needed to protect the data and system from unauthorized use. By following the procedures described in this book and making use of the C programs and shell scripts provided as examples, you can protect your UNIX system from most attackers.

The author begins by examining four high-profile breaches of UNIX security as illustrations of how a UNIX system can be attacked. He then provides the information necessary to protect against these forms of attack, and offers the tools that can be used to do so. Focusing on the most recent release of Berkeley and System V UNIX, and such vendor derivatives as SunOS and ULTRIX, the book gives information that can be applied to any version of UNIX since Seventh Edition.

Issues discussed include account and password security, securing the file system, encryption and authentication systems, TCP/IP network security, the Network Information Service (NIS), NFS, RFS, workstation security, terminals and modems, and UUCP. Other chapters describe how to respond if your system is attacked and how to develop a comprehensive security policy for your organization. The book also gives comprehensive lists of freely available security software, and publications and mailing lists dealing with UNIX security.

[Dam1996] Kenneth W. Dam and Committee to Study National Cryptography Policy. Cryptography's Role In Securing the Information Society. Technical report, Committee to Study National Cryptography Policy, Computer Science and Telecommunications Board, Commission on Physical Sciences, Mathematics, and Applications, National Research Council, 2101 Constitution Avenue, NW Washington, DC 20418, 1996.

This report undertook the following tasks: *Framing the problem.... Understanding the underlying technology issues and their expected development and impact on policy over time.... Describing current cryptography policy.... Articulating a framework for thinking about cryptography policy.... Identifying a range of feasible policy options. Making recommendations regarding cryptography policy....* The committee concludes that **the debate over national cryptography policy can be carried out in a reasonable manner on an unclassified basis....** The committee believes that **U.S. national policy should be changed to support the broad use of cryptography in ways that take into account competing U.S. needs and desires for individual privacy, international economic competitiveness, law enforcement, national security, and world leadership....** The committee found that **current national cryptography policy is not adequate to support the information security requirements of an information society....**

[Daye1997] Rifaat A. Dayem. *Mobile Data & Wireless LAN Technologies.* Prentice Hall Series in Computer Networking and Distributed Systems. Prentice Hall, Prentice Hall PTR / Upper Salle River, NJ 07458, 1997. Radia Perlman, Series Advisor.

The complete, state fo the art guide to wireless data for engineers, networking professionals and managers.

Wireless data networking will be the next revolution in communications. **Mobile Data and Wireless LAN Technologies** is the most complete, independent, authoritative guide to wireless data: where it stands now, and what to expect tomorrow.

Based on Rifaat Dayem's internationally-recognized Networld+Interop tutorials, it presents detailed technical and business information for every leading and emerging wireless LAN and WAN technology, including spread spectrum, packet radio, infrared, and data-over-circuit-switched solutions. Both U.S. and global markets are covered.

Dayem reviews potential applications, market forecasts, services offered, traffic capacities and bandwidth issues, achievable throughput, spectrum allocation, standards, products, and key players. Technologies covered include:

- Pocket radio
- CDPD
- Two-way paging
- Spread Spectrum (FHSS and DSSS)
- MAC protocols for wireless networks
- Mobile IP
- Data over cellular and PCS
- 802.11
- Hiperlan
- Wireless ATM

This book also includes a primer on wireless networking, mobile data, wireless spectra and international standards.

Demand for wireless data will skyrocket as more robust, mature systems emerge. If you are an engineer or manager developing these systems, or an IS/networking professional evaluating them, *Mobile Data and Wireless LAN Technologies* will be an indispensable sourcebook.

[Deer1998] Stephen E. Deering and Robert M. Hinden. Internet Protocol, Version 6 (IPv6). Request for Comments (RFC) 2460, December 1998. Internet Engineering Task Force (IETF); http://www.ietf.org.

IP version 6 (IPv6) is a new version of the Internet Protocol, designed as the successor to IP version 4 (IPv4) [Post1981b]. The changes from IPv4 to IPv6 fall primarily into the following categories:

- Expanded Addressing Capabilities: IPv6 increases the IP address size from 32 bits to 128 bits, to support more levels of addressing hierarchy, a much greater number of addressable nodes, and simpler auto-configuration of addresses. The scalability of multicast routing is improved by adding a "scope" field to multicast addresses. And a new type of address called an "anycast address" is defined, used to send a packet to any one of a group of nodes.
- Header Format Simplification: Some IPv4 header fields have been dropped or made optional, to reduce the common-case processing cost of packet handling and to limit the bandwidth cost of the IPv6 header.
- Improved Support for Extensions and Options: Changes in the way IP header options are encoded allows for more efficient forwarding, less stringent limits on the length of options, and greater flexibility for introducing new options in the future.
- Flow Labeling Capability: A new capability is added to enable the labeling of packets belonging to particular traffic "flows" for which the sender requests special handling, such as non-default quality of service or "real-time" service.
- Authentication and Privacy Capabilities: Extensions to support authentication, data integrity, and (optional) data confidentiality are specified for IPv6. This document specifies the basic IPv6 header and the initially-defined IPv6 extension headers and options. It also discusses packet size issues, the semantics of flow labels and traffic classes, and the effects of IPv6 on upper-layer protocols. The format and semantics of IPv6 addresses are specified separately in [Hind1998]. The IPv6 version of ICMP, which all IPv6 implementations are required to include, is specified in [Cont1998].

[Denn1986] Dorothy E. Denning. An Intrusion-Detection Model. In *IEEE Symposium on Security and Privacy*, pages 118–131, SRI International, 333 Ravenswood Ave. Menlo Park, CA 94025, 1986. IEEE.

A model of a real-time intrusion-detection expert system capable of detecting break-ins, penetrations, and other forms of computer abuse is described. The model is based on the hypothesis that security violations can be detected by monitoring a system's audit records for abnormal patterns of system usage. The model includes profiles for representing the behavior of subjects with respect to objects in terms of metrics and statistical models, and rules for acquiring knowledge about this behavior from audit records and for detecting anomalous behavior. The model is independent of any particular system, application environment, system vulnerability, or type of intrusion, thereby providing a framework for a general-purpose intrusion-detection expert system.

[Denn1987] Dorothy E. Denning. An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, SE-13(2):222–232, February 1987.

A model of a real-time intrusion-detection expert system capable of detecting break-ins, penetrations, and other forms of computer abuse is described. The model is based on the hypothesis that security violations can be detected by monitoring a system's audit records for abnormal patterns of system usage. The model includes profiles for representing the behavior of subjects with respect to objects in terms of metrics and statistical models, and rules for acquiring knowledge about this behavior from audit records and for detecting anomalous behavior. The model is independent of any particular system, application environment, system vulnerability, or type of intrusion, thereby providing a framework for a general-purpose intrusion-detection expert system.

[Denn1989] Peter J. Denning. The Internet Worm. *American Scientist*, 77:126–128, March–April 1989.

Late in the evening of 2 November 1988, someone released a "worm" program into the ARPAnet. The program expropriated the resources of each invaded computer to generate replicas of itself on other computers, but did no apparent damage. Within hours, it had spread to several thousand computer attached to the world-wide Research Internet.

Computers infested with the worm were soon laboring under a huge load of programs that looked like innocuous "shell" programs (command interpreters). Attempts to kill these programs were ineffective: new copies would appear from Internet connections as fast as old copies were deleted. Many systems had to be shut down and the security loopholes closed before they could be restarted on the network without reinfestation....

[Denn1990a] Dorothy E. Denning. Concerning Hackers Who Break into Computer Systems. In *13th National Computer Security Conference, Information Systems Security: Standards — The Key to the Future*, volume II, pages 653–664, 1990.

A diffuse group of people, often called "hackers," has been characterized as unethical, irresponsible, and a serious danger to society for actions related to breaking into computer system.s This paper attempts to construct a picture of hackers, their concerns, and the discourse in which hacking takes place. My initial findings suggest that hackers are learners and explorers who want to help rather than cause damage, and who often have very high standards of behavior. My findings also suggest that the discourse surrounding hacking belongs at the very least to the gray areas between larger conflicts that we are experiencing at every level of society and business in an information age where many are not computer literate. These conflicts are between the idea that information cannot be owned and the idea that it can, and between law enforcement and the First and Fourth Amendments. Hackers have raised important issues about values and practices in an information society. Based on my findings, I recommend that we work closely with hackers, and suggest several actions that might be taken.

[Denn1990b] Peter J. Denning, editor. *Computers Under Attack: Intruders, Worms, and Viruses*. Addison-Wesley, Reading, Massachusetts, 1990.

This book collects some of the most informative, provocative, and frightening reports on the vulnerability of computer systems to harmful, if not catastrophic, attacks. Whether these attacks are carried out against computer networks or against personal computers — whether driven by mischief or by malice — the consequences can be costly and dangerous. Countermeasures are, in most cases, straightforward.

Peter J. Denning, Editor-in-Chief of *Communications of the ACM*, has been following and writing about computer security for many years. For this book, he has carefully selected a range of articles and commentaries to illuminate recent events and ongoing issues for both nontechnical and technical readers. The book, which is organized into six parts, opens with a discussion of the worldwide networks vulnerable to computer attack. Denning then defines and distinguishes among the various threats lately given widespread attention: intruders, worms, and viruses. The discussion of viruses provides the history and mode of operation of more than 100 viruses, with a guide for virus detection and removal and MS-DOS-based PCs.

Subsequent sections reach into the background, the technical details, and the effects of particular attacks, with reports from Cornell and M.I.T. on the Internet Worm incident, and Clifford Stoll's original account of the Wily Hacker. Part V transports the reader into the cyberpunk subculture, including and interview with the editor of *W.O.R.M.* magazine. A final section explores the sociopolitical, ethical, and legal implications of computer attacks, and introduces the concept of crimoids — media-driven, high-tech crimes — leaving the reader with this ominous suggestion: Future threats may be even greater than intruders, worms, and viruses.

Computers touch everyone in a modern society. The stories recounted in this book are a pointed warning that our computer systems are already under attack, that the privacy and integrity of information in our personal, business, and research activities are seriously threatened, and that the security of free societies is on the line. In order to thwart these attacks, we must all first recognize understand the threat.

[Denn1999] Dorothy Elizabeth Robling Denning. *Information Warfare and Security*. Addison Wesley, 1999.

Every day, we hear reports of hackers who have penetrated computer networks, vandalized Web pages, and accessed sensitive information. We hear how they have tampered with medical records, disrupted emergency

911 systems, and siphoned money from bank accounts. Could information terrorists, using nothing more than a personal computer, cause plane crashes, widespread power blackouts, or financial chaos? Such real and imaginary scenarios, and our defense against them, are the stuff of information warfare — operations that target or exploit information media to win some objective over an adversary.

In **Information Warfare and Security**, Dorothy E. Denning, a pioneer in computer security, provides a framework for understanding and dealing with information-based threats: computer break-ins, fraud, sabotage, espionage, piracy, identity theft, invasions of privacy, and electronic warfare. She describes these attacks with astonishing, real examples, as in her analysis of information warfare operations during the Gulf War. Then, offering sound advice for security practices and policies, she explains countermeasures that are both possible and necessary.

Key features include:

- A comprehensive and coherent treatment of offensive and defensive information warfare, identifying the key actors, targets, methods, technologies, outcomes, policies, and laws;
- A theory of information warfare that explains and integrates within a single framework operations involving diverse actors and media;
- An accurate picture of the threats, illuminated by actual incidents;
- A description of information warfare technology and their limitations, particularly the limitations of defensive technologies.

Whatever your interest or role in the emerging field of information warfare, this book provides the background you need to make informed judgments about potential threats and our defenses against them.

[deVi1998] Marco deVivo, Gabriela O. de Vivo, and Germinal Isern. Internet Securty Attacks at the Basic Levels. *Operating Systems Review*, 32(2):4–15, April 1998. Primary author is de Vivo.[17]

The Internet put the rest of the world at the reach of our computers. In the same way it also made our computers reachable by the rest of the world. Good news and bad news!.(sic) Over the last decade, the Internet has been subject to widespread security attacks. Besides the classical terms, new ones had to be found in order to designate a large collection of threats: *Worms, break-ins, hackers, crackers, hijacking, phrackers, spoofing, man-in-the-middle, password-sniffing, denial-of-service*, and so on.

Since the Internet was born of academic efforts to share information, it never strove for high security measures. In fact in some of its components, security was consciously traded for easiness in sharing. Although the advent of *electronic commerce* has pushed for "real security" in the Internet, there is yet a huge amount of users (including scientists) very vulnerable to attacks, mostly because they are not aware of the nature (and ease) of the attacks and still believe that a "good" password is all they need to be concerned about.

We wrote this paper aiming for a better understanding of the subject. In the paper we report some of the major actual known attacks. Besides the description of each attack (the *what*), we also discuss the way they are carried on (the *how*) and, when possible, the related means of prevention, detection and/or defense.

[deVi1999] Marco deVivo, Gabriela O. de Vivo, Roberto Koeneke, and Germinal Isern. Internet Vulnerabilities Related to TCP/IP and T/TCP. *Computer Communication Review*, 29(1):81–85, January 1999. Primary author is de Vivo.[18]

The Internet put the rest of the world at the reach of our computers. In the same way it also made our computers reachable by the rest of the world. Good news and bad news! Over the last decade, the Internet has been subject to widespread security attacks. Besides the classical terms, new ones had to be found in order to designate a large collection of threats: *Worms, break-ins, hackers, crackers, hijacking, phrackers (sic), spoofing, man-in-the-middle, password-sniffing, denial-of-service*, and so on.

Since the Internet was born of academic efforts to share information, it never strove for high security

[17]Last name is actually de Vivo, but to get it listed correctly in BibTeX it was listed without a space. My apologies to the authors.

[18]Last name is actually de Vivo, but to get it listed correctly in BibTeX it was listed without a space. My apologies to the authors.

measures. In fact in some of its components, security was consciously traded for easiness in sharing. Although the advent of *electronic commerce* has pushed for "real security" in the Internet, there are still a large number of users (including computer scientists) that are very vulnerable to attacks, mostly because they are not aware of the nature (and ease) of the attacks and still believe that a "good" password is all they need to be concerned about.

Aiming for a better understanding of the subject, we wrote a first paper in which we discussed several threats and attacks related to **TCP/IP**. The present work is an extension of the first one, and its main goal is to include **T/TCP** in the discussion. Additionally, in an effort to make this paper more comprehensive, we included some sections from the former.

Besides the description of each attack (the *what*), we also discuss the way they are carried out (the *how*) and, when possible, the related means of prevention, detection and/or defense.

[Dier1999] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246, January 1999. Network Working Group.

This document specifies Version 1.0 of the Transport Layer Security (TLS) protocol. The TLS protocol provides communications privacy over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

[Dild1998] Dildog. The Tao of Windows Buffer Overflow. File 351 of Cult of the Dead Cow; http://www.cultdeadcow.com/cDc_files/cDc-351/ (Written April 16, 1998; Published 1 May), May 1, 1998.

Essence: Throughout these ages / our operating systems / infested by bugs / The ignorant world / turns to Windows for safety / Safety from themselves / It is now the time / for the world to realize / that we all feel pain....

Exploiting the buffer overflow takes patience, knowledge, skill, and imagination. I can not teach you patience, and I can can (sic) not clear your mind. I will however, give you the tools and concepts required to build your own exploits for buffer overflow bugs in the Windows 95, 98 and NT operating systems....

[DoD1985] DoD. *Department of Defense Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD edition, December 1985. Also known as the Orange Book.

The trusted computer system evaluation criteria defined in this document classify systems into four broad hierarchical divisions of enhanced security protection. They provide a basis for the evaluation of effectiveness of security controls built into automatic data processing system products. The criteria were developed with three objectives in mind: (a) to provide users with a yardstick with which to assess the degree of trust that can be placed in computer systems for the secure processing of classified or other sensitive information; (b) to provide guidance to manufacturers as to what to build into their new, widely-available trusted commercial products in order to satisfy trust requirements for sensitive applications; and (c) to provide a basis for specifying security requirements in acquisition specifications. Two types of requirements are delineated for secure processing: (a) specific security feature requirements and (b) assurance requirements. Some of the latter requirements enable evaluation personnel to determine if the required features are present and functioning as intended....

[DoD1994] Defense Science Board of DoD. Report of the Defense Science Board Summer Study Task Force on Information Architecture for the Battlefield. Technical report, Department of Defense, United States of America, Office of the Under Secretary of Defense for Acquisition & Technology, Washington, D.C. 20301-3140, October 1994.

This report is a product of the Defense Science Board (DSB). The DSB is a Federal Advisory Committee established to provide independent advice to the Secretary of Defense. Statements, opinions, conclusions and recommendations in this report do not necessarily represent the official position of the Department of Defense.

This Defense Science Board Summer Study Task Force was charged to make recommendations for implementing an information architecture that would enhance combat operations by providing commanders and forces at all levels with required information displayed for assimilation. The Task Force was instructed to focus on information support to the theater or joint task force commander in preparation for and during combat operations.

The global security environment provided the background for understanding the information needs of warfighting commanders in scenarios likely to occur in the coming decade. Based upon this environment, the Task Force assessed four aspects of information architecture for the battlefield:

- the use of information in warfare;
- the use of information warfare, both offensive and defensive;
- the business practices of the Department of Defense (DoD) in acquiring and using battlefield information systems; and
- the underlying technology required to develop and implement these systems.

This report provides detailed analysis and supporting rationale for the findings and recommendations of the Task Force, which are summarized as follows:

Key Findings:

- The warfighter must be an informed customer, with an integral role in the determination of the operational output (specification of requirements), acquisition, and implementation of information systems;
- Warfighters require flexible information systems that can be readily and rapidly adapted and/or altered to accomplish different missions;
- DoD information systems are highly vulnerable to information warfare, but so are those of potential adversaries; and,
- The DoD can greatly leverage limited DoD resources by exploiting available commercial practices and technology plus "buying into" commercial practices....

[DoD1999] Office of General Counsel DoD. An Assessment of International Legal Issues in Information Operations. Technical report, Department of Defense, May 1999. www.cs.georgetown.edu/denning/infosec/DOD-IO-legal.doc.

International law consists of binding legal obligations among several states. Two of the basic principles of the international legal systems are that sovereign states are legally equal and independent actors in the world community, and that they generally assume legal obligations only by affirmatively agreeing to do so. The most effective instruments in creating international law are international agreements, which may be either bilateral or multilateral....

We can make some educated guesses as to how the international legal system will respond to information operations, but the direction that response actually ends up taking may depend a great deal on the nature of the events that draw the nations' attention to the issue. If information operations techniques are seen as just another new technology that does not greatly threaten the nations' interests, on dramatic legal developments may occur. If they are seen as a revolutionary threat to the security of nations and the welfare of their citizens, it will be much more likely that efforts will be made to restrict or prohibit information operations by legal means. These are considerations that national leaders should understand in making decisions on using information operation techniques in the current formative period, but it should also be understood that the course of future events is often beyond the control of the statesmen....

[Dörn1996] Dietrich Dörner. *The Logic of Failure: Why Things Go Wrong and What We Can Do To Make Them Right.* Metropolitan Books, 115 West 18th Street, New York, New York 10011, 1996. Translated by Rita and Robert Kimber; originally published in Germany in 1989 under the title *D*ie Logik des Misslingenes by Rowhlt Verlag.

The subject of this book is the nature of our thinking when we deal with complex problems. I describe the kinds of mistakes human beings make, the blind alleys they follow down and the detours they take in attempting to cope with such problems. But I am not concerned with thinking along, for thinking is always rooted in the total process of psychic activity. There is no thinking without emotion. We get angry, for example, when we can't solve a problem, and our anger influences our thinking. Thought is embedded in a context of feeling and affect; thought influences, and is in turn influenced by, that context....

Failure does not strike like a bolt from the blue; it develops gradually according to its own logic. As we watch individuals attempt to solve problems, we will see that complicated situations seem to elicit habits of thought that set failure in motion from the beginning. From that point, the continuing complexity of the task and the growing apprehension of failure encourage methods of decision making that make failure even more likely and then inevitable.

We can learn, however. People court failure in predictable ways. Readers of this book will find many examples of confusion, misperception, shortsightedness, and the like; they will also find that the sources of these failing are often quite simple and can be eliminated without adopting a revolutionary new mode of thought. Having identified and understood these tendencies in ourselves, we will be much better problem solvers. We will be more able to start wisely, to make corrections in midcourse, and, most important, to learn from failure we did not avert. We need only apply the ample power of our minds to understanding and then breaking the logic of failure.

[Drak1995] Chris Drake and Kimberley Brown. *Panic! UNIX System Crash Dump Analysis*. SunSoft Press, A Prentice Hall Title, 1995.

UNIX systems crash. It's a fact of life. Until now, little information has been available regarding system crashes. *Panic!* is the first book to concentrate solely on system crashes and hangs, explaining what triggers them and what to do when they occur. *Panic!* guides you through system crash dump postmortem analysis towards problem resolution.

Analysis of system crash dumps usually requires the skills and resources of a UNIX guru, including a wide set of programming skills, an indepth knowledge of UNIX internals, and access to source code. However, by following the advice and "tricks of the trade" presented in *Panic!*, you will be able to establish what killed your system.

Although written for the system administrator, *Panic!* includes information that even the most seasoned UNIX guru will find useful. Topics covered include: What is a panic? What is a hang? Header files, symbols, and symbol tables. A comprehensive tutorial on adb, the absolute debugger. How to read adb macros and how to write your won. Stacks and stack tracebacks. Introduction to assembly language. Overview of UNIX internals. The SPARC processor and its instruction set. Actual case studies of postmortem analysis. A CD-ROM containing several useful analysis tools, such as adb macros and C tags output from the source trees of two different UNIX systems, is included.

Panic! presents this highly technical and intricate subject in a friendly, easy style which even the novice UNIX system administrator will find readable, educational, and enjoyable.

[Drog1999] Bob Drogin. U.S. Scurries to Erect Cyber-Defenses. *Los Angeles Times*, October 31 1999. In Business / Technology Section.

Distant forests dominate the view from the eighth-floor director's suite at the National Security Agency, American's largest intelligence gathering operation. But the talk inside is of a more troubling horizon: cyberspace.

"Think of it as a physical domain, like land, sea and air,"said Air Force Lt. Gen. Michael V. Hayden in his first interview since taking the NSA's helm in May. "Now think of America conducting operations in that new domain."

These days, many in the U.S. intelligence, law enforcement and national security community are thinking of little else....

[Dunn1995] James F. Dunnigan and Albert A. Nofi. *Victory & Deceit: Dirty Tricks at War*. Quill William Morrow, New York, 1995.

The most potent weapon in any soldier's arsenal is deception. That you don't hear much about deception in warfare tells you something about how elusive and apparently rare this item is. Yet, as the ancient Chinese adage puts it, "There can never be enough deception in war." In fact, Sun-tzu, the noted Chinese strategist, went further, saying, "All warfare is based on deception."....

These examples generally use one or more of the traditional deception techniques, which can be summarized briefly as follows:

- Concealment: hiding your forces from the enemy using natural cover, obstacles, or simply great distance....
- Camouflage: hiding your troops and movements from the enemy by artificial means....
- False and Planted Information: letting the enemy get his hands on information that will hurt him and help you, but he won't know that he's being snookered....
- Ruses: tricks, such as displays that use enemy equipment and procedures to deceive....
- Displays: using techniques to make the enemy see what isn't there....
- Demonstrations: making a move with your forces that implies imminent action, but is not followed through....
- Feints: like a demonstration, but you actually make an attack, or retreat....
- Lies: flat-out lying when communicating with the enemy is something that is timeless....
- Insight: the ability of one general to deceive his opponent by outthinking him....

[East1994] D. Eastlake, III, S. Crocker, and J. Schiller. Randomness Recommendation for Security. Request for Comments (RFC) 1750, December 1994. Internet Engineering Task Force (IETF); http://www.ietf.org.

Security systems today are built on increasingly strong cryptographic algorithms that foil pattern analysis attempts. However, the security of these systems is dependent on generating secret quantities for passwords, cryptographic keys, and similar quantities. The use of pseudo-random processes to generate secret quantities can result in pseudo-security. The sophisticated attacker of these security systems may find it easier to reproduce the environment that produced the secret quantities, searching the resulting small set of possibilities, than to locate the quantities in the whole of the number space.

Choosing random quantities to foil a resourceful and motivated adversary is surprisingly difficult. This paper points out many pitfalls in using traditional pseudo-random number generation techniques for choosing such quantities. It recommends the use of truly random hardware techniques and shows that the existing hardware on many systems can be used for this purpose. It provides suggestions to ameliorate the problem when a hardware solution is not available. And it gives examples of how large such quantities need to be for some particular applications.

[EFF1998] Electronic Frontier Foundation EFF. *Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design.* O'Reilly & Associates, May 1998.

Sometimes you have to do good engineering to straighten out twisted politics. The Electronic Frontier Foundation has done so by exploding the government-supported myth that the Data Encryption Standard (DES) has real security.

National Security Agency and FBI officials say our civil liberties must be curtailed because the government can't crack the security of DES to wiretap bad guys. But somehow a tiny nonprofit has designed and built a $200,000 machine that cracks DES in a week. Who's lying, and why?

For the first time, the book reveals full technical details on how researchers and data-recovery engineers can build a working DES Cracker. It includes design specifications and board schematics, as well as full source code for the custom chip, a chip simulator, and the software that drives the system. The US government makes it illegal to publish these details on the Web, but they're printed here in a form that's easy to read and understand, legal to publish, and convenient for scanning into your computer.

The Data Encryption Standard withstood the test of time for twenty years. This book shows exactly how it was brought down. Every cryptographer, security designer, and student of cryptography policy should read this book to understand how the world changed as it fell.

[Elga1997] Taher Elgamal and Kipp E.B. Hickman. Secure Socket Layer Application Program Apparatus and Method. U.S. Patent #5,657,390, August 12 1997.

A computer program product comprising: a computer useable medium having computer readable program code means embodied therein for encrypting and decrypting information transferred over a network between a client application program running in a client computer and a server application program running in a server computer, the computer readable program code means in the computer program product comprising: computer readable program code means for providing a socket application program interface to an application layer program; computer readable program code means for providing encrypted information to transport protocol layer services; computer readable program code means for encrypting information received from an application layer program; and computer readable program code means for decrypting information received from transport protocol layer services.

[Farm1990] Daniel Farmer and Eugene H. Spafford. The COPS Security Checker System. In *USENIX Summer Conference*, pages 165–170. USENIX, June 11–15 1990.

In the past several years, there have been a large number of published works that have graphically described a wide variety of security problems particular to UNIX. Without fail, the same problems have been discussed over and over again, describing the problems with SUID (set user ID) programs, improper file permissions, and bad passwords (to name a few). There are two common characteristics to each of these problems: first, they are usually simple to correct, if found; second, they are fairly easy to detect.

Since almost all UNIX systems have fairly equivalent problems, it seems appropriate to create a tool to detect potential security problems as an aid to system administrators. This paper describes one such tool: COPS. COPS (Computerized Oracle and Password System) is a freely-available, reconfigurable set of programs and shell scripts that enable system administrators to check for possible security holes in their UNIX systems.

This paper briefly describes the COPS system. Included are the underlying design goals, the functions provided by the tool, possible extensions, and some experiences gained from its use. We also include information on how to obtain a copy of the initial COPS release.

[Farm1995] Dan Farmer and Wietse Venema. SATAN: Security Administrator's Tool for Analyzing Networks. http://www.fish.com/satan/, April 1995. Latest version is Satan 1.1.1 (Version 1.0 Released April 5, 1995 at 1400 GMT; Version 1.1 released April 11, 1995; Version 1.1.1 released shortly after Version 1.1).

SATAN was written because we realized that computer systems are becoming more and more dependent on the network, and at the same becoming more and more vulnerable to attack via that same network.

The rationale for SATAN is given in a paper posted in December 1993 admin guide to cracking, a flat text compressed with the UNIX compress command).

SATAN is a tool to help systems administrators. It recognizes several common networking-related security problems, and reports the problems without actually exploiting them.

For each type or problem found, SATAN offers a tutorial that explains the problem and what its impact could be. The tutorial also explains what can be done about the problem: correct an error in a configuration file, install a bugfix from the vendor, use other means to restrict access, or simply disable service.

SATAN collects information that is available to everyone on with access to the network. With a properly-configured firewall in place, that should be near-zero information for outsiders.

We have done some limited research with SATAN. Our finding is that on networks with more than a few dozen systems, SATAN will inevitably find problems. Here's the current problem list:

- NFS file systems exported to arbitrary hosts
- NFS file systems exported to unprivileged programs
- NFS file systems exported via the portmapper
- NIS password file access from arbitrary hosts
- Old (i.e. before 8.6.10) sendmail versions
- REXD access from arbitrary hosts

- X server access control disabled
- arbitrary files accessible via TFTP
- remote shell access from arbitrary hosts
- writable anonymous FTP home directory

[Felt1997] Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach. Web Spoofing: An Internet Con Game. In *20th National Information Systems Security Conference*, pages 95–103, Department of Computer Science, Princeton University, October 7–10 1997. NIST, Ft.,Meade, MD National Computer Security Center.

This paper describes an Internet security attack that could endanger the privacy of World Wide Web users and the integrity of their data. The attack can be carried out on today's systems, endangering users of the most common Web browsers, including Netscape Navigator and Microsoft Internet Explorer.

Web spoofing allows an attacker to create a "shadow copy" of the entire World Wide Web. Accesses to the shadow Web are funneled through the attacker's machine, allowing the attacker to monitor all of the victim's activities including any passwords or account numbers the victim enters. The attacker can also cause false or misleading data to be sent to the Web servers in the victim's name, or to the victim in the name of any Web server. In short, the attacker observes and controls everything the victim does on the Web.

We have implemented a demonstration version of this attack.

[Ferg2000] P. Ferguson. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827, May 2000. Internet Engineering Task Force, Request for Comments. http://www.ietf.org.

Recent occurrences of various Denial of Service (DoS) attacks which have employed forged source addresses have proven to be a troublesome issue for Internet Service Providers and the Internet community overall. This paper discusses a simple, effective, and straightforward method for using ingress traffic filtering to prohibit DoS attacks which use forged IP addresses to be propagated from 'behind' an Internet Service Provider's (ISP) aggregation point.

[Finn1988] Gregory G. Finn. Reducing the Vulnerability of Dynamic Computer Networks. ISI Research Report ISI/RR-88-201, Information Sciences Institute (ISI), University of Southern California, 4676 Admiralty Way / Marina del Ray / California 90292-6695, June 1988.

Networks are becoming important in the day-to-day operations of business, the military, and government. As the use of networks grows, it is a wise precaution to assume that malicious attempts to sabotage a network will occur. Network operating software should not make the network susceptible to widespread failure if one router, or even several, deviate from acceptable behavior. Network software should be resistant to this manner of attack while preserving the desirable network attributes of flexibility and efficiency. This report points out that several commonly used routing procedures imply a vulnerability to attack, and presents a routing procedure that allows the development of operating software that is highly resistant to attack.

[Finn1996] Kate Finney. Mathematical notation in Formal Specification: Too Difficult for the Masses? *IEEE Transactions on Software Engineering*, 22(2):158–159, February 1996.

The phrase "not much mathematics required" can imply a variety of skill levels. When this phrase is applied to computer scientists, software engineers, and clients in the area of formal specification, the word "much" can be widely misinterpreted with disastrous consequences. A small experiment in reading specifications revealed that students already trained in discrete mathematics and the specification notation performed very poorly; much worse than could reasonably be expected if formal methods proponents are to be believed.

[Free1997] David H. Freedman and Charles C. Mann. *At Large: The Strange Case of the World's Biggest Internet Invasion*. Simon and Schuster, Rockefeller Center 1230 Avenue of the Americas New York, NY 10020, 1997.

At Large is the astonishing, never-before-revealed tale of perhaps the biggest and certainly the most disturbing computer attack to date, with ominous implications for the Internet, the digital highway over which much of the nation's business is now conducted.

For two years a computer break-in artist known only as "Phantom Dialer" seized control of hundreds — perhaps thousands — of computer networks across the country and around the world. Frightened network administrators watched helplessly as the intruder methodically slipped into universities, corporations, banks, federal agencies, and military facilities, including top-secret weapons-research sites. Working up to twenty hours a day, Phantom Dialer obsessively broke into one network after another — and no one knew who he was or what he was after. Was he a spy? Was he laying the groundwork for a single, massive theft?

As the number of victims mounted, Phantom Dialer became the subject of the first major investigation of the FBI's new computer-crime squad and one of the biggest manhunts in the history of electronic crime. Stoop-shouldered, monitor-tanned network administrators; nerdy, antisocial hacker wannabes; egotistical, visionary code jockeys; bureaucracy-bound computer-security agencies – all were caught up in the alternately frightening and absurd chase for Phantom Dialer. But when FBI agents finally burst into Phantom Dialer's house, they were stunned and dismayed by what they found. The decision was made not to prosecute but instead to keep the story quiet. And so the incident has remained secret, until now.

Though it reads like a thriller, *A*t Large is more than just a spellbinding account of one of the stranger episodes in the electronic America of the 1990s. it is also a sharply observed group portrait of the new wired world and an exposé of the technological flaws at its very core.

Most of all, *A*t Large is a warning bell for a nation rushing on-line. Even as it carries an ever-increasing amount of financial and personal information, the Internet is growing less, not more secure. The story of Phantom Dialer demonstrates the vulnerability of the global network: anyone can break in almost anywhere. Indeed, though few recognize it, the massive crime wave has already begun.

[GAO1996a] GAO. Information Security, Computer Attacks at Department of Defense Pose Increasing Risks. United States General Accounting Office, Report to Congressional Requesters, May 1996. GAO/AIMD-96-84. Included in Senate Hearing 104-701.

Attacks on Defense computer systems are a serious and growing threat. The exact number of attacks cannot be readily determined because only a small portion are actually detected and reported. However, Defense Information Systems Agency (DISA) data implies that Defense may have experienced as many as 250,000 attacks last year. DISA information also shows that attacks are successful 65 percent of the time, and that the number of attacks is doubling each year, as Internet use increases along with the sophistication of "hackers"[19]

[19]The term hackers has had a relatively long history. Hackers were at one time persons who explored the inner workings of computer systems to expand their capabilities, as opposed to those who simply used computer systems. Today the term generally refers to unauthorized individuals who attempt to penetrate

and their tools.

At a minimum, these attacks are a multimillion dollar nuisance to Defense. At worst, they are a serious threat to national security. Attackers have seized control of entire Defense systems, many of which support critical functions, such as weapons systems research and development, logistics, and finance. Attackers have also stolen, modified, and destroyed data and software. In a well-publicized attack on Rome Laboratory, the Air Force's premier command and control research facility, two hackers took control of laboratory support systems, established links to foreign Internet sites, and stole tactical and artifical intelligence research data.

The potential for catastrophic damage is great. Organized foreign nationals or terrorists could use "information warfare" techniques to disrupt military operations by harming command and control systems, the public switch network, and other systems or networks Defense relies on.

Defense is taking action to address this growing problem, but faces significant challenges in controlling unauthorized access to its computer systems. Currently, Defense is attempting to react to successful attacks as it learns of them, but it has no uniform policy for assessing risks, protecting its systems, responding to incidents, or assessing damage.

Training of users and system and network administrators is inconsistent and constrained by limited resources. Technical solutions being developed, including firewalls[20], smart cards[21], and network monitoring systems, will improve protection of Defense information. however, the success of these measures depends on whether Defense implements them in tandem with better policy and personnel solutions.

[GAO1996b] GAO. Information Security, Computer Attacks at the Department of Defense Pose Increasing Risks. Testimony Before the Permanent Subcommittee on Investigations, Committee on Governemental Affairs, U.S. Senate, May 22, 1996. Statement of Jack L. Brock, Jr., Director Defense Information and Financial Management Systems Accounting and Infromation Management Division, United States General Accounting Office; GAO/T-AIMD-96-92. Included in Senate Hearing 104-701.

Mr. Chairman and Members of the Subcommittee: Thank you for the opportunity to participate in the Subcommittee's hearings on the security of our nation's information systems. The Ranking Minority member and other Subcommittee members have expressed serious concerns about unauthorized access to sensitive information in computer systems at the Department of Defense and directed that we review information security at the Department. These concerns are well-founded. Defense has already experienced what it estimates to be hundreds of thousands of computer attacks originating from network connections, some of which have caused considerable damage. As you will learn from our testimony, these so-called hacker intrusions not only cost Defense tens of millions of dollars, but pose a serious threat to our national security....

[Garb2000] Lee Garber. Denial-of-Service Attacks Rip the Internet. *Computer*, 33(4):12–17, April 2000.

The Internet community is trying to cope with the series of distributed denial-of-service attacks that shut down some of the world's most high-profile and frequently visited Web sites, including Yahoo and Amazon.com, in February.

information systems; browse, steal, or modify data; deny access or service to others; or cause damage or harm in some other way.

[20]Firewalls are hardware and software components that protect on set of system resources (e.g., host systems, local area networks) from attack by outside network users (e.g., Internet users) by blocking and checking all incoming network traffic....

[21]Smart cards are access cards containing encoded information and sometimes a microprocessor and a user interface. The encoded information and/or the information generated by the processor are used to gain access to a computer system or facility

The attacks, which observers say cost victims millions of dollars, sent shock waves through the industry because they crippled some of the world's premier e-commerce sites.

And the problem was even worse than many people realize because more companies were attacked than those mentioned in the media, said Stephen Northcutt, director of the Global Incident Analysis Center (GIAC), an organization that conducts research and education programs on system administration, networking, and security....

[Garf1996] Simson Garfinkel and Gene Spafford. *Practical UNIX & Internet Security*. O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol, CA 95472, Second edition, 1996.

When *Practical UNIX Security* was first published in 1991, it became an instant classic. Crammed with information about host security, it saved many a UNIX system administrator and user from disaster.

This second edition is a complete rewrite of the original book. It's packed with twice the pages and offers even more practical information for UNIX users and administrators. You'll find coverage of features of many types of UNIX systems, including SunOS, Solaris, BSDI, AIX, HP-UX, Digital UNIX, and Linux. The first edition was practical, entertaining, and full of useful scripts, tips, and warnings. This edition is all those things — and more.

Practical UNIX and Internet Security includes detailed coverage of Internet security and networking issues, including World Wide Web security, wrapper and proxy programs, integrity management tools, secure programming, and how to secure TCP/IP services (e.g., FTP, SMTP, DNS). Chapters on host security contain up-to-date details on passwords, the UNIX filesystems, cryptography, backups, logging, physical security, telephone security, UUCP, firewalls, and dealing with breakins. You'll also find extensive summary appendixes on freely available security tools, references, and security-related organizations.

[Garf1997] Simson Garfinkel and Gene Spafford. *Web Security & Commerce*. O'Reilly & Associates, Inc, 1997.

Attacks on government web sites, break-ins at Internet service provides, electronic credit card fraud, invasion of personal privacy by merchants as well as hackers — is this what the World Wide Web is really all about?

Web Security & Commerce cuts through the hype and the front page stories. It tells you what the real risks are and explains how you can minimize them. Whether you're a casual (but concerned) web surfer or a system administrator responsible for the security of a critical web server, this book will tell you what you need to know. Entertaining as well as illuminating, it looks behind the headlines at the technologies, risks, and benefits of the Web. Topics include:

1. User safety — browser vulnerabilities, privacy concerns, and issues with Java, JavaScript, ActiveX, and plug-ins.
2. Digital certificates and cryptography — how digital certificates assure identity, what code signing is about, and the basics of how encryption works on the Internet today.
3. Web server security — detailed technical information about SSL, TLS, host security, server access methods, and secure CGI/API programming.
4. Commerce and society — how digital payments work, what blocking and censorship software is about, and what civil and criminal issues you need to understand.

[Gass1988] Morrie Gasser. *Building a Secure Computer System*. Van Nostrand Reinhold Company Inc., 115 Fifth Avenue, New York, New York 10003, 1988.

This book is for the practicing computer professional who wants to understand — and perhaps implement — technical solutions to computer security problems. It covers the state of the art of applied computer security technology developed over the last fifteen or twenty years. It is a guide to building systems, not an exhaustive academic study, and provides enough information about selected techniques to give you a well-rounded understanding of the problems and solutions.

It is not possible in one book to treat all applications of security while retaining the technical depth

needed to cover each topic adequately. I have concentrated on applications for which prevailing literature is weak: operating systems, hardware architecture, networks, and practical verification. Subjects about which books are already available, such as database security and cryptographic algorithms, receive less discussion here....

[Giff1988] David K. Gifford. Natural Random Numbers. Technical Report MIT / LCS / TM-371, Massachusetts Institute of Technology, August 1988.

We present a method for generating random numbers from natural noise sources that is able to produce random numbers to any desired level of perfection. The method works by transducing a physical noise source to generate a stream of biased natural bits, and then applying an unbiasing algorithm. The Wiener-Kinchine relation is used to derive the autocorrelation present in the stream of biased bits and to define safe sampling rates. Experimental results from an implementation of our method support our analysis. One consequence of our analysis is that a broad class of natural random number generators, including ours, can not generate absolutely perfect random numbers.

[Glig1983] Virgil D. Gligor. A Note on the Denial-of-Service Problem. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 139–149, 1983.

A simple and general definition of denial of service in operating systems is presented herein. It is argued that no current protection mechanism nor model resolves this problem in any demonstrable way. A set of examples from known systems is presented in order to delimit the scope of the problem. The notion of interuser dependency is introduced and identified as the common cause for all problem instances. Necessary and sufficient conditions for solutions are stated and justified informally. The relative complexity of undesirable (and unspecified) interuser dependencies is also discussed.

[Gold1996] H.H. Goldstine and Adele Goldstine. The Electronic Numerical Integrator and Computer (ENIAC). *IEEE Annals of the History of Computing*, 18(1):10–16, Spring 1996. This paper was first published in Mathematical Tables and Other Aids to Computation just after the ENIAC was announced in 1946.... reprinted in this issue [with] permission of the American Mathematical Society and the National Academy of Sciences.

It is our purpose in the succeeding pages to give a brief description of the ENIAC and an indication of the kinds of problems for which it can be used. This general purpose electronic computing machine was recently made public by the Army Ordinance Department for which it was developed by the Moore School of Electrical Engineering. The machine was developed primarily for the purpose of calculating firing tables for the armed forces. Its design is, however, sufficiently general to permit the solution of a large class of numerical problems which could hardly be attempted by more conventional computing work....

[GR1995] Wilson George R. Data Security by Design. *Progressive Architecture*, pages 82–84, March 1995.

Most office buildings are designed to stop physical intrusion, but electronic surveillance makes it easy to lift computer data and to eavesdrop on meetings. The author discusses a number of techniques the architect can use to deter electronic surveillance, including metal shielding and specially designed windows.

[Grah1999] Bradley Graham. Military Grappling With Rules for Cyber Warfare. *Washington Post*, page A1, November 8 1999.

During last spring's conflict with Yugoslavia, the Pentagon considered hacking into Serbian computer networks to disrupt military operations and basic civilian services. But it refrained from doing so, according to senior defense officials, because of continuing uncertainties and limitations surrounding the emerging field of cyber warfare....

[Gram1984] F.T. Grampp and R.H. Morris. UNIX Operating System Security. *AT&T Bell Laboratories Technical Journal*, 63(8):1649–1672, October 1984.

Computing systems that are easy to access and that facilitate communication with other systems are by their nature difficult to secure. Most often, though, the level of security that is actually achieved is far below what it could be. This is due to many factors, the most important of which are the knowledge and attitudes of the administrators and users of such systems. We discuss here some of the security hazards of the *U*NIX (TM) operating system, and we suggest ways to protect against them, in the hope that an educated community of users will lead to a level of protection that is stronger, but far more importantly, that represents a reasonable and thoughtful balance between security and ease of use of the system. We will not construct parallel examples for other systems, but we encourage readers to do so for themselves.

[Gros1988] Morton Grosser. Hack at the Screen Stalk. *Communications of the ACM*, 31(8):945–946, August 1988. This is part of the "Letters" in the "ACM Forum" section of the journal; the entire set of unrelated letters appears on pp. 944–947.

I immensely enjoyed Clifford Stoll's article "Stalking the Wily Hacker" in the May 1988 issue of *Communications* (pp. 484–97). Since Stoll included a sidebar with some interpretations of the word hacker, I would like to add a gloss on the origins of the term as presently used in the computing community.

The "legitimate" etymology of this slang word is often traced to the noun or verb form "hack." Eric Partridge points out in his *Dictionary of Slang and Unconventional English* that the noun has been slang for a harlot or bawd at least as far back as 1730, and Robert Chapman's *New Dictionary of American Slang* notes that since the early 1800s the word has meant a try or attempt....

[Guha1995] Biswaroop Guha. Vulnerability Analysis of the TCP/IP Suite. Master's thesis, University of California Davis, August 1995.

Networking is an important aspect of the modern computing environment, and the Transmission Control Protocol / Internet Protocol (TCP/IP) [1][22] suite is a very widely used technique that is employed to interconnect systems. However, there exist several security vulnerabilities in the TCP specification and additional weaknesses in a number of widely-available implementations of TCP. These vulnerabilities may enable an intruder to "attack" TCP-based systems, enabling him/her to "hijack" a TCP connection or cause denial of service to legitimate users. We analyze TCP code via a "reverse engineering" technique called "slicing" to identify several of these vulnerabilities, especially those that are related to the TCP state-transition diagram. We discuss many of the flaws present in the TCP implementation of many widely used operating systems, such as SUNOS 4.1.3, SVR4, and ULTRIX 4.3. We describe the corresponding TCP attack "signatures" (including the well-known 1994 Christmas Day Mitnick Attack) and provide recommendations to improve the security state of a TCP-based system, e.g., incorporation of a "timer escape route" from every TCP state.

[22] [Post1981b].

[Guha1996] Biswaroop Guha and Biswanath Mukherjee. Network Security Via Reverse Engineering of TCP Code: Vulnerability Analysis and Proposed Solutions. In *IEEE Infocom*, pages 603–610, 1996.

The Transmission Control Protocol/Internet Protocol (TCP/IP) suite is widely used to interconnect computing facilities in modern network environments. However, there exist several security vulnerabilities in the TCP specification and additional weaknesses in a number of its implementations. These vulnerabilities may allow an intruder to "attack" TCP-based systems, enabling him/her to "hijack" a TCP connection or cause denial of service to legitimate users. We analyze the TCP code via a "reverse engineering" technique called "program slicing" to identify several of these vulnerabilities, especially those that are related to the TCP state-transition diagram. We discuss many of the flaws present in the TCP implementation of many widely used operating systems, such as SUNOS 4.1.3, SVR4, and ULTRIX 4.3. We describe the corresponding TCP attack "signatures" (including the well-known 1994 Christmas Day Mitnick Attack) and provide recommendations to improve the security state of a TCP-based system, e.g., incorporation of a "timer escape route" from every TCP state.

[Guha1997] Biswaroop Guha and Biswanath Mukherjee. Network Security via Reverse Engineering of TCP Code: Vulnerability Analysis and Proposed Solutions. *IEEE Network*, 11(4):40–48, July/August 1997.

The Transmission Control Protocol/Internet Protocol (TCP/IP) suite is widely employed to interconnect computing facilities in today's network environments. However, there exist several security vulnerabilities in the TCP specification and additional weaknesses in a number of its implementations. These vulnerabilities may allow an intruder to "attack" TCP-based systems, enabling him/her to "hijack" a TCP connection or cause denial of service to legitimate users. The authors analyze the TCP code via a "reverse engineering" technique called "program slicing" to identify several of these vulnerabilities, especially those that are related to the TCP state-transition diagram. They discuss many of the flaws present in the TCP implementation of many widely used operating systems, such as SUNOS 4.1.3, SVR4, and ULTRIX 4.3. The corresponding TCP attack "signatures" (including the well-known 1994 Christmas Day Mitnick Attack) are described, and recommendations are provided to improve the security state of a TCP-based system (e.g., incorporation of a "timer escape route" from every TCP state). Also, it is anticipated that wide dissemination of this article's results may not only lead to vendor patches to TCP code to plug security holes, but also raise awareness of how program slicing may be used to analyze other networking software and how future designs of TCP and other software can be improved.

[Gupt1991] Sarbari Gupta and Virgil D. Gligor. Towards a Theory of Penetration-Resistant Systems and its Applications. In *Proceedings of the Computer Security Foundations Workshop IV*, pages 62–78, 1991.

A theoretical foundation for penetration analysis of computer systems is presented, which is based on a set of formalized design properties that characterize resistance to penetration. By separating the policy-enforcement mechanisms of a system from the mechanisms necessary to protect the system itself, and by using a unified framework for representing a large set of penetration scenarios, we develop an extensible model for penetration analysis. Furthermore, we illustrate how the model is used to implement automated tools for penetration analysis. The theory, model, and tools only address system-penetration patterns caused by unprivileged users' code interactions with a system.

[Gupt1992] Sarbari Gupta and Virgil D. Gligor. Experience with a Penetration Analysis Method and Tool. In *15th National Computer Security Conference*, pages 165–183, October 13–18 1992.

We present a penetration-analysis method, an experimental tool to support it, and the experience gained from applying this method and tool to the Secure Xenix (TM) source code. We also present several properties of penetration resistance, and illustrate their interpretation in Secure Xenix using several penetration experiments. We argue that the properties of reference monitor mechanisms are necessary but insufficient to provide penetration resistance for a system. However, the assurance process for establishing penetration resistance need not differ from that required for demonstrating support for access control policies.

[Gutm1996] Peter Gutmann. Secure Deletion of Data from Magnetic and Solid-State Memory. In *6th USENIX Security Symposium*, pages 77–89, Department of Computer Science, University of Auckand, July 22–25 1996. USENIX Association.

With the use of increasingly sophisticated encryption systems, an attacker wishing to gain access to sensitive data is forced to look elsewhere for information. One avenue of attack is the recovery of supposedly erased data from magnetic media or random-access memory. This paper covers some of the methods available to recover erased data and presents schemes to make this recovery significantly more difficult.

[Haar2000] Jaap C. Haartsen. The Bluetooth Radio System. *IEEE Personal Communications*, 7(1):28–36, February 2000.

A few years ago it was recognized that the vision of a truly low-cost, low-power radio-based cable replacement was feasible. Such a ubiquitous link would provide the basis for portable devices to communicate together in an ad hoc fashion by creating personal area networks which have similar advantages to their office environment counterpart, the local area network. Bluetooth^{TM} is an effort by a consortium of companies to design a royalty-free technology specification enabling this vision. This article describes the critical system characteristics and motivates the design choices that have been made.

[Hals1996] Fred Halsall. *Data Communications, Computer Networks and Open Systems*. Electronic Systems Engineering Series. Addison-Wesley, Harlow, England, fourth edition, 1996.

Drawing on his many years as a researcher and teacher, Fred Halsall presents the complex world of data communications and networks with clarity and thoroughness. An invaluable resource to both the student and the practicing computer professional, this fourth edition of the very successful *Data Communications, Computer Networks and Open Systems* has been extensively updated to reflect the rapid development in this field.
Highlights of the book include detailed coverage of:

- The essential theory associated with digital transmission
- Digital leased circuits included PDH, SONET and SDH
- Protocol basics including specification and implementation methods
- Legacy and wireless LANs
- High-speed LANs including 100 Base T and 100 VG AnyLAN
- Transparent and source routing bridges
- Packet switching and frame relay networks and their protocols
- Internetworking architectures, protocols and routing algorithms
- Multiservice broadband networks including ATM LANs and MANs
- The TCP/IP and OSI application protocols including X.400 and X.500
- Data encryption and network security algorithms
- Network management architectures including SNMP and CMIP

Fred Halsall is Newbridge Professor of Communications Engineering at the University of Wales, Swansea. He has been involved in research in this field for over 20 years and has published extensively during this time.

[Hebb1980] B. Hebbard, P. Grosso, T. Baldridge, C. Chan, D. Fishman, P. Goshgarian, T. Hilton, J. Hoshen, K. Hoult, G. Huntley, M. Stolarchuk, and L. Warner. A Penetration Analysis of the Michigan Terminal System. *Operating Systems Review*, 14(1):7–20, January 1980.

The successful penetration testing of a major time-sharing operating system is described. The educational value of such a project is stressed, and principles of methodology and team organization are discussed as well as the technical conclusions from the study.

[Hebe1996] L. Todd Heberlein and Matt Bishop. Attack Class: Address Spoofing. In *19th National Information Systems Security Conference*, volume 1, October 22–25 1996.

We present an analysis of a class of attacks we call address spoofing. Fundamentals of internetwork routing and communication are presented, followed by a discussion of the address spoofing class. The attack class is made concrete with a discussion of a well known incident. We conclude by dispelling several myths of purported security solutions including the security provided by one-time passwords.

[High1988] Harold Joseph Highland. Electromagnetic Eavesdropping Machines for Christmas? *Computers & Security*, 7(4):341–344, 1988. Highland is the editor of Computers & Security.

Almost 3 years ago we published "Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk" by Wim van Eck of the Netherlands PTT....[23]

Late this spring, I received a letter and a manual from John J.,Williams of Consumertronics. Mr.,Williams, a specialist in electronics and cryptography, has often communicated with me in the past about various topics. This time he sent me an extensive letter, a detailed manual and a letter received from Wim van Eck. He had written to Wim van Eck after reading the article published in the journal to point out that some technical details were missing.... A complete copy of van Eck's reply appears in Fig.,1....

In preparing the original copy of the van Eck paper, one element had not been included since he did not wish to reveal the electronic circuitry. Another omission was made when we did the final editing since we felt too that full data should not be disclosed....

Mr. Williams also provides the reader of his manual with a comprehensive schematic diagram, including the microchips and their names, to build the external synchronization unit. This had purposely been left out of van Eck's paper. The manual also includes the necessary formulae to adjust the horizontal and vertical frequencies.

Also missing from the van Eck paper was information about interfacing the external synch [sic] unit and the TV receiver. These are provided in the manual as shown in Fig.,2....

[Hind1998] Robert M. Hinden and Stephen E. Deering. IP Version 6 Addressing Architecture. Request for Comments (RFC) 2373, July 1998. Internet Engineering Task Force (IETF); http://www.ietf.org.

This specification defines the addressing architecture of the IP Version 6 protocol [Deer1998]. The document includes the IPv6 addressing model, text representations of IPv6 addresses, definition of IPv6 unicast addresses, anycast addresses, and multicast addresses, and an IPv6 node's required addresses.

[23] [vanE1985].

[Hink1997] Thomas H. Hinke, Harry S. Delugach, and Randall P. Wolf. Protecting Databases from Inference Attacks. *Computers & Security*, 16(8):687–708, 1997.

This paper presents a model of database inference and a taxonomy of inference detection approaches. The Merlin inference detection system is presented as an example of an automated inference analysis tool that can assess inference vulnerabilities using the schema of a relational database. A manual inference penetration approach is then offered as a means of detecting inferences that involve instances of data or characteristics of groups of instances. These two approaches are offered as practical approaches that can be applied today to address the database inference problem. The final section discusses future directions in database inference research.

[Hoff1990] Lance J. Hoffman, editor. *Rogue Programs: Viruses, Worms, and Trojan Horses*. Van Nostrrand Reinhold, 115 Fifth Avenue, New York, New York 10003, 1990.

The situation with computer virus protection today [reminds] me of that with automobiles prior to the advent of seat belts. Car manufacturers typically added safeguards (seat belts, air bags, etc.) only after security requirement, market demand, and government regulations became such that it made economic sense for the manufactures and did not threaten to put them at a competitive disadvantage....

Until now, we have also been in the *early warning stage* with respect to the use of computers. But currently, with the establishment of a handful of organizations around the world that study, capture, or attempt to control rogue programs, and with the appearance of books like this, we are entering the next stage — the *study stage*. Eventually, research may lead to technological developments and to laws and other evidence of a *regulatory stage*; indeed we have already seen embryonic legislation (see Part 2) that addresses these problems....
There are five parts in the book:

1. The introductory part contains overview material on virus identification, prevention, detection, and mitigation, as well as a comparison with immunology in the medical world.
2. The next part discusses societal, legal and ethical issues that are often ignored by the technical community but that will ultimately be resolved with or without its input to policymakers.
3. The third part examines virus attacks on personal computer systems and defenses against these attacks. A number of the better known viruses are discussed here. By examining these papers, the reader should get a good feel for typical PC-oriented attacks and for antiviral software mechanisms.
4. The next part deals with attacks of rogue programs (usually worms rather than viruses) on networks and what can be done to prevent or mitigate them.
5. Finally, the last part presents some theoretical models of computer viruses. Although these models may not be useful for the practitioner today, they may be extremely important in developing software and/or hardware that will defeat rogue programs in the year to come.

[Hoff1995] Lance J. Hoffman, editor. *Building in Big Brother: The Cryptographic Policy Debate*. Springer-Verlag New York, Inc., 1995.

With the ever-increasing flow of information on electronic highways, the need for secure and private communication is taking center stage. Whether it be the electronic transfer of money, the transmission of commercial information, or electronic mail among friends, senders and receivers need to know that others cannot intercept or read their messages or transmit false messages in their place. A controversial proposal by the American government involves the implementation of the "Clipper chip," a technical standard which raises the possibility of the insertion of a secure but tappable chip in many telephones and computers.

This book presents the best readings on cryptographic policy and current cryptography trends. Topics include: a survey of cryptography, the new "key escrow" systems, the government solution, the debate between law enforcement views and civil liberties, and export control analysis. Detailed technological descriptions of promising new software schemes are included as well as analysis of the constitutional issues by legal scholars. Important government cost analyses appear for the first time in any book.

Other highlights include the text of the new U.S.,digital telephony law and the pending encryption regulation bill and a list of hundreds of cryptographic products available around the world. There is even a paper on how to commit the perfect crime electronically, using public encryption.

[Holl1974] Dennis Hollingworth, Steve Glaseman, and Marsha Hopwood. Security Test and Evaluation Tools: An Approach to Operating System Security Analysis. The Rand Paper Series P-5298, Rand Corporation, Santa Monica, California 90406, September 1974.

As a result of studies of the security characteristics of selected large operating systems, it has become increasingly evident that any complex operating system requires testing and evaluation in order to validate the functional characteristics of the system and verify claims of improved security safeguards. Furthermore, over the next decade, it is likely that new systems will be subject to continuous testing and evaluation in much the same fashion, and for the same purposes, as are existing systems. As yet, the techniques employed in determining the security characteristics of system software are presently quite primitive, based primarily upon the notion of penetration testing — manually examining system source materials for security vulnerabilities. This suggests the development and refinement of tools and techniques of operating system security analysis. Some of the more desirable characteristics of such tools are explored in this document, and several example tools are described.

[Holl1976] Dennis Hollingworth and Richard Bisbey II. Protection Errors in Operating Systems: Allocation / Deallocation Residuals. Technical Report ISI/SR-76-7, USC / Information Sciences Institute, 4676 Admiralty Way / Marina del Rey, CA 90291, June 1976. Reproduced by U.S. Dept of Commerce, National Technical Information Service, Springfield, VA 22161.

A common security problem is the residual — data or access capability left after the completion of a process and not intended for use outside the context of that process. If the residual becomes accessible to another process, a security error may result. A major source of such residuals is improper or incomplete allocation / deallocation processing. The various types of allocation / deallocation residuals are discussed in terms of their characteristics and the manner in which they occur, and a semiautomatable search strategy for detecting sources of these residuals is presented.

[Horn1984] Charles Hornig. A Standard for the Transmission of the IP Datagrams over Ethernet Networks. Request for Comments (RFC) 894, April 1984. Internet Engineering Task Force (IETF); http://www.ietf.org.

This memo applies to the Ethernet (10-megabit/second, 48-bit addresses). The procedure for transmission of IP datagrams on the Experimental Ethernet (3-megabit/second, 8-bit addresses) is described in [3].[24].

[Howa1997] John D. Howard. *An Analysis of Security Incidents on the Internet.* Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213 USA, April 1997.

This research analyzed trends in Internet security through an investigation of 4,299 security-related incidents on the Internet reported to the CERT (R) Coordination Center (CERT (R)/CC) from 1989 to 1995. Prior to this research, our knowledge of security problems on the Internet was limited and primarily anecdotal. This information could not be effectively used to determine what government policies and programs should be,

[24]Postel, J., "A Standard for the Transmission of IP Datagrams over Experimental Ethernet Networks", RFC-895, USC/Information Sciences Institutes, April 1984.

or to determine the effectiveness of current policies and programs. This research accomplished the following: 1) development of a taxonomy for the classification of Internet attacks and incidents, 2) organization, classification, and analysis of incident records available at the CERT (R)/CC, and 3) development of recommendations to improve Internet security, and to gather and distribute information about Internet security.

With the exception of denial-of-service attacks, security incidents were generally found to be decreasing relative to the size of the Internet. The probability of any severe incident not being reported to the CERT (R)/CC was estimated to be between 0% and 4%. The probability that an incident would be reported if it was above average in terms of duration and number of sites, was around 1 out of 2.6. Estimates based on this research indicated that a typical Internet domain was involved in no more than around one incident per year, and a typical Internet host in around one incident every 45 years.

The taxonomy of computer and network attacks developed for this research was used to present a summary of the relative frequency of various methods of operation and corrective actions. This was followed by an analysis of three subgroups: 1) a case study of one site that reported all incidents, 2) 22 incidents that were identified by various measures as being the most severe in the records, and 3) denial-of-service incidents. Data from all incidents and these three subgroups were used to estimate the total Internet incident activity during the period of the research. This was followed by a critical evaluation of the utility of the taxonomy developed for this research. The analysis concludes with recommendations for Internet users, Internet suppliers, response teams, and the U.S. government.

[Icov1995] David Icove, Karl Seger, and William VonStorch. *Computer Crime: A Crimefighter's Handbook*. O'Reilly & Associates, Inc., 103 Morris Street, Suite A, Sebastopol, CA 95472, 1st edition, 1995.

Terrorist attacks on computer centers, electronic fraud on international funds transfer networks, viruses and worms in our software, corporate espionage on business networks, and crackers breaking into systems on the Internet...Computer criminals are becoming ever more technically sophisticated, and it's an increasing challenge to keep up with their methods.

Computer Crime: A Crimefighter's Handbook is for anyone who needs to know what today's computer crimes look like, how to prevent them, and how to detect, investigate, and prosecute them if they do occur. It contains basic computer security information as well as guidelines for investigators, law enforcement, and computer system managers and administrators.

Part I of the book contains a discussion of computer crimes, the computer criminal, and computer crime laws. It describes the various categories of computer crimes and profiles the computer criminal (using techniques developed for the FBI and other law enforcement agencies). Part II outlines the the risks to computer systems and personnel, operational, physical, and communications measures that can be taken to prevent computer crimes. Part III discusses how to plan for, investigate, and prosecute computer crimes, ranging from the supplies needed for criminal investigation, to the detection and audit tools used in investigation, to the presentation of evidence to a jury.

Part IV of the book contains a compendium of the computer-related U.S. federal statutes and all of the statutes of the individual states, as well as representative international laws. Part V contains a resource summary, detailed papers on computer crime, and a sample search warrant for a computer crime.

[IEEE1999] IEEE. IEEE Symposium on Security and Privacy 1980–1999. CD-ROM, 1999. Sponsored by the IEEE Computer Society Technical Committee on Security and Privacy.

Contains all twenty years of papers in Acrobat Portable Document Format (PDF) format.

[Jaya1997] N.D. Jayaram and P.L.R. Morse. Network Security — A Taxonomic View. In *European Conference on Security and Detection*. School of Computer Science, University of Westminster, UK, IEE, 28–30 April 1997. Conference Publication No. 437.

Rapid advancement in the technologies of communication and computers coupled with failing costs of communication and computer hardware have made networked computers the systems of choice in all organisations. The phenomenal growth of the internet, its non-discriminatory access philosophy, and the growing practice of internetworking have all provided unprecedented opportunities not only for benign information/ resource access but also for malign intrusions which pose enormous security problems for organisations. Breakthroughs in network connectivity bring in new security problems. This paper quantifies the class of security threats and mechanisms for meeting these threats in the age of the ubiquitous Web.

[Jonc1995] Laurent Joncheray. A Simple Active Attack Against TCP. In *5th USENIX Security Symposium*, pages 7–19, June 5–7 1995.

This paper describes an active attack against the Transport Control Protocol (TCP) which allows a cracker to redirect the TCP stream through his machine thereby permitting him to bypass the protection offered by such a system as a one-time password [SKEY] or ticketing authentication [Kerberos]. The TCP connection is vulnerable to anyone with a TCP packet sniffer and generator located on the path followed by the connection. Some schemes to detect this attack are presented as well as some methods of prevention and some interesting details of the TCP protocol behaviors.

[Kaba1995] M.E. Kabay. Penetrating Computer Systems and Networks. In Hutt Arthur E., editor, *Computer Security Handbook*, chapter 18, pages 18.1 – 18.22. John Wiley & Sons, 1995.

As preceding chapters have shown, information systems security, like all aspects of security, is far more than merely a technical sisue. Security depends on human beings to understand and carry out security procedures. Security must become part of the corporate culture — a consistent way of approaching all aspects of one's work.... Breaking into information systems can involve technical attacks (working on weaknesses in operating systems, security programs, networks, and application programs) or by what criminals have called *social engineering* (lies, bribes, and fraudulent misrepresentation). hackers, both criminal and recreational, also share information through underground bulletin boards....

[Kahn1996] David Kahn. *The Codebreakers: The Story of Secret Writing.* Scribner, 1230 Avenue of the Americas, New York, New York 10020, second (first copyright 1967) edition, 1996. The Comprehensive History of Secret Communication from Ancient Times to the Internet.

Codebreaking is the most important form of secret intelligence in the world today. It produces much more and much more trustworthy information than spies, and this intelligence exerts great influence upon the policies of governments. Yet it has never had a chronicler.

It badly needs one. It has been estimated that cryptanalysis saved a year of war in the Pacific, yet the histories give it but passing mention.... I have tried in this book to write a serious history of cryptology. It is primarily a report to the public on the important role that cryptology has played, but it may also orient cryptology with regard to its past and alert historians to the sub rosa influence of cryptanalysis. This book seeks to cover the entire history of cryptology. My goal has been twofold: to narrate the development of the various methods of making and breaking codes and ciphers, and to tell how these methods have affected men....

The magnificent, unrivaled history of codes and ciphers — how they're made, how they're broken, and the many and fascinating roles they've played since the dawn of civilization in war, business, diplomacy, and espionage — updated with a new chapter on computer cryptography and the Ultra secret.

Man has created codes to keep secrets and has broken codes to learn those secrets since the time of the Pharaohs. For 4,000 years, fierce battles have been waged between codemakers and codebreakers, and the story of these battles is civilization's secret history, the hidden account of how wars were won and lost, diplomatic intrigues foiled, business secrets stolen, governments ruined, computers hacked. From the XYZ Affair to the Dreyfus Affair, from the Gallic War to the Persian Gulf, from Druidic runes and the kaballah to outer space,

from the Zimmermann telegram to Enigma to the Manhattan Project, codebreaking has shaped the course of human events to an extent beyond any easy reckoning. Once a government monopoly, cryptology today touches everybody. It secures the Internet, keeps e-mail private, maintains the integrity of cash machine transactions, and scrambles TV signals on unpaid-for channels. David Kahn's *The Codebreakers* takes the measure of what codes and codebreaking have meant in human history in a single comprehensive account, astonishing in its scope and enthralling in its execution. Hailed upon first publication as a book likely to become the definite work of its kind, *The Codebreakers* has more than lived up to that prediction: it remains unsurpassed. With a brilliant new chapter that makes use of previously classified documents to bring the book thoroughly up to date, and to explore the myriad ways computer codes and their hackers are changing all of our lives, *The Codebreakers* is the skeleton key to a thousand thrilling true stories of intrigue, mystery, and adventure. It is a masterpiece of the historian's art.

[Karg1974] Paul A. Karger and Roger R. Schell. Multics Security Evaluation: Vulnerability Analysis. Technical Report ESD-TR-74-193, Information Systems Technology Applications Office; Deputy for Command and Management Systems; Electronic Systems Division (AFSC); L.G. Hanscon AFB, MA 01730, June 1974.

A security evaluation of Multics for potential use as a two-level (Secret / Top Secret) system in the Air Force Data Services Center (AFDSC) is presented. An overview is provided of the present implementation of the Multics Securtiy controls. The report then details the results of a penetration exercise of Multics on the HIS 645 computer. In addition, preliminary results of a penetration exercise of Multics on the new HIS 6180 computer are presented. The report concludes that Multics as implemented today is not certifiably secure and cannot be used in an open use multi-level system. However, the Multics security design principles are significantly better than other contemporary systems. Thus, Multics as implemented today, can be used in a benign Secret / Top Secret environment. In addition, Multics forms a base from which a certifiably secure open use multi-level system can be developed.

[Kauf1995] Charlie Kaufman, Radia Perlman, and Mike Speciner. *Network Security: PRIVATE Communication in a PUBLIC World*. Prentice Hall, Englewood Cliffs, New Jersey, 1995.

A comprehensive yet comprehensible and witty guide to the latest advances in computer network security protocols. The author team includes Charlie Kaufman, currently chief security architect for Lotus Notes, and formerly Network Security Architect at Digital Equipment Corporation; best-selling author Radia Perlman, currently with Novell, and a specialist in the areas of bridging and routing, as well as sabotage-proof networks; and Mike Speciner, Chief Architect at ColorAge, an expert in number theory and operating systems, and formerly the security expert for Camex, Inc.

Network Security examines the state of computer network security — what works, what doesn't, and why. explains clearly the cryptography algorithms on which most network data systems depend. provides comprehensive descriptions of many authentication systems, including Kerberos, NetWare, Lotus Notes, DASS, and KryptoKnight. offers a rigorous treatment of secure electronic mail standards, including PEM, PGP, and X.400. describes classic security pitfalls and how to avoid them when designing protocols. In this book, the authors go beyond documenting standards and technology; they contrast competing schemes, explain weaknesses and strengths, and describe common mistakes people make when intending to design secure systems.

[Kels1999] John Kelsey, Bruce Schneier, and Niels Ferguson. Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator. In Howard Heys and Carlisle Adams, editors, *Selected Areas in Cryptography: 6th Annual International Workshop, SAC'99*, pages 13–33, Coutnerpane Systems; {kelsey, schneier, niels}@counterpane.com, August 9–10 1999.

Springer, Kingston, Ontario, Canada. http://www.counterpane.com/yarrow-notes.html.

We describe the design of Yarrow, a family of cryptographic pseudo-random number generators (PRNG). We describe the concept of a PRNG as a separate cryptographic primitive, and the design principles used to develop Yarrow. We then discuss the ways that PRNGs can fail in practice, which motivates our discussion of the components of Yarrow and how they make Yarrow secure. Next, we define a specific instance of a PRNG in the Yarrow family that makes use of available technology today. We conclude with a brief listing of open questions and intended improvements in future releases.

[Kemm1983] Richard A. Kemmerer. Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channels. *ACM Transactions on Computer Systems*, 1(3):256–277, August 1983.

Recognizing and dealing with storage and timing channels when performing the security analysis of a computer system is an elusive task. Methods for discovering and dealing with these channels have mostly been informal, and formal methods have been restricted to a particular specification language.

A methodology for discovering storage and timing channels that can be used through all phases of the software life cycle to increase confidence that all channels have been identified is presented. The methodology is presented and applied to an example system having three different descriptions: English, formal specification, and high-order language implementation.

[Kent1998] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, November 1998. Network Working Group.

This memo specifies the base architecture for IPsec compliant systems. The goal of the architecture is to provide various security services for traffic at the IP layer, in both the IPv4 and IPv6 environments. This document describes the goals of such systems, their components and how they fit together with each other and into the IP environment. It also describes the security services offered by the IPsec protocols, and how these services can be employed in the IP environment. This document does not address all aspects of IPsec architecture. Subsequent documents will address additional architectural details of a more advanced nature, e.g., use of IPsec in NAT environments and more complete support for IP multicast. The following fundamental components of the IPsec security architecture are discussed in terms of their underlying, required functionality....
a. Security Protocols — Authentication Header (AH) and Encapsulating Security Payload (ESP)
b. Security Associations — what they are and how they work, how they are managed, associated processing
c. Key Management — manual and automatic (The Internet Key Exchange (IKE))
d. Algorithms for authentication and encryption
This document is not an overall Security Architecture for the Internet; it addresses security only at the IP layer, provided through the use of a combination of cryptographic and protocol security mechanisms.

[Klei1976] Leonard Kleinrock. ARPANET Lessons. In *International Conference on Communications*, pages 20–1 – 20–6, 1976.

Flow control is an essential function in computer networks but it is beset with subtle dangers. The ARPANET has taught us many lessons in this regard, some of which we discuss in this paper. Specifically, we identify and expose a number of deadlocks and degradations and then present the remedy to these traps as implemented in the ARPANET.

[Knig2000] Eric Knight. Computer Vulnerabilities. www.securityparadigm.com, March 2000. DRAFT.

Vulnerabilities are the tricks-of-the-trade for hackers, giving an intruder the ability to heighten one's access by exploiting a flawed piece of logic inside the code of a computer. Like the hackers that seek them out, vulnerabilities are usually quite mysterious and hard to prove they even exist. Many people whom are introduced to vulnerabilities for the first time are confused or disturbed at what they see — undocumented source code, usually performing a series of tasks which don't make a considerable amount of sense to the uninformed. Rightly so, because many vulnerabilities may exist in unfamiliar environments or using unfamiliar techniques.

As security experts get acquainted with vulnerabilities and how they are exploited, the methods of exploitation appear random and chaotic — each and every one with seemingly unpredictable results. It has been theorized that this comes from the fact that bugs are mistakes, and does not follow the course of intelligent reason. however, vulnerabilities can be categorized in ways that make more sense to the person investigating the problems at hand.

This book describes the vulnerabilities, both categorization and the exploitation logic, stemming from a centralized "gray area" approach. As the book author, I've decided to pull no punches at all, explaining how, in step by step detail, how one could take any form of vulnerability at any level and use it to control computer systems, the users, and administrators. The intent here is to teach, in as graphic detail as possible, the extent of each and every problem, and how it can be exploited. A good working knowledge of Microsoft Windows, UNIX, and TCP/IP are mandatory for a good understanding of computer vulnerabilities.

Hopefully this document will be used to define the forensic sciences stemming from computer crime, providing answers to the reasoning that hackers would use in a break-in. By following the approaches given in this book, an investigator can mirror the tracks of a hacker's logic as they intrude upon a computer network and understand the reasoning that goes on behind the attack.

[Knut1981] Donald Ervin Knuth. *Random Numbers*, volume 2. Seminumerical Algorithms of *Computer Science and Information Processing*, chapter 3, pages 1–177. Addison-Wesley, 2d edition, 1981. Michael A. Harrison, Consulting Editor.

The algorithms discussed in this book deal directly with numbers; yet I believe they are properly called *seminumerical*, because they lie on the borderline between numeric and symbolic calculation. Each algorithm not only computes the desired answers to a problem, it also is intended to blend well with the internal operators of a digital computer. In may cases a person will not be able to appreciate the beauty of such an algorithm unless he or she also has some knowledge of a computer's machine language; the efficiency of the corresponding machine program is a vital factor that cannot be divorced from the algorithm itself. The problem is to find the best ways to make computers deal with numbers, and this involves tactical as well as numerical considerations. Therefore the subject matter of this book is unmistakeably a part of computer science, as well as of numerical mathematics....

This volume comprises Chapters 3 and 4 of the complete series. Chapter 3 is concerned with "random numbers": it is not only a study of various methods for generating random sequences, it also investigates statistical tests for randomness, as well as the transformation of uniform random numbers into other types of random quantities; the latter subject illustrates how random numbers are used in practice. I have also included a section about the nature of randomness itself....

[Koch1999] Paul Kocher, Joshua, Jaffe, and Benjamin Jun. Differential Power Analysis. In *Advanced in Cryptology - CRYPTO 99: 19th Annual International Cryptology Conference*, number 1666 in Lecture Notes in Computer Science, pages 388–397. Springer Verlag, Berlin, Heidelberg, New York, 1999.

Cryptosystem designers frequently assume that secrets will be manipulated in closed, reliable computing environments. Unfortunately, actual computers and microchips leak information about the operations they process. This paper examines specific methods for analyzing power consumption measurements to find secret keys from tamper resistant devices. We also discuss approaches for building cryptosystems that can operate securely in existing hardware that leaks information.

[Koni1976] William L. Konigsford. A Taxonomy of Operating-System Security Flaws. Technical Report UCID-17422, Lawrence Livermore Laboratory, November 1 1976.

Concern over the privacy and security of computerized data has caused government and the private sector to investigate means of insuring the protection of such data from unauthorized disclosure. Experience with the past breakdown of existing protection mechanisms shows that the computer operating system is a primary cause of such disclosures.

RISOS Project members at Lawrence Livermore laboratory are engaged in this type of research, and they have developed a systematic approach to the problem of flaw detection. The three main components of this approach use test teams, computerized test tools, and past experience in developing generic classes of security flaws.

This paper addresses the categorization of flaws into generic classes so that the nature of such flaw groups can be understood. This understanding can then be utilized in detecting flaws in existing systems and applied in future system development.

[Kopk1999] Helmut Kopka and Patrick W. Daly. *A Guide to LaTeX*. Addison-Wesley, third edition, 1999.

If you are a user with little or no experience of computer computers or text formatting and you want to master LATEX to produce documents of high quality, then this book is essential reading. Fully revised to cover the most up-to-date versions of LATEX this accessible and practical tutorial contains all of the information you will need to get up and running with LATEX and is an essential reference tool to users at all levels.
This book will enable you to:

- Master the basics of LATEX and explore more advanced topics including user-defined extensions
- Get up to speed with the latest LATEX extensions for adaptations to other languages
- Explore numerous practical examples and pick up handy tips for avoiding common problems
- Benefit from detailed appendices including the Command Summary and Summary Tables

New to this edition NEW Completely updated to cover the latest releases and upgrades of LATEX **NEW** Covers new features including graphics importation and PostScript font installation **NEW** Section on LATEX and the World Wide Web **NEW** Section on LATEX on Windows & Windows NT **NEW** Section on installations for 32 bit PCs

[Krau1979] Leonard I. Krauss and Aileen MacGahan. *Computer Fraud and Countermeasures*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1979.

This book deals with computer fraud prevention, detection, deterrents, investigation, loss recovery, and risk management. The term "computer fraud" is our shorthand way of referring to *computer-assisted* or *computer-related* crimes. The people who commit these crimes may use the computer either directly or as a vehicle for deliberate misrepresentation or deception, usually to cover up the embezzlement or theft of money, goods, services, or information.

Avoiding the sensationalism that so often characterizes the current literature on the subject, this book is addressed to business executives, financial and administrative officers, data processing managers, systems analysts, auditors, corporate and computer security supervisors, law enforcement investigators, and others who are responsible for providing direction in coping with the risk of computer fraud.... The book is divided into four main sections: *Section I — Understanding the Problem: .... Section II — Prevention, Detection, and Deterrents: .... Section III — Loss Recovery, Legal, and Investigative Considerations: .... Section IV — Implementing Your Loss Control Program:....*

[Krsu1997] Ivan Krsul. Computer Vulnerability Analysis Thesis Proposal. Technical Report CSD-TR-97-026, The *COAST* Laboratory Department of Computer Sciences Purdue University, West Lafayette, IN 47909-1398, April 15 1997.

Computer security professionals and researchers do not have a history of sharing and analyzing computer vulnerability information. Scientists and engineers from older or more established fields have long understood that publicizing, analyzing, and learning from other people's mistakes is essential to the stepwise refinement of complex systems. Computer scientists, however, have not followed suit. Programmers reinvent classical programming mistakes, contributing to the reappearance of known vulnerabilities.

In the recent past, computer systems have come to be a part of critical systems that have a direct effect on the safety and well-being of human beings and hence we must have lower tolerance for software failures.

In the dissertation I will attempt to show that computer vulnerability information presents important regularities and these can be detected, and possibly visualized, providing important insight about the reason of their prevalence and existence. The information derived from these observations could be use to improve on all phases of the development of software systems, as could be in the design, development, debugging, testing and maintenance of complex computer systems that must implement a set of policies defined by security analysis.

A significant portion of the work that must be performed will concentrate on the development of classifications and taxonomies that will permit the visualization and analysis of computer vulnerability information. I hope that these classifications and taxonomies applied to a collection of vulnerabilities will provide a set of features whose analysis will show that there are clear statistical clusterings and patterns caused because developers and programmers are not learning from each others mistakes. This analysis may be performed by applying statistical analysis and knowledge discovery tools.

[Krsu1998] Ivan Victor Krsul. *Software Vulnerability Analysis*. Ph.D. dissertation, Purdue University, May 1998. http://www.krsul.org.

The consequences of a class of system failures, commonly known as *software vulnerabilities*, violate security policies. They can cause the loss of information and reduce the value or usefulness of the system.

An increased understanding of the nature of vulnerabilities, their manifestations, and the mechanisms that can be used to eliminate and prevent them can be achieved by the development of a unified definition vulnerabilities, the development of a framework for the creation of taxonomies for vulnerabilities, and the application of learning, visualization, and statistical tools on a representative collection of software vulnerabilities.

This dissertation provides a unifying definition of software vulnerability based on the notion that it is security policies that defines what is allowable or desirable in a system. It also includes a framework for the development of classifications and taxonomies for software vulnerabilities.

This dissertation presents a classification of software vulnerabilities that focuses on the assumptions that programmers make regarding the environment in which their application will be executed and that frequently do not hold during the execution of the program.

This dissertation concludes by showing that the unifying definition of software vulnerability, the framework for the development of classification, and the application of learning and visualization tools can be used to improve security.

[Kuhn1998] Markus G. Kuhn. Cipher Instruction Search Attack on the Bus-Encryption Security Microcontroller DS5002FP. *IEEE Transactions on Computers*, 47(10):1153–1157, October 1998.

A widely used bus-encryption microprocessor is vulnerable to a new practical attack. This type of processor decrypts on-the-fly while fetching code and data, which are stored in RAM only in encrypted form. The attack allows easy, unauthorized access to the decrypted memory content.

[Kuma1994] Sandeep Kumar and Eugene H. Spafford. A Pattern Matching Model for Misuse Intrusion Detection. *National Computer Security Conference*, 17:11–21, October 11–14 1994. Proceedings: Communicating our Discipline, Strategies for the Emerging Information Infrastructures; Baltimore, Maryland.

This paper describes a generic model of matching that can be usefully applied to misuse intrusion detection. The model is based on Colored Petri Nets. Guards define the context in which signatures are matched. The

notion of start and final states, and paths between them define the set of event sequences matched by the net. partial order matching can also be specified in this model. The main benefits of the model are its generality, portability and flexibility.

[Kuma1995] Sandeep Kumar. *Classification and Detection of Computer Intrusion*. Ph.D. dissertation, Purdue University, August 1995.

Some computer security breaches cannot be prevented using access and information flow control techniques. These breaches may be a consequence of system software bugs, hardware or software failures, incorrect system administration procedures, or failure of the system authentication module. Intrusion detection techniques can have a significant role in the detection of computer abuse in such cases.

This dissertation describes a pattern matching approach to representing and detecting intrusions, a hitherto untried approach in this field. We have classified intrusions on the basis of structural interrelationships among observable system events. The classification formalizes detection of specific exploitations by examining their manifestations in the system event trace. Thus we can talk about intrusion signatures belonging to particular categories in the classification, instead of the vulnerabilities that result in intrusions.

The classification developed in this dissertation can also be used for developing computational models to detect intrusions in each category by exploiting the common structural interrelationships of events comprising the signatures in that category. We can then look at the signatures of interest that can be matched efficiently, instead of attempting to devise a comprehensive set of techniques to detect any violation of the security policy. We define and justify a computational model in which intrusions from our classification can be represented and matched. We also present experimental results based on an implementation of the model tested against real-world intrusions.

[Lai1982] Wai Sum Lai. Protocol Traps in Computer Networks — A Catalog. *IEEE Transactions on Communications*, COM-30(6):1434–1449, June 1982.

This paper is a compendium of potential protocol "traps" compiled from relevant literature on computer networks. The diversity of deadlocks and message ping-ponging conditions that can arise in computer networks is presented together with associated methods of solution.

[Lamp1973] Butler W. Lampson. A Note on the Confinement Problem. *Communications of the ACM*, 16(10):613–615, October 1973.

This note explores the problem of confining a program during its execution so that it cannot transmit information to any other program except its caller. A set of examples attempts to stake out the boundaries of the problem. Necessary conditions for a solution are stated and informally justified.

[Land1993] Carl E. Landwehr. How Far Can You Trust A Computer? In *SAFECOMP '93, Proceedings 12th International Conference on Computer Safety, Reliability and Security*, pages 313–325, October 1993. Proceedings published by Springer-Verlag.

The history of attempts to secure computer systems against threats to confidentiality, integrity, and availability of data is briefly surveyed, and the danger of repeating a portion of that history is noted. Areas needing research attention are highlighted, and a new approach to developing certified systems is described.

[Land1994] Carl E. Landwehr, Alan R. Bull, John P. McDermott, and William S. Choi. A Taxonomy of Computer Program Security Flaws. *ACM Computing Surveys*, 26(3):211–254, September 1994.

An organized record of actual flaws can be useful to computer system designers, programmers, analysts, administrators, and users. This survey provides a taxonomy for computer program security flaws, with an Appendix that documents 50 actual security flaws. These flaws have all been described previously in the open literature, but in widely separated places. For those new to the field of computer security, they provide a good introduction to the characteristics of security flaws and how they can arise. Because these flaws were not randomly selected from a valid statistical sample of such flaws, we make no strong claims concerning the likely distribution of actual security flaws within the taxonomy. However, this method of organizing security flaw data can help those who have custody of more representative samples to organize them and to focus their efforts to remove and, eventually, to prevent the introduction of security flaws.

[Lapr1995] Jean-Claude Laprie. *Dependability — Its Attributes, Impairments and Means*, chapter 1, pages 3–24. Basic Research Series. Springer Verilag Berlin Heidelberg New York, 1995. B. Randell J. - C. Laprie H. Kopetz B. Littlewood (Eds.).

This paper gives the main definitions relating to dependability, a generic concept including as special cases such atributes (sic) as reliability, availability, safety, security, maintainability. The various impairments to dependability (faults, errors, failures) and means for dependability (especially fault tolerance, fault removal, fault forecasting) are emphasized.

[Levy1984] Steven Levy. *Hackers: Heroes of the Computer Revolution.* Dell Publishing, a division of Bantam Doubleday Dell Publishing Group, Inc., 666 Fifth Avenue / New York, NY 10103, 1984.

I was first drawn to writing about hackers — those computer programmers and designers who regard computing as the most important thing in the world — because they were such fascinating people. Though some in the field used the term "hacker" as a form of derision, implying that hackers were either nerdy social outcasts or "unprofessional" programmers who wrote dirty, "nonstandard" computer code, I found them quite different. Beneath their often unimposing exteriors, they were adventurers, visionaries, risk-takers, artists . . . and the ones who most clearly saw why the computer was a truly revolutionary tool. Among themselves, they knew how far one could go by the immersion in to the deep concentration of the hacking mind-set: one could go infinitely far. I came to understand why true hackers consider the term an appellation of honor rather than a pejorative.

As I talked to these digital explorers, ranging from those who tamed multimillion-dollar machines in the 1950s to contemporary young wizards who masters computers in their suburban bedrooms, I found a common element, a common philosophy which seemed tied to the elegantly flowing logic of the computer itself. It was a philosophy of sharing, openness, decentralization, and getting your hands on machines at any cost — to improve the machines, and to improve the world. This Hacker Ethic is their gift to us: something with value even to those of use with no interest at all in computers.

It is an ethic seldom codified, but embodied instead in the behavior of hackers themselves....

[Levy1996] Steven Levy. Wisecrackers. *Wired*, pages 128–134, 196–198, 200, 202, March 1996.

If you're putting your faith in cryptography to protect your privacy, we have some garage-band hackers who have been famously cracking, not creating crypto — that we'd like you to meet....

A highlight of Crypto'95 was a rambling speech by a grizzled, bearded man [whose] name was Robert Morris Sr.... His presence drew an auditorium full of fascinated cryptographers, who leaned forward in their seats, hoping for an epiphany... Trade secrets were not forthcoming. But Morris, in sort of the spirit of the Eastern masters, did utter a pair of truisms — fundamental tenets of the crypto creed, as it were.

Tenet Number One: Never underestimate the time, expense, and effort an opponent will expend to break a code.... Remember: beware the frontal assault.

Tenet Number Two spoke to the code breakers: Look for plaintext.... Remember, exploit your opponent's mistakes.

[Lewi1999] Ted Lewis. UbiNet: The Ubiquitous Internet Will Be Wireless. *Computer*, 32(10):128, 126–127, October 1999.

> According to the Gartner Group, by 2005 the world will have a billion mobile phone users ("Market Risks: Security: The Downside of .com," Research Briefs, http://www.infoworld.com, 20 July 1999). By 2008, more people will access the Internet from a wireless device than a wired one. Cell phones will outnumber PCs sometime around 2005, and devices in 45 million cars, 300 million homes, and countless other nonoffice locations will render the Wintel PC as obsolete as the horse and buggy.
>
> Today's Internet is tethered to telephone wires and coaxial cable. Remove the encumbering wires, and Internet access rates soar. Eastern Europe, South America, and Asia learned this lesson a decade ago, finding it much faster and cheaper to interconnect by cell phone than PC. The only way the Internet will ever become ubiquitous is if it goes wireless. Therefore, it's the new wireless telephone companies, not the computer industry, that will bring a wireless, ubiquitous Internet — the UbiNet — to you.

[Lind1975] Richard R. Linde. Operating System Penetration. In *National Computer Conference*, pages 361–368, Santa Monica, California, May 19–22 1975. System Development Corporation.

> One of the favorite diversions of university students involves "beating" the system. In the case of operating systems, this has been a remarkably easy accomplishment. An extensive lore of operating system penetration, ranging from anecdotes describing students who have outsmarted the teacher's grading program to students who captured the system's password list and posted it on one of the bulletin boards, [Organick, Elliott I., *The MULTICS System: An Examination of Its Structure*, the MIT Press, Cambridge, Massachusetts, 1972.] has been collected on college campuses. Private industry has been victimized much more seriously. Here the lore of the "system" penetrations contains scenarios involving the loss of tens of thousands of dollars. [Palme, Jacob, "Software Security," *Datamation*, Vol. 20, No. 1, January 1974, pp. 51–55.]
>
> The Research and Development organization at SDC has been seriously involved with legitimate operating system penetration efforts. Under contract to government agencies and industry, SDC has assessed the secure-worthiness of their systems by attempts to gain illegal access to their operating system. As of this date, seven operating systems have been studied. This paper examines the successful penetration methodology employed, and the generic operating system functional weaknesses that have been found. Recommendations are made for improvement that can strengthen the penetration methodology.

[Lind1997] Ulf Lindqvist and Erland Jonsson. How to Systematically Classify Computer Security Intrusions. In *IEEE Security and Privacy*, pages 154–163, Department of Computer Engineering, Chalmers University of Technology, SE-412 96 Götegorg, Sweden, 1997.

> This paper presents a classification of intrusions with respect to technique as well as to result. The taxonomy is intended to be a step on the road to an established taxonomy of intrusions for use in incident reporting, statistics, warning bulletins, intrusion detection system etc. Unlike previous schemes, it takes the viewpoint of the system owner and should therefore be suitable to a wider community than that of system developers and vendors only It is based on data from a realistic intrusion experiment, a fact that supports the practical applicability of the scheme. The paper also discusses general aspects of classification, and introduces a concept called dimension. After having made a broad survey of previous work in the field, we decided to base our classification of intrusion techniques on a scheme proposed by Neumann and Parker in 1989 and to further refine relevant parts of their scheme. Our classification of intrusion results is derived from the traditional three aspects of computer security: confidentiality, availability and integrity.

[Lind1998] Ulf Lindqvist and Erland Jonsson. A map of security risks associated with using cots. *Computer*, 31(6):60–66, June 1998.

The widespread use of commercial off-the-shelf products in combination with increased internetworking calls for an analysis of the associated security risks. Combining Internet connectivity and COTS-based systems results in increased threats from both external and internal sources.

[Linn1766] Carolus Linnaeus. *Systema Naturae per Regna Tria Naturae, Secundum Classes, Ordines, Genera, Species, cum Characteribus, Differentis, Synonymis, Locis.* n/a, editio duodecima, reformata edition, 1766. Tomus I, Regnum Animale, 1766; Tomus II, Regnum Vegetabile, 1767; Tomus III, Regnum Lapideum, 1768.

Before his death in 1778, Lennaeus authored twelve editions of his *Systema Naturae*. Evidence suggests, however, that the eleventh edition was never published. Because the tenth edition became the foundation of animal systematics and nomenclature, the nine earlier editions are of interest only from an historical standpoint. Both the tenth and the twelfth editions, however, are of considerable taxonomic importance.

[Lipn1975] Steven B. Lipner. A Comment on the Confinement Problem. *ACM Operating System Review*, 9(5):192–196, November 1975.

The confinement problem, as identified by Lampson,[25] is the problem of assuring that a borrowed program does not steal for its author information that it processes for a borrow. An approach to proving that an operating system enforces confinement, by preventing borrowed programs from writing information in storage in violation of a formally stated security policy, is presented. The confinement problem presented by the possibility that a borrowed program will modulate its resource usage to transmit information to its author is also considered. This problem is manifest by covert channels associated with the perception of time by the program and its author; a scheme for closing such channels is suggested. The practical implications of the scheme are discussed.

[Litt1995] Bev Littlewood, Sarah Brocklehurst, Norman Fenton, Peter Mellor, Stella Page, David Wright, John Dobson, John McDermid, and Dieter Gollmann. *Towards Operational Measures of Computer Security: Concepts*, chapter 8A, pages 537–553, 571–572. Basic Research Series. Springer Verilag Berlin Heidelberg New York, 1995. B. Randell J. - C. Laprie H. Kopetz B. Littlewood (Eds.).

Ideally, a measure of the security of a system should capture quantitatively the intuitive notion of the 'the ability of the system to resist attack'. That is, it should be *operational*, reflecting the degree to which the system can be expected to remain free of security breaches under particular conditions of operation (including attack). Instead, current security *levels* at best merely reflect the extensiveness of safeguards introduced during the design and development of a system. Whilst we might expect a system developed to a higher level than another to exhibit 'more secure behaviour' in operation, this cannot be guaranteed; more particularly, we cannot infer what the actual security behaviour will be from knowledge of such a level. In the paper we discuss similarities between reliability and security with the intention of working towards measures of 'operational security' similar to those that we have for reliability of systems. Very informally, these measures could involve expressions such as the rate of occurrence of security breaches (cf rate of occurrence of failures in reliability), or the probability that a specified 'mission' can be accomplished without a security breach (cf reliability function). This new approach is based on the analogy between *system failure* and *security breach*. A number of other analogies to support this view are introduced. We examine this duality critically, and have identified a number of important open questions that need to be answered before this quantitative approach can be taken further. The work described here is therefore somewhat tentative, and one of our major intentions is to invite discussion about the plausibility and feasibility of this new approach.

[25] [Lamp1973].

[Loeb2000a] Vernon Loeb and Walter Pincus. CIA is Faulted For Not Probling Deutch's Actions. *Washington Post*, page A08, 2 February 2000.

The CIA should have asked the Justice Department to open a criminal investigation as soon as it discovered in December 1996 that former CIA director John M. Deutch had kept highly classified information on his home computers, according to a classified report by the CIA's inspector general....

Three days after Deutch left the CIA in December 1996, CIA security officials discovered thousands of pages of highly sensitive documents on Macintosh computers that Deutch used in his home. They also discovered classified information that Deutch had stored on portable memory cards for use in writing memorandums and keeping his personal journal, according to the inspector general's report, first reported in yesterday's New York Times.

There is no evidence that any of the information was obtained by unauthorized individuals, according to intelligence officials familiar with the report. But theft of the secrets cannot be rule out, they said, because Deutch's computers were connected to the Internet through America Online and Citibank's online personal banking system....

[Loeb2000b] Vernon Loeb. Tenet Offers 'No Excuse'. *Washington Post*, page A21, 3 February 2000.

CIA Director George J. Tenet said yesterday he has "no excuse" for the CIA's failure to notify the Justice Department in December 1996 that former CIA director John M. Deutch had kept "enormously sensitive material" on unsecure computers in his home....

[Losc1998] Peter A. Loscocco, Stephen D. Smalley, Patrick A. Muckelbauer, Ruth C. Taylor, S. Jeff Turner, and John F. Farrell. The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments. In *21st National Information Systems Security Conference*, pages 303–314, 1998.

Although public awareness of the need for security in computing systems is growing rapidly, current efforts to provide security are unlikely to succeed. Current security efforts suffer from the flawed assumption that adequate security can be provided in applications with the existing security mechanisms of mainstream operating systems. In reality, the need for secure operating systems is growing in today's computing environment due to substantial increases in connectivity and data sharing. The goal of this paper is to motivate a renewed interest in secure operating systems so that future security efforts may build on a solid foundation. This paper identifies several secure operating system features which are lacking in mainstream operating systems, argues that these features are necessary to adequately protect general application-space security mechanisms, and provides concrete examples of how current security solutions are critically dependent on these features.

[Loug1997] Daniel L. Lough, T. Keith Blankenship, and Kevin J. Krizman. A Brief Tutorial on Wireless LANs and IEEE 802.11. *looking .forward, a supplement to IEEE Computer*, 5(2):9–12, August 1997. Summer 1997.

Over recent years, the market for wireless communications has enjoyed tremendous growth. Wireless technology now reaches or is capable or reaching virtually every location on the face of the earth. Hundreds of millions of people exchange information every day using pagers, cellular telephones, and other wireless communication products. With tremendous success of wireless telephony and messaging services, it is hardly surprising that wireless communication is beginning to be applied to the realm of personal and business computing. No longer bound by the harnesses of wired networks, people will be able to access and share information on a global scale nearly anywhere they venture. this article will try and answer some basic questions of why and where wireless local area networks can be used, and present a brief description of some protocols that have been developed, with emphasis on IEEE 802.11.

[Lync1993] Daniel C. Lynch and Marshall T. Rose, editors. *Internet System Handbook*. Addison-Wesley, 1993.

The Internet covers the globe like no other vehicle for electronic communication. Expanding each day and encompassing a variety of different technologies, this electronic infrastructure is so massive and so diverse that its size and scope can only be estimated. The Internet is built upon a framework of open networking protocols and diverse internetworking technologies. A practical grasp of this infrastructure and of these protocols is critical to a successful experience with the Internet system.

The purpose of the *Internet System Handbook* is to provide in one comprehensive volume, for all Internet users, the most important and most useful technical information about the system. With contributions from specialists and pioneers in each aspect of the Internet's underlying technology, the handbook not only describes the system, but also furnishes expert insight into its use. Each chapter is self-contained, so that readers can move through the handbook in any order they like, or simply refer to the relevant chapters as a particular need arises.

Part I is an introduction to and a review of the Internet's geographic and technological evolution. Part II is devoted to underlying technologies, covering the core protocols, routing, and the major applications. Part III discusses the Internet infrastructure, covering directory services, tools for network management and performance, and operational security issues. Part IV focuses on changes in the Internet architecture made necessary by enormous growth in usage and the advent of new technologies. Finally, an annotated bibliography leads the reader through the most important publications on this subject.

This substantial volume, and the wealth of information it contains, has been designed for every professional — engineer, manager, administrator — working with the Internet system. One of the editors believes the the book will find a prominent place on your book shelf, while the other believes that you will lock it in your desk every night. Either way, the *Internet System Handbook* should be your favorite reference for Internet technology.

[Mars1997] Robert T. Marsh. Critical Foundations: Protecting America's Infrastructures: The Report of the President's Commission on Critical Infrastructure Protection. Technical report, United States Government, P.O. Box 46258, Washington, DC 20050-6258, October 13 1997. Presently, the report is available online at: http://www.ciao.gov/CIAO_Document_Library/PCCIP_Report.pdf.

Our national defense, economic prosperity, and quality of life have long depended on the essential services that underpin our society. These critical infrastructures — energy, banking and finance, transportation, vital human services, and telecommunications — must be viewed in a new context in the Information Age. The rapid proliferation and integration of telecommunications and computer systems have connected infrastructures to one another in a complex network of interdependence. This interlinkage has created a new dimension of vulnerability, which, when combined with an emerging constellation of threats, poses unprecedented national risk.

For most of our history, broad oceans, peaceable neighbors and our military power provided all the infrastructure protection we needed. But just as the terrible long-range weapons of the Nuclear Age made us think differently about security in the last half of the 20th Century, the electronic technology of the Information Age challenges us to invent new ways of protecting ourselves now. We must learn to negotiate a new geography, where borders are irrelevant and distances meaningless, where an enemy may be able to harm the vital systems we depend on without confronting our military power. National defense is no longer the exclusive preserve of government, and economic security is no longer just about business. The critical infrastructures are central to our national defense and our economic power, and we must lay the foundations for their future security on a new form of cooperation between government and the private sector.

[McGr1997] Gary McGraw and Edward W. Felten. *Java Security: Hostile Applets, Holes, and Antidotes*. John Wiley & Sons Inc., New York, 1997.

Do you know how to sort out fact from fiction when it comes to Java security? Did you know whenever you surf the Web with Netscape or Internet Explorer you are using Java? That means that someone else's code is running untested on your computer. Don't wait for hostile applet to show you how vulnerable you site is.

International security experts Gary McGraw and Edward Felten — leader of the famed Princeton team — tell you how Java security works, and how it doesn't. McGraw and Felten give you all the information you need to create a reasonable Java use strategy. *Java Security* gives you: *Guidelines for using Java more safely today. What to expect in the Java security future. A clear treatment of the risks of using Java. Vital information explaining the three prongs of the Java security model: the Byte Code Verifier, the Applet Class Loader, and the Security Manager. Clear explanations of holes in the Java security model.* Whether you're a webmaster, an information technology manager charged with creating an intelligent security policy for your organization, or a concerned Web user, this book is *must* reading.

[McPh1974] W. S. McPhee. Operating System Integrity in OS/VS2. *IBM System Journal*, 13(3):230–252, 1974.

System integrity is a major step in the direction of increased operating system security capability. This paper provides an explanation of the system integrity problem and how it relates to security. The general classes of integrity problems / solutions are discusses, and primary techniques used in VS2 Release 2 to correct or avoid integrity "exposures" are presented. User procedural requirements necessary to maintain system integrity, and the impact of system integrity support on the overall system are also addressed.

[Mead1996] Catherine Meadows. The NRL Protocol Analyzer: An Overview[26] *The Journal of Logic Programming*, 26(2):113–131, February 1996.

The NRL Protocol Analyzer is a prototype special-purpose verification tool, written in Prolog, that has been developed for the analysis of cryptographic protocols that are used to authenticate principals and services and distribute keys in a network. In this paper we give an overview of how the Analyzer works and describe its achievements so far. We also show how our use of the Prolog language benefited us in the design and implementation of the Analyzer.

[Morr1985a] Robert T. Morris. A Weakness in the 4.2BSD Unix[27] TCP/IP Software. Computing Science Technical Report 117, AT&T Bell Laboratories, Murray Hill, New Jersey 07974, February 25, 1985.

The 4.2 Berkeley Software Distribution of the Unix operating system (4.2BSD for short) features an extensive body of software based on the "TCP/IP" family of protocols. In particular, each 4.2BSD system "trusts" some set of other systems, allowing users logged into trusted systems to execute commands via a TCP/IP network without supplying a password. These notes describe how the design of TCP/IP and the 4.2BSD implementation allow users on untrusted and possibly very distant hosts to masquerade as users on trusted hosts. Bell Labs has a growing TCP/IP network connecting machines with varying security needs; perhaps steps should be taken to reduce their vulnerability to each other.

[Morr1985b] William Morris, editor. *The American Heritage Dictionary.* Houghton Mifflin, Boston, second edition, 1982,1985.

The publication of *The American Heritage Dictionary* in 1969 was a major event in the history of American lexicography. The goal of its editors, expressed by William Morris, was to create a new dictionary that would not only faithfully record our language but also add the sensible dimension of guidance toward grace and precision in the use of our language, which intelligent people seek in a dictionary. The overwhelming critical and popular success of the Dictionary has been testimony to the validity and achievement of that goal....

[26] This paper is an extended version of the paper The NRL Protocol Analyzer: An Overview, published in *The Proceedings of the Second International Conference on the Practical Applications of Prolog*, April 1994.

[27] Unix is a Trademark of AT&T Bell Laboratories.

[Mudg1997] Peter Mudge and Yobie Benjamin. Déjà Vu All Over Again. *Byte*, 22(11):81–86, November 1997.

Windows NT security is under fire. It's not just that there are holes, but that the[re] are holes that other OSes patched years ago....

Do you have a strange feeling? The feeling you've been somewhere or done something before? It's déjà vu, and we're developing a serious case of it as we hunt down bugs in Windows NT. It's not strange that there are bugs in it. We certainly have not come across any OS or piece of software that is bug-free.

The peculiar feeling comes from the fact that the bugs we're seeing are the same security holes that were fixed many years ago in older OSes....

The shame of it is that none of these threats are new to the security world. Why does an OS only five years old (compared to Unix's 25-year history) have these problems? NT may be another example of the veracity of Santayana's statement that "those who cannot remember the past are condemned to repeat it."

Let's look at NT's security by highlighting some of the breaches, how they work, and what you can do about them.

[Myer1980] Philip A. Myers. Subversion: The Neglected Aspect of Computer Security. Master's thesis, Naval Postgraduate School, Monterey, California, June 1980. Thesis Advisor: Roger R. Schell.

This thesis distinguishes three methods of attacks internal protection mechanisms of computers: inadvertent disclosure, penetration, and subversion. Subversion is characterized by three phases of operations: the inserting of trap doors and Trojan horses, the exercising of them, and the retrieval of the resultant unauthorized information. Insersion occurs over the entire life cycle of the system from the system design phase to the production phase. This thesis clarifies the high risk of using computer system, particularly so-called "trusted" subsystems for the protection of sensitive information. This leads to a basis for countermeasures based on the lifetime protection of security related system components combined with the application of adequate technology as exemplified in the security kernel concept.

[Nanc2000] Richard E. Nance and James D. Arthur. Future Operating Systems Transistion for the Tomahawk and UAV Programs. Technical Report SRC-00-004, Systems Research Center and Department of Computer Science, Virginia Polytechnic Institute and State University, Department of Computer Science, Virginia Tech, Blacksburg, VA 24061, 2000.

This report describes the work done under the Operating Systems Transition project, performed for the TOMAHAWK (THWK) and Unmanned Aerial Vehicle (UAV) programs at the Naval Surface Warfare Center, Dahlgran Division, from June 4, 1998 to October 31, 1999. The task involved four responsibilities:

1. Establish a set of high-level operating system functions needed to meet THWK/UAV requirments.
2. Review and revise the initial set of requirements to evolve a detailed set admitting to measurement and testing.
3. Identify and assess the likely effects of the Defense Information Infrastructure and Common Operating Environment (DII COE) program.
4. Assess the applicability of Windows NT as a hosting operating system for both programs.

Numerous documents and web-based sources were consulted in the development of this report. In a separate attachment, copies of the most significant sources are included to provide a more comprehensive understanding of the results reported herein....

This report stresses several issues that deserve consideration as both programs move forward with the emphasis on commercial-off-the-shelf (COTS) conformance on both hardware and software.

- The application systems must be developed and sustained in an environment that stresses correctness, reliability and adaptability.

- The UNIX derivation in each system currently includes some inadvertent restrictions that should be revisited and possibly removed....

With regard to Windows NT or Windows 2000, previously designated Windows NT 5.0, serving as the hosting operating system, the report raises several considerations. Currently, the reliability of Windows 2000 is questions because 80% of the source code is new and untested. No hard real-time capabilities are provided in Windows NT, and the security issue remains in question as it does for any COTS operating system (several vulnerabilities have appeared in the popular press over the past few months). Windows NT appears currently to suffer a scalability problem, and its application in a domain with eight processors or more exacts no speed-up. Finally, the source code in unavailable and the use of proprietary protocols is evident, giving real concerns as to any support of open systems architectures.

While the task did not permit in-depth investigation of other operating system alternatives, the rapid penetration of Red Hat LINUX 6.0 in server adoptions suggests a possible future consideration....

[Need1994] Roger M. Needham. Denial of Service: An Example. *Communications of the ACM*, 37(11):42–46, November 1994.

Security threats are often divided into three categories: breach of confidentiality, failure of authenticity, and unauthorized denial of service.... The objective of the present article is to consider a particular instance of a denial of service problem and to look at engineering considerations relevant to an appropriate defense. A major aspect is the complexity and danger that result from unthinking use of what seem to be simple cost-savings measures.

[Need1997] Roger M. Needham. The Changing Environment for Security Protocols. *IEEE Network*, 11(3):12–15, May/June 1997.

The systematic study of security protocols started, as far as the public literature is concerned, almost 20 years ago. A paper by M. D. Schroeder and the present writer [R. M. Needham and M.D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," Commun. ACM, 1978, pp. 993-99] may be taken as a specimen; it was written in 1977 and published in 1978. It was, of course, written against the background of the technology of the time and made various assumptions about the organizational context in which its techniques would be used. The substantial research literature that has since appeared has, by and large, made similar assumptions about the technological and organizational environments. Those environments have in fact changed very considerably, and the purpose of the present note is to consider whether the changes should affect our approach to security problems. It turns out that where confidentiality is concerned, as distinct from authenticity and integrity, there is indeed a new range of options.

[Nels1994] Ruth Nelson. What is a Secret — and — What does that have to do with Computer Security? In *Proceedings of New Security Paradigms Workshop*, pages 74–81, Information System Security, 48 Hardy Avenue, Watertown, MA 02172, 1994.

This paper questions some of the basic assumptions of computer security in the context of keeping secrets, and it finds some major discrepancies. It then proposes a new paradigm for functional security in computer systems.

The first conclusion of the paper is that secrecy and security cannot be expressed both algorithmically and accurately. The second conclusion of the paper is that functional security models, which look at the application software as well as the data, can be very useful. Use of more realistic models involves a more complex definition of secure systems, but it may reduce the conflict between security and function and may result in more effective secure systems.

[Nels1995] Victor P. Nelson, H. Troy Nagle, Bill D. Carroll, and J. David Irwin. *Digital Logic Circuit Analysis & Design*. Prentice Hall, 1995.

Four highly respected scholars and authors have blended their talents to craft a book that presents the reader with a clear, comprehensive, and state-of-the-art view of digital design theory and practice. **Features**

- Covers topic authoritatively and in-depth but with a minimum of formal mathematics
- Places a strong emphasis on developing and using systematic problem-solving and design approaches and includes over 250 working examples, a large number of in-depth design examples, and an entire chapter of design case projects
- Presents a thorough discussion of CAD issues and practices in an integrated manner, allowing CAD methods to be applied to the correlative concepts and design principles
- Contains two comprehensive chapters describing programmable logic devices and their applications in implementing digital circuits
- Includes an in-depth introduction ot testing and design for testability
- Offers good coverage of hierarchical modular design and standard digital circuit modules

[Neum1975] P.G. Neumann, L. Robinson, K.N. Levitt, R.S. Boyer, and A.R. Saxena. A Provably Secure Operating System. Technical Report SRI Project 2581, Contract DAAB03-73-C-1454, Prepared for USAECOM, Stanford Research Institute, Menlo Park, California 94025, 13 June 1975.

This report summarized work to date toward the development of a provably secure operating system. Discussed here are

- a methodology for the design, implementation, and proof of properties of large computing systems,
- the design of a secure operating system using the methodology,
- the security properties to be proven about this system,
- considerations for implementing such a system, and
- an approach to monitoring security and performance.

[Neum1978] Peter G. Neumann. Computer System Security Evaluation. In *National Computer Conference Proceedings*, volume 47, pages 1087–1095. AFIPS, June 1978.

This paper considers the problem of attaining computer systems and applications programs that are both highly secure and highly reliable. It contrasts two current alternative approaches, one remedial, the other preventive. A remedial approach is outlined based on a classification of software security violations suggested by Bisbey, Carlstedt, and Hollingworth at ISI. This remedial analysis is then related to a preventive approach, illustrated here by the formal SRI Hierarchical Development Methodology. Evaluation of system security is then considered by combining concepts from the preventive and remedial approaches. This combination of techniques seems to have significant potential in the attainment and evaluation of computer system security. Illustrations are given for three types of systems, the first two being systems explicitly designed with security in mind, and the first of those begin designed according to a formal methodology. The first system is the SRI design for a Provably Secure Operating System (PSOS), the second is Multics, and the third is UNIX....

[Neum1989] Peter G. Neumann and Donn B. Parker. A Summary of Computer Misuse Techniques. In *12th National Computer Security Conference*, pages 396–407, 1989.

We consider here general classes of computer misuse, including international security abuses and accidental misuses. The classification approach is indended to provide a basis for methodological threat analysis that assesses the significance of vulnerabilities in specific systems and networks. It is intended to increase the understanding of exploitable abuse techniques, and thereby to aid in reducing both the number of vulnerabilities and their seriousness.

[Neum1990] Peter G. Neumann. Rainbows and Arrows: How the Security Criteria Address Computer Misuse. In *13th National Computer Security Conference*, pages 414–422, 1–4 October 1990.

This paper examines the two main sets of computer security evaluation criteria and considers the extent to which each criterion combats various types of threats. Differences among the criteria sets are summarized, and recommendations are offered for improved coverage.

[Neum1995] Peter G. Neumann. *Computer Related Risks*. The ACM Press, a division of the Association for Computing Machinery, Inc. (ACM), 1995.

This book is based on a remarkable collection of mishaps and oddities relating to computer technology. It considers what has gone wrong in the past, what is likely to go wrong in the future, and what can be done to minimize the occurrence of further problems....

Many of the events described here have been discussed in the on-line computer news group, the *Forum on Risks to the Public in the Use of Computers and Related Systems*, which I have moderated since its inception in 1985, under the auspices of the Association for Computing Machinery (ACM)....

Most of the events selected for inclusion relate to roles that computers and communication systems play in our lives. Some events exhibit problems with technology and its application; some events illustrate a wide range of human behavior, such as malice, inadvertent actions, incompetence, ignorance, carelessness, or lack of experience; some events are attributable to causes over which we have little control, such as natural disasters.... Because such events continue to happen and because they affect us in so many different ways, it is essential that we draw realistic conclusions from this collection — particularly if the book is to help up avoid future disasters. Indeed, the later chapters focus on the technology itself and discuss what can be done to overcome or control the risks....

[Niel1976] N.R. Nielsen, D.H. Brandin, J.D. Madden, B. Ruder, and G.F. Wallace. Computer System Integrirty Safeguards: System Integrity Maintenance. Technical Report Grant Number DCR74-23774, Prepared for the National Science Foundation, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, October 1976.

This report presents the results of the first phase of the Computer System Integrity Research Program. This research focused on the identification and analysis of the types of computer system integrity safeguards that would have been effective in preventing, detecting, or mitigating the effects of reported incidents of computer system integrity violations.

More than 350 cases of integrity violations form the base for this research. Cases are classified into 71 violation categories: 34 categories of safeguards encompassing 143 individual safeguards are identified as having application against these cases. Each individual safeguard is evaluated with respect to its range of applicability, effectiveness, cost characteristics, and other factors.

This report describes the violation and safeguard categorizations that were developed, and it provides a description of each of the individual safeguards with examples and an evaluation summary. A variety of data tables are included that portray the distribution of reported violations over the various violation categories, the distribution of applicable types of safeguards over the reported cases and violation categories, and the applicability of the individual safeguards to violations. Information is also presented on the more highly effective safeguards, the more broadly applicable safeguards, the cost-effective and cost-ineffective safeguards, the type of application for which the effective safeguards are intended, and the safeguards most applicable to each violation category.

In addition to the above, the report contains sections describing the background of the project and areas that have been identified as having potential for further research. A summary section is included that briefly presents all major findings stemming from the first phase research work.

[NSS1998] Network Security Solutions NSS. Techniques Adopted By 'System Crackers' When Attempting To Break Into Corporate or Sensitive Private Networks. http://www.ns2.co.uk/archive/cracker.txt, December 1998. Front-line Information Security Team, fist@ns2.co.uk.

This white paper was written to help give systems administrators and networks operations staff an insight into the tactics and methodologies adopted by typical system crackers when targeting large networks.

This document is not a guide about how to secure your networks, although it should help you identify security risks in your networked environment and maybe help point out any accidents that are waiting to happen....

[O'Ha1999] Bob O'Hara and Al Petrick. *The IEEE 802.11 Handbook: A Designer's Companion*. Standards Infomration Network, IEEE Press, 3 Park Avenue, New York, New York, 10016-5997, 1999.

A number of books have been written in the last several years on the topic of WLANs. Why is it necessary to bring another one to your shelves? We believe that, with the advent of the IEEE 802.11 standard for WLANs, the consolidation of the WLAN market will commence. Therefore, it is important that WLAN designers, network planners and administrators, and users understand the operation and application of IEEE 802.11. This handbook will provide the detail required to attain that understanding....

There is a huge amount of information in the IEEE 802.11 standard and its extensions. Finding the information required in a short time can be challenging. To help meet the challenge, a mapping between the information in the standards and that presented in this handbook is given here. IEEE standards are divided into clauses and annexes. Information in the standard is referred to by the clause and annex in which it is found. This book is divided into chapters. Information in this book is referred to by the chapter in which it is found.

Clauses 1 through 4 of the standard contain a brief overview of the standard.... This information corresponds to the Introduction and abbreviations in this handbook.

Clause 5 of the standard provides a description of the architecture and components of an IEEE 802.11 WLAN system. This corresponds to Chapter 2 in this handbook.

Clause 6 of the IEEE 802.11 standard describes the MAC service interface.... This is not described explicitly in this handbook.

Clause 7 of the standard describes the MAC frames and their content. Clause 8 of the standard describes the WEP functionality that may be implemented in an IEEE 802.11 station. Clause 9 describes the functionality and frame exchange protocols of the MAC. Information from these clauses is found in Chapter 3.

Clause 10 describes the layer management service interface primitives and their functionality. Clause 11 describes the MAC management functionality and protocols. This information may be found in Chapter 4.

Clause 12 describes the PHY service interface. This is an abstract interface for the exchange of data between the MAC and PHY. Clause 13 describes the PHY management service interface, which consists solely of the MIB interface. This is not described explicitly in this handbook.

Clause 14 describes the frequency hopping spread spectrum physical layer. Clause 15 describes the direct sequence spread spectrum physical layer. Clause 16 describes the infrared baseband physical layer. Clause 17 (IEEE 802.11a) describes the orthogonal frequency division multiplexed physical layer. Clause 18 (IEEE 802.11b) describes the higher rate direct sequence spread spectrum physical layer. Information on all physical layers is found in Chapter 6....

[Olov1995] Tomas Olovsson, Erland Jonsson, Sarah Brocklehurst, and Bev Littlewood. *Towards Operational Measures of Computer Security: Experimentation and Modelling*, chapter 8B, pages 555–569,571–572. Basic Research Series. Springer Verilag Berlin Heidelberg New York, 1995. B. Randell J. - C. Laprie H. Kopetz B.

Littlewood (Eds.).

The two experiments described here were intended to investigate the empirical issues that arise from the probabilistic view of security assessment discussed in the previous paper [Litt1995]. Specifically, they investigated the problems of measuring *effort* and *reward* associated with security attacks and breaches.

[Pall1999] Alan Paller. CyberTerrorism: A Reality Check and Essential Security Actions. Director of Research, The SANS Institute paller@sans.org, www.sans.org, Fall 1999.

Presentation given at Shadowcon 1999 in Dahlgren, VA. It overviews four myths of intruders: (1) only the world wide web gets attacked, (2) no harm is done, (3) need technical expertise, and (4) attacks can't threaten the United States. And it shows how each fails. He references *At Large* [Free1997] in saying that an attacker got root access to computers that controlled every dam in the northern part of the state [of California]. In addition, the fictitious cyberwar in [Arqu1998] is analyzed with the NSA's Indicator and Warnings (I&W) assessment.

[Park1973] Donn B. Parker, Susan Nycum, and S. Stephen Oüra. Computer Abuse. final report PB-231 320, Stanford Research Institute, 1973. Distributed by National Technical Information Service, U.S. Department of Commerce.

An appendix may contain some of the more valuable information in this report — summaries of 148 cases of reported computer abuse that form the basis for a continuing study by Stanford Research institute. Proceedings of the first national invitational conference on computer abuse reported in another appendix provides a candid view and reaction to the research by a group with wide ranging backgrounds from an ex-computer criminal to a noted EDP auditor. The conclusions of this group strongly influenced the conclusions derived from the study.

The report is the second of a planned series of papers on computer abuse. The first was a final report done for Project RISOS at Lawrence Radiation Laboratory on "Threats to Computers". This second report provides more generalized views of computer abuse — technical, legal, and socialogical perspectives — and is the first attempt to document and define the problem based on a typology of reported cases and investigation in detail of several of them. The purpose of the report is to alert business and government users of computers and the technological and sociological research community of the seriousness, extent, and potential of computer abuse as a new and emerging serious social and technological problem.

Computer abuse is defined as any act associated with computers where victims have suffered or could have suffered a loss and perpetrators made or could have made gain. The antisocial nature to computer abuse, as yet not clearly perceived, requires further study and attention.

[Park1975a] Donn B. Parker. Computer Abuse Assessment. Technical Report SRI Project 5068, Volume 1, Grant MCS76-09183, Stanford Research Institute, Menlo Park, California 94025, December 1975. Prepared for National Science Foundation.

Computer abuse is an emerging problem on intentional computer related acts resulting in losses to victims. It includes white-collar crime, espionage and sabotage. The problem is defined and described in terms of data obtained from 381 recorded cases. A typology is presented and assessments made relative to the nature and growth of white-collar crime and the proliferation of computers into environments where white-collar crime has traditionally flourished.

[Park1975b] Donn B. Parker. Computer Abuse Perpetrators and Vulnerabilities of Computer Systems. Technical Report Vol 2 / 6, Stanford Research Institute, Menlo Park, California 94025, December 1975. Prepared for National Science Foundation.

Analysis of computer abuse experience is valuable in threat and risk studies performed to develop appropriate safeguards in computer use. A profile of computer abuse perpetrators has been developed on the basis of interviews with 17 offenders involved in a total of 15 cases. Common characteristics, occupations, and modus operandi are documented and analyzed. Computer systems' and user organizations' vulnerabilities that facilitated perpetrators' actions are also described, based on study of 375 reported cases of abuse. Eight main vulnerable functions and nine main vulnerable functional locations are identified and ranked by incidence of occurrence. Each vulnerability is described by examples in the form of brief case descriptions. Finally, priorities for safeguards are deduced from the results of the study.

[Park1976] Donn B. Parker. *Crime by Computer*. Charles Scribner's Sons, New York, 1976.

Our society is fast becoming dependent on the correct, reliable, and near-continuous operation by electronic data processing (EDP) personnel of digital computers and data telecommunications. As the sensitive functions of computers and the people who run them proliferate in society, the potential grows for serious and even catastrophic losses involving failures in computer systems, caused, of course, by people. If sound decisions concerning the safe use of computers are to be made, more people must have a deeper understanding of the nature of EDP [Electronic Data Processing] technologists, their computers, and data telecommunications and of how such technologies can be misused. This understanding must transcend the giant electronic brain image of computers as depicted in the media and must be founded, instead, on a solid knowledge of both the theory and the practice involved. A view of how safe and unsafe these powerful tools in the hands of our technologists can be is presented in this book, based mostly on actual losses experienced from the aspect most difficult to deal with — intentionally perpetrated acts. This view is presented at an appropriate level for all those people who come in contact with computers, those who are affected by the use of computers, and those who think seriously about the current role of computers.... The presentation is organized into roughly three parts: a discussion about computers and computer abuse, case studies of intentional acts involving computers, and finally a discussion of what can be done and what may happen.

[Park1983] Donn B. Parker. *Fighting Computer Crime*. Charles Scribner's Sons, New York, 1983.

Considering the power and leverage of computers, the dependence on them, and their increasing role in society, it is not surprising to see attention focusing on computer crime, even though it basically consists of tranditional types of crime committed by people, of course, and not computers.... My first book, *Crime by Computer* (1976), was the two-by-four applied to the head to direct everybody's attention to a new, emerging problem. It was heavy with horror stories and light on remedies. *Fighting Computer Crime* is an intermediate book, still heavy on loss experience, including some horror stories, but presented in a more organized and categorized fashion and, most important, heavy on remedies derived and discovered mostly the past six years.... This book concludes with a consideration of the future escalation possibilities of computer crime, starting with the SARK Report from Sweden and including the potential of computer security to reduce the risks. The book also ends with the insight that computer crime is a "people" problem, not just a technological one, and that it can be licked by applying already known safeguards and practices as long as the problem is sufficiently understood and correctly anticipated.

[Park1984] Donn B. Parker. The Many Faces of Data Vulnerability. *IEEE Spectrum*, 21(5):46–49, May 1984.

Information can be tapped when resident in memory. It can also be intercepted en route, on paper, and elsewhere....

With the advent of the information age, the potential threats and losses to human beings and property have not necessarily increased or decreased. They have simply changed. One big change is that communications today encompasses not only interchanges between people but also between people and machines and between one machine and another. Therefore, security measures have to be changed as well.

[Park1988] Stephen K. Park and Keith W. Miller. Random Number Generators: Good Ones Are Hard To Find. *Communications of the ACM*, 31(10):1192–1201, October 1988.

Practical and theoretical issues are presented concerning the design, implementation, and use of a good, minimal standard random number generator that will port to virtually all systems.

[Park1989] Donn B. Parker. *COMPUTER CRIME Criminal Justice Resource Manual*. U.S. Department of Justice **National Institute of Justice** *Office of Justice Programs*, August 1989. Prepared by SRI International under contract to Abt Associates for National Institute of Justice, U.S. Department of Justice, contract #OJP-86-C-002.

The original *Criminal Justice Resource Manual on Computer Crime* was written at SRI International by Donn B. Parker and Susan Nycum in 1979 for the U.S. Department of Justice, Bureau of Justice Statistics. This revision of the manual reflects the extensive technical and statutory changes as well as computer crime loss experience that have occurred over the last 10 years. In that time, computer crime has become a mature subject of interest to a criminal justice community that must cope with 48 state and two federal statutes defining computer crime offenses.

The manual is written as both a training and reference guide for prosecutors and investigators who know only a little about computer technology as well as those with extensive technical knowledge. For lay persons, this manual provides guidelines for determining when technical and criminal justice expertise should be used and how to interact with the people who provide it. Investigators or prosecutors experienced in computer technology will find much information that will assist them in dealing with even the most sophisticated of computer crimes. Overall, then, the manual presents a simple, straightforward means of successfully prosecuting suspected computer crime perpetrators and the associated technical context....

[Park1991] Donn B. Parker. Restating the Foundation of Information Security. In *14th National Computer Security Conference*, volume 14, pages 480–493, October 1991.

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 system 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.

[Park1992] Donn [B.] Parker. *Computer Security Reference Book*, chapter 34, Computer Crime, pages 437–476. CRC Press, K.M. Jackson and J. Hruskh, U.S. Associate Editor Donn B. Parker, Boca Raton, Florida, 1992.

Business, economic, and white-collar crimes have rapidly changed as computers proliferated into the activities and environments in which these crimes occur. Computers have engendered a different form of crime even though they are called by familiar names of fraud, embezzlement, larceny, espionage, and so forth. The evolution of occupations in this field has extended the traditional categories of criminals to include computer programmers,

computer operators, tape librarians, and electronic engineers who function in new environments. Although crime has traditionally occurred in ordinary human environments, some crime is now perpetrated using personal computers in bedrooms, or mainframe computers in the specialized environment of rooms with raised flooring, lowered ceilings, large grey boxes, flashing lights, moving tapes, and the hum of air-conditioning motors. The methods of committing crime have changed. A new jargon has developed, identifying automated criminal methods such as data diddling, Trojan horses, logic bombs, salami techniques, superzapping, piggy backing, scavenging, data leakage, and asynchronous attacks (see later sections). The forms of many of the targets of computer crime are also different. Electronic transactions and money, as well as paper and plastic money (credit cards), represent assets subject to intentionally caused, automated loss. Money in the form of electronic signals and magnetic patterns is stored and processed in computers and transmitted over telephone lines. Money is debited and credited to accounts inside computers. In fact, the computer has become the vault for the business community. Many other physical assets, including inventories of products in warehouses and of materials leaving or entering factories, are represented by electronic and optical documetns of record inside computer systems. Electronic data interchange (EDI), which connects trading partners for conducting contract negotiations, sales, invoicing, and collections, focus traditional sources of business crime on computers and data communications.

[Paul2000] Linda Dailey Paulson. Exploring the Wireless LANscape. *Computer*, 33(10):12–16, October 2000.

Traditionally, the work "networking" has evoked images of yards of spaghetti-like wiring in walls, on floors, and hanging from the backs of computers and peripherals.

However, several trends in the computer industry are rapidly driving the development and adoption of newer wireless networking technologies, which link devices to each other and to corporate LANs, primarily via radio-frequency (RF) technology....

[Perk1998] Charles E. Perkins. *Mobile IP: Design Principels and Practices*. Addison-Wesley Wireless Communications Series. Addison-Wesley, One Jacob Way / Reading, Massachusetts 01867, 1998.

This book introduces the TCP/IP-savvy reader to the design and implementation of Internet Protocols useful for maintaining network connections while moving from place to place. It describes the technology that make mobile networking possible; in particular, it focuses on Mobile IP, the Internet Engineering Task Force (IETF) Standard for mobile networking. Written by Charles E. Perkins, a leader in the mobile networking field, this book discusses:

- Mobile IP
- Route optimizaiton
- IP version 6
- Use of Dynamic Host Configuration Protocol (DHCP)
- Encapsulation

After reading *Mobile IP*, a network engineer will be able to produce implementations of Mobile IP for mobile node, foreign agents, and home agents. As with any Internet protocol, Mobile IP requires precise handling of packetized control data; all of the needed steps for handling that data are detailed fully in this book. The necessary control mechanisms for processing advertisements (perhaps received over wireless media) are given first, followed by the main part of the Mobile IP protocol, which addresses how the mobile node registers its current IP attachment information with the support infrastructure on its home network. This book also details how Mobile IP specifies the handling of data packets destined form the mobile node.

In addition to the base protocol, this book also presents newly specified enhancements to Mobile IP, and details the protocol support needed for enabling mobile networks using IPv6, the new version of IP with 128-bit addresses. Interactions between Mobile IP and other protocols (such as DHCP) are described, enabling network engineers to get a complete understanding of the system effects of deploying Mobile IP in enterprise networks. Other current trends in protocol development relevant to Mobile IP are also described, such as ways to reduce registration traffic with the home network, and first steps toward integrating Mobile IP with enterprise security installations such as firewalls and border routers.

[Perr1984] Tekla S. Perry and Paul Wallich. Can Computer Crime Be Stopped? *IEEE Spectrum*, 21(5):34–45, May 1984.

The proliferation of microcomputers in today's information society has brought with it new problems in protecting both computer systems and their resident intelligence....

Computer crime, by its nature, is thought to be frequently undetected and underreported. A skilled computer criminal often can leave no evidence — information can be copied instead of removed, records of entry can be erased. A victim whose business depends on its reputation of trustworthiness — such as a bank or an insurance company — is unlikely to publicize the fact that its records were tampered with and the criminal got away. Clearly reported computer crimes are probably only the tip of the iceberg — but just how big the iceberg is no one knows.

[Pete1995] Ivars Peterson. *Fatal Defect: Chasing Killer Computer Bugs.* Time Books, subsidiary of Random House, New York, 1995.

Despite concerted efforts to prevent malfunctions and aliminate defects, problems continue to surface. Moreover, as computer designers and software engineers constructincreasingly complicated systems, their chances of eradicating all possible bugs shrink to zero....

The steadily increasing speed of computers and the growing complexity of computer systems and networks make flaws ever more difficult to track down. Frequently, problems occur in environments where there are so many things happening simultaneously that by the time an error is detected, one no longer knows where or when it happened....

Taking on ever greater responsibilities, computer systems also seem to be edging beyond human control and understanding. Designed to help us cope with complexity, the systems themselves are becoming too complicated for us to grasp in their entirety. This trend bodes ill for a future that could include unmanned oil tankers and other automated vehicles, automatically controled, "smart" homes and office buildings, and the vast worldwide web of computers and communications equipment that is to serve as the information superhighway.

The fact that we can never be sure that a computer system will function flawlessly constitutes a fatal defect. It limits what we can hope to achieve by using computers as our servants and surrogates. As computer-controlled systems become more complex and thoroughly entwined in the fabric of our live,s their potential for costly, life-threatening failures keeps growing. Are we courting disaster by placing too much trust in computers to handle complexities that no one fully understands?

[Pfle1997] Charles P. Pfleeger. *Security in Computing.* Prentice Hall, Second edition, 1997.

Every day, more and more critical information is created, transmitted, and archived by computers. This ever-growing reliance on technology has made computer security a higher priority than ever before, yet the pace of computer development has far outstripped the improvements in computer security. Today's computer professionals need a comprehensive understanding of all aspects of security in computing.

Security in Computing is the most complete and up-to-date college textbook now available. Enlivened by actual case studies and supported by more than 175 exercises, the book covers: Viruses, worms, Trojan horses, and other forms of malicious code. Firewalls and the protection of networked system. E-mail privacy, including PEM, PGP, key management, and certificates. Key escrow — both as a technology and in the "Clipper" program. Evaluation of trusted systems, including the Common Criteria, the ITSEC, and the OrangeBook. Standards for program development and quality, including ISO9000 and SEI CMM. Administering secure installations of Pcs, UNIX, and networked environments. Ethical and legal issues in computing.

[Pipk1997] Donald L. Pipkin. *Halting the Hacker: A Practical Guide to Computer Security.* Hewlett-Packard Professional Books. Prentice Hall, 1997.

When it comes to computer security, you livelihood and your company's future are on the line. It's not enough to simply follow a security "cookbook": you need to get into the mind of your adversary, the hacker. In *Halting the Hacker*, a leading Fortune 500 security consultant shows you the approaches and techniques hackers use to gain access, privileges, and control of your UNIX system. You'll learn to look at your system the way a hacker does, identifying potential vulnerabilities. You'll learn what specific countermeasures to take

now. Even more important, you'll learn how to recognize and respond to future security concerns — before they become catastrophes. You'll discover

- How hackers transform minor oversights into major security breaches.
- How hackers cover their tracks while leaving "back doors" into your system.
- How to protect your system against disgruntles or dishonest insiders.
- How to detect break-ins — and what to do next....

[Post1981a] Jon Postel. Internet Protocol: DARPA Internet Program Protocol Specification. Request for Comments (RFC) 791, September 1981. Internet Engineering Task Force (IETF) http://www.ietf.org.

This document specifies the DoD Standard Internet Protocol. This document is based on six earlier editions of the ARPA Internet Protocol Specification, and the present text draws heavily from them. There have been many contributors to this work both in terms of concepts and in terms of text. This edition revises aspects of addressing, error handling, option codes, and the security, precedence, compartments, and handling restriction features of the internet protocol.

[Post1981b] Jon Postel. Transmission Control Protocol: DARPA Internet Program Protocol Specification. Request for Comments (RFC) 793, September 1981. Internet Engineering Task Force; http;//www.ietf.org.

This document describes the DoD Standard Transmission Control Protocol (TCP). There have been nine earlier editions of the ARPA TCP specification on which this standard is based, and the present text draws heavily from them. There have been many contributors to this work both in terms of concepts and in terms of text. This edition clarifies several details and removes the end-of-letter buffer-size adjustments, and redescribes the letter mechanism as a push function.

[Post1988] J[on] Postel and J. Reynolds. A Standard for the Transmission of IP Datagrams over IEEE 802 Networks. Request for Comments (RFC) 1042, February 1988. Internet Engineering Task Force (IETF); http://www.ietf.org.

The goal of this specification is to allow compatible and interoperable implementations for transmitting IP datagrams and ARP requests and replies. To achieve this it may be necessary in a few cases to limit the use that IP and ARP make of the capabilities of a particular IEEE 802 standard. The IEEE 802 specifications define a family of standards for Local Area Networks (LANs) that deal with the Physical and Data Link Layers as defined by the ISO Open System Interconnection Reference Model (ISO/OSI). Several Physical Layer standards (802.3, 802.4, and 802.5)[28][29][30] and one Data Link Layer Standard (802.2)[31] have been defined. The IEEE Physical Layer standards specify the ISO/OSI Physical Layer and the Media Access Control Sublayer of the ISO/OSI Data Link Layer. The 802.2 Data Link Layer standard specifies the Logical Link Control Sublayer of the ISO/OSI Data Link Layer. This memo describes the use of IP and ARP on the three types of networks. At this time, it is not necessary that the use of IP and ARP be consistent across all three types of networks,

[28]IEEE, "IEEE Standards for Local Area Networks: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", IEEE, New York, New York, 1985.

[29]IEEE, "IEEE Standards for Local Area Networks: Token-Passing Bus Access Method and Physical Layer Specification", IEEE, New York, New York, 1985.

[30]IEEE, "IEEE Standards for Local Area Networks: Token Ring Access Method and Physical Layer Specifications", IEEE, New York, New York, 1985.

[31]IEEE, "IEEE Standards for Local Area Networks: Logical Link Control", IEEE, New York, New York, 1985.

only that it be consistent within each type. This may change in the future as new IEEE 802 standards are defined and the existing standards are revised allowing for interoperability at the Data Link Layer. It is the goal of this memo to specify enough about the use of IP and ARP on each type of network to ensure that:

1. all equipment using IP or ARP on 802.3 networks will interoperate,
2. all equipment using IP or ARP on 802.4 networks will interoperate,
3. all equipment using IP or ARP on 802.5 networks will interoperate.

Of course, the goal of IP is interoperability between computers attached to different networks, when those networks are interconnected via an IP gateway[32]. The use of IEEE 802.1 compatible Transparent Bridges to allow interoperability across different networks is not fully described pending completion of that standard.

[Poul2001] Kevin L. Poulsen. War Driving by the Bay. Presently only published on the Internet at: http://www.securityfocus.com/templates/article.html?id=192 and at http://www.theregister.co.uk/content/8/18285.html., April 12 2001.

In a parking garage across from Moscone Center, the site of this year's RSA Conference, Peter Shipley reaches up though the sunroof of his car and slaps a dorsal-shaped Lucent antenna to the roof — where it's held firm by a heavy magnet epoxied to the base.... "The important part of getting this to work is having the external antenna. It makes all the difference" says Shipley, snaking a cable into the car and plugging it into the wireless network card slotted into his laptop. The computer is already connected to a GPS receiver — with its own mag-mount roof antenna — and the whole apparatus is drawing juice through an octopus of cigarette-lighter adapters. He starts some custom software on the laptop, starts the car and rolls out. Shipley, a computer security researcher and consultant, is demonstrating what many at the security super-conference are quietly describing as the next big thing in hacking. It doesn't take long to produce results. The moment he pulls out of the parking garage, the laptop displays the name of a wireless network operating within one of the anonymous downtown office buildings: "SOMA AirNet." Shipley's custom software passively logs the latitude and longitude, the signal strength, the network name and other vital stats. Seconds later another network appears, then another: "addwater," "wilson," "tangentfund." After fifteen minutes, Shipley's black Saturn has crawled through twelve blocks of rush hour traffic, and his jerry-rigged wireless hacking setup has discovered seventeen networks beaconing their location to the world. After an hour, the number is close to eighty. "These companies probably spend thousands of dollars on firewalls," says Shipley. "And they're wide open...."

[Ptac1998] Thomas H. Ptacek and Timothy N. Newsham. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Technical report, Secure Networks, January 1998.

All currently available network intrusion detection (ID) systems rely upon a mechanism of data collection — passive protocol analysis — which is fundamentally flawed. In passive protocol analysis, the intrusion detection system (IDS) unobtrusively watches all traffic on the network, and scrutinizes it for patterns of suspicious activity. We outline in this paper two basic problems with the reliability of passive protocol analysis: (1) there isn't enough information on the wire on which to base conclusions about what is actually happening on networked machines, and (2) the fact that the system is passive makes it inherently "fail-open," meaning that a compromise in the availability of the IDS doesn't compromise the availability of the network. We define three classes of attacks which exploit these fundamental problems — insertion, evasion, and denial of service attacks — and describe how to apply these three types of attacks to IP and TCP protocol analysis. We present the results of tests of the efficacy of our attacks against four of the most popular network intrusion detection systems on the market. All of the ID systems tested were found to be vulnerable to each of our attacks. This indicates that network ID systems cannot be fully trusted until they are fundamentally redesigned.

[Rapp1996] Theodore S. Rappaport. *Wireless Communications: Principles & Practice*. Prentice Hall PTR, One Lake Street, Upper Saddle River, NJ 07458, 1996.

[32]Braden, R., and J. Postel, "Requirements for Internet Gateways", RFC-1009, USC/Information Sciences Institute, June 1987

As cellular telephones become commonplace business tools, interest in wireless technology is booming. This book responds to that demand with a comprehensive survey of the field, suitable for educational or technical use. Materials are drawn for academic and business sources, numerous journals, and an IEEE professional reader. Extensively illustrated, **Wireless Communications** is filled with examples and problems, solved step by step and clearly explained.

Wireless Communications covers the design fundamentals of cellular systems, including issues of frequency reuse, channel assignments, radio propagation, and both analog and digital modulation techniques. Speech coding, channel coding, diversity, spread spectrum, and multiple access are also discussed. A separate chapter is devoted to wireless networking, including SS7 and ISDN.

Beyond theory, **Wireless Communications** offers practical reference sections, including:

- Complete technical standards for cellular, cordless telephone, and personal communications systems
- International standards for Europe, the Americas, and the Asia-Pacific region
- Noise figure calculations and Gaussian approximations of spread spectrum CDMA interference
- Mathematical tables, identities, and the Q, erf, and erfc functions
- Glossary of abbreviations and acronyms
- Full list of references

This book is designed for use in graduate and undergraduate classrooms, but is also suitable for use by professional engineers and technicians. It can be used for both teaching and reference, and is also appropriate for the interesting cellular phone consumer who wants to understand the technology.

[Raym1996] Eric S. Raymond, editor. *The New Hacker's Dictionary.* The MIT Press, Cambridge, Massachusetts, London England, third edition, 1996. with forward and cartoons by Guy L. Steele Jr.

This document is a collection of slang terms used by various subcultures of computer hackers. Though some technical material is included for background and flavor, it is not a technical dictionary; what we describe here is the language hackers use among themselves for fun, social communication, and technical debate....

This new edition of the hackers' own phenomenally successful lexicon includes more than 100 new entries and updates or revises 200 more. Historically and etymologically richer than its predecessor, it supplies additional background on existing entries and clarifies the murky origins of several important jargon terms (overturning a few long-standing folk etymologies), while still retaining its high giggle value.

[Rich1999] Tom Richardson, Jim Davis, Doug Jacobson, John Dickerson, and Laura Elkin. Developing a Database of Vulnerabilities to Support the Study of Denial of Service Attacks. IEEE Symposium on Security and Privacy 5-minute presentation, May 1999. http://www.issl.org.

Iowa State is taking Krsul's taxonomy and expanded it to include the following three categories of attack: *specification weakness*, *implementation weakness*, and *brute force attacks*. They are taking attacks from rootshell.org and plotting them into a database using Krsul's categories. They hope to look for clustering and develop countermeasures for clustered exploits. Some early results were presented.

[Rich2001] Thomas Winfred Richardson. *The Development of a Database Taxonomy of Vulnerabilities to Support the Study of Denial of Service Attacks.* PhD thesis, Iowa State University, 2001.

As computer networks continue to proliferate, the world's dependence on a secure communication infrastructure is of prime importance. Disruption of service through Denial of Service (DoS) attacks can result in great financial loss for Internet-based companies and major inconveniences for users of Internet services. The purpose of this two-year study was to study and understand network denial of service attacks so that methods

may be developed to detect and prevent them.

Initially, the researcher constructed a database of system and network exploits that revealed the underlying vulnerabilities in the software or protocols they attack. The database was populated with exploits posted at popular reporting sites such as Rootshell, Bugtraq, Security Focus. To encourage the use of a common vulnerability taxonomy and to facilitate sharing of data, parts of the classification scheme proposed by Krsul (1998) in his research were included, as well as developing a taxonomy tree based on the current research. Sifting through the reports and categorizing the attacks has been a challenging experience; and creating categories that are unambiguous, repeatable, and exhaustive has proven to be a difficult task. The results were two to three methods of classification that are useful for developing categories of vulnerabilities.

The next phase of the project was to look for any clustering of attacks based on these vulnerability categories, and to determine if effective countermeasures can be deployed against them. Although past history is no guarantee of future exploit activity, it is hoped that the countermeasures proposed based on these 630 exploits will remain valid for future DoS attacks. Toward this goal, the research made use of data mining software packages to plot the various categories of attacks so that the interrelationships could be more easily discovered and studied. A sampling of the database plots, an interpretation of the plotted data, and the countermeasures proposed for the vulnerability categories developed as part of the database creation are presented in this research.

[Riez2000] Michael J. Riezenman. Cellular Security: Better, but Foes Still Lurk. *IEEE Spectrum*, 37(6):39–42, June 2000.

Service providers have largely solved the cloning problem, but eavesdropping is still an issue, and e-commerce has barely been addressed....

[Rist1988] Marlin P. Ristenbatt. Methodology for Network Communication Vulnerability Analysis. In *MILCOM: 21st Century Military Communications — What's Possible*, pages 0493–0499, 1988.

The framework for a developing network vulnerability assessment methodology is described. An original network taxonomy is used to both orient the analyst to the new network and to initially assess the potential vulnerability issues. Four modules (similar to DVAL)[33] are used: susceptibility; interceptibility; accessibility; and feasibility. Susceptibility issues are pursued in terms of three perspectives: topology; communication protocols; and management and control. The concept of layers, as in the OSI reference model, is used for pursuing the protocol susceptibilities. Security of network control data, to deny spoofing, is included.

[Robe1975] Lawrence G. Roberts. Aloha Packet System with and without Slots and Capture. *Computer Communications Review*, 5(2):28–42, April 1975.

Editor's note: This paper was originally distributed informally as ARPA Satellite System Note 8 on June 26, 1972. The paper is an important one and since its initial limited distribution, the paper has been frequently referenced in the open literature, but the paper itself has been unavailable in the open literature. Publication here is meant to correct the previous gap in the literature.

As the paper was originally distributed only o other researchers intimately familiar with the area covered by the paper, the paper makes few concessions to the reader along the lines of introductory or tutorial material. Therefore, a bit of background material follows.

ALOHA packet systems were originally described by Abramson ("The ALOHA System — Another Alternative for Computer Communication," Proceedings of the AFIPS Fall Joint Computer Conference, Vol.,37, 1970, p.,281–285).[34] In an ALOHA a single broadcast channel is shared by a number of communicating devices.

[33]Data Link Vulnerability Analysis; DVAL Methodology, DVAL Joint Task Force (JTF), Kirkland AFB, New Mexico, November 1984, Volume 1: Susceptibility Module, Susceptibility Handbook; Volume 2: Interceptibility Module, Interceptibility Handbook; Volume 3: Accessibility Module, Accessibility Handbook; Volume 4: Feasibility Module, Feasibility Handbook.

[34][Abra1970].

In the version originally described by Abramson, every device transmits its packets independent of any other device or any specific time. That is, the device transmits the whole packet at a random point in time; the device then times out for receiving an acknowledgment. If an acknowledgment is not received, it is assumed that a collision occurred with a packet transmitted by some other device and the packet is retransmitted after a random additional waiting time (to avoid repeated collisions). Under a certain set of assumptions, Abramson showed that the effective capacity of such a channel is 1/(2e).

Roberts in the present paper investigates methods of increasing the effective channel capacity of such a channel. One method he proposes to gain in capacity is to consider the channel to be slotted into segments of time whose duration is equal to the packet transmission time, and to require the devices to begin a packet transmission at the beginning of a time slot. Another method Roberts proposes to gain in capacity is to take advantage of the fact that even through packets from two devices collide in the channel (i.e., they are transmitted so they pass through the channel at overlapping times), it may be possible for the receive(s) to "capture" the signal of one of the transmitters, and thus correctly receive one of the conflicting packets, if one of the transmitters has a sufficiently greater signal than the other. Roberts considres the cases of both satellite and ground radio channels.

(Some of the text for the above background material was abstracted from "On the Capacity of Slotted ALOHA Networks and Some Design Problems," Israel Gitman, IEEE Transactions on Communications, Vol. COM-23, No.,3, March 1975.)

[Roch1989] Jon A. Rochlis and Mark W. Eichin. With Microscope and Tweezers: The Worm from MIT's Perspective. *Communications of the ACM*, 32(6):689–698, June 1989.

The actions taken by a group of computer scientists at MIT during the worm invasion represents a study of human response to a crisis. The authors also relate the experiences and reactions of other groups throughout the country, especially in terms of how they interacted with the MIT team.

[Rose1981a] Eric C. Rosen. Vulnerabilities of Network Control Protocols: An Example. *ACM SIGSOFT, Software Engineering Notes*, 6(1):6–8, January 1981. Appears to be identical to [Rose1981b].

On October 27, 1980, there was an unusual occurrence on the ARPANET. For a period fo several hours, the network appeared to be unusable, due to what was later diagnosed as a high priority software process running out of control. Network-wide disturbances are extremely unusual in the ARPANET (non has occurred in several years), and as a result, many people have expressed interest in learning more about the etiology of this particular incident. The purpose of this note is to explain what the symptoms of the problem were, what the underlying causes were, and what lessons can be drawn. As we shall see, the immediate cause of the problem was a rather freakish hardware malfunction (which is not likely to recur) which caused a faulty sequence of network control packets to be generated. This faulty sequence of control packets in turn affected the apportionment of software resources in the IMPs, causing one of the IMP processes to use an excessive amount of resources, to the detriment of other IMP processes. Restoring the network to operational condition was a relatively straightforward task. There was no damage other than the outage itself, and no residual problems once the network was restored. Nevertheless, it is quite interesting to see the way in which unusual (indeed, unique) circumstances can bring out vulnerabilities in network control protocols, and that shall be the focus of this paper.

[Rose1981b] Eric C. Rosen. Vulnerabilities of Network Control Protocols: An Example. *SIGCOMM Computer Communication Review*, 11(3):10–16, July 1981. Appears to be identical to [Rose1981a].

See annotation in [Rose1981a].

[Rush1993] John Rushby. Critical System Properties: Survey and Taxonomy. Technical Report CSL-93-01, Computer Science Laboratory / SRI International, Menlo Park, CA 94025, May 1993. Under contract through NASA: NAS1-18969 and Naval Research Laboratory: N00014-92-C-2177.

Computer systems are increasingly employed in circumstances where their failure (or even their correct operation, if they are built to flawed requirements) can have serious consequences.

There is a surprising diversity of opinion concerning the properties that such "critical systems" should possess, and the best methods to develop them. The *dependability* approach grew out of the tradition of ultra-reliable and fault-tolerant systems, while the *safety* approach grew out of the tradition of hazard analysis and system safety engineering. Yet another tradition is found in the *security* community, and there are further specialized approaches in the tradition of *real-time* systems. In this report, I examine the critical properties considered in each approach, and the techniques that have been developed to specify them and to ensure their satisfaction.

Since systems are now being constructed that must satisfy several of these critical system properties simultaneously, there is particular interest in the extent to which techniques from one tradition support or conflict with those of another, and in whether certain critical system properties are fundamentally compatible or incompatible with each other. As a step toward improved understanding of these issues, I suggest a taxonomy, based on Perrow's analysis[35], that considers the complexity of component interactions and tightness of coupling as primary factors.

[Salt1975] Jerome H. Saltzer and Michael D. Schroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, 63(9):1278–1308, September 1975.

This tutorial paper explores the mechanics of protecting computer-stored information from unauthorized use or modification. It concentrates on those architectural structures — whether hardware or software — that are necessary to support information protection. The paper develops in three main sections. Section I describes desired functions, design principles, and examples of elementary protection and authentication mechanisms. Any reader familiar with computers should find the first section to be reasonably accessible. Section II requires some familiarity with descriptor-based computer architecture. It examines in depth the principles of modern protection architectures and the relation between capability systems and access control list systems, and ends with a brief analysis of protected subsystems and protected objects. The reader who is dismayed by either the prerequisites or the level of detail in the second section may wish to skip to Section III, which reviews the state of the art and current research projects and provides suggestions for further reading.

[Sava1999] Stefan Savage, Neal Cardwell, David Wetherall, and Tom Anderson. TCP Congestion Control with a Misbehaving Receiver. *ACM SIGCOMM Computer Communication Review*, 29(5):71–78, October 1999.

In this paper, we explore the operation of TCP congestion control when the receiver can misbehave, as might occur with a greedy Web client. We first demonstrate that there are simple attacks that allow a misbehaving receiver to drive a standard TCP sender arbitrarily fast, without losing end-to-end reliability. These attacks are widely applicable because they stem from the sender behavior specified in RFC2581 rather than implementation bugs. We then show that it is possible to modify TCP to eliminate this undesirable behavior entirely, without requiring assumptions of any kind about receiver behavior. This is a strong result: with out solution a receiver can only *reduce* the data transfer rate by misbehaving, thereby eliminating the incentive to do so.

[35] C. Perrow. *Normal Accidents: Living with High Risk Technologies*. Basic Books, New York, NY, 1984.

[Sche1979] Lieutenant Colonel Roger R. Schell. Computer Security: the Achilles' Heel of the Electronic Air Force? *Air University Review*, pages 16–33, January–February 1979.

- ....The high *vulnerability* of contemporary computer has been clearly indicated in the author's experience with undetected penetration of security mechanisms. In addition, security weaknesses are documented in both military and civil reports.
- The *capability* of the Soviets (or any other major hostile group) to accomplish the required penetration is quite evident. In fact, no particular skills beyond those of normally competent computer professionals are required.
- The *motivation* for such an infomraiton collection activity is apparent in prima facie evidence. The broad scope and high intensity of Soviet intelligence efforts in areas such as communication interception are frequently reported.
- The potential *damage* from penetration is growing with the ever increasing concentration of sensitive information in computers and the interconnection of these computers into large networks. Through computer penetration an enemy could, for example, compromise plans for employment of tactical fighters or compromise operational plans and targeting for nuclear missiles.
- The *opportunity* for hostile exploitation of these vulnerabilities is increasing markedly both because of the increased use of computers and the lack of a meaningful security policy controlling their use. In the name of efficiency many more people with less (or no) clearance are permitted easier access to classified computer systems.

[Schn1996] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Second edition, 1996.

This new edition of the cryptography classic provides you with a comprehensive survey of modern cryptography. The book details how programmers and electronic communications professionals can use cryptography — the technique of enciphering and deciphering messages — to maintain the privacy of computer data. It describes dozens of cryptography algorithms, gives practical advice on how to implement them into cryptographic software, and shows how they can be used to solve security problems. Covering the latest developments in practical cryptographic techniques, this new edition shows programmers who design computer applications, networks, and storage systems how they can build security into their software and systems.

New information on the Clipper Chip, including ways to defeat the key escrow mechanism. New encryption algorithms, including algorithms from the former Soviet Union and South Africa, and the FC4 stream cipher. The latest protocols for digital signatures, authentication, secure elections, digital cash, and more. More detailed information on key management and cryptographic implementations.

[Schn1997] Bruce Schneier and David Banisar. *The Electronic Privacy Papers: Documents on the Battle for Privacy in the Age of Surveillance*. Wiley Computer Publishing; John Wiley & Sons, Inc., New York, 1997.

A realistic look at the major issues, players, and key strategies in the war over electronic privacy....

Edited by internationally recognized security expert Bruce Schneier and privacy advocate David Banisar, this is the definitive collection of critical and previously classified government and industry documents. It enables you to fully understand government policies and their impact on both individuals and companies involved with the Internet. *The Electronic Privacy Papers* offers readers a close look at regulatory and technical issues, including:

- The economic and political rationale for demanding digital wire tapping and surveillance
- The legal foundations of, and limitations to, government surveillance
- Governement strategies for soliciting cooperation from telephone companies and equipment manufacturers

- Which policies that industries and individuals can expect the government to pursue in the future.

The Electronic Privacy Papers includes excerpts from the House Judiciary Committee report on the digital telephony bill, the final text of the bill, the FBI's wish list for electronic surveillance, U.S. cryptography policy statements from the White House, and many other government documents. *The Electronic Privacy Papers* is *must* reading for anyone involved with public policy and the delivery of online information.

[Schn1998a] Bruce Schneier and Mudge. Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP). In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 132–141, November 1998.

The Point-to-Point Tunneling Protocol (PPTP) is used to secure PPP connections over TCP/IP links. In this paper, we analyze Microsoft's Windows NT implementation of PPTP. We show how to break both the challenge/response authentication protocol (Microsoft CHAP) and the RC4 encryption protocol (MPPE), as well as how to attack the control channel in Microsoft's implementation. These attacks do not necessarily break PPTP, but only Microsoft's implementation of the protocol.

[Schn1998b] Bruce Schneier. Cryptographic Design Vulnerabilities. *Computer*, 31(9):29–33, September 1998.

Strong cryptography is very powerful when it is done right, but it is not a panacea. Focusing on cryptographic algorithms while ignoring other aspects of security is like defending your house not by building a fence around it, but by putting an immense stake in the ground and hoping that your adversary runs right into it. Smart attackers will just go around the algorithms. Counterpane Systems has spent years designing, analyzing, and breaking cryptographic systems. While they do research on published algorithms and protocols, most of their work examines actual products. They've designed and analyzed systems that protect privacy, ensure confidentiality, provide fairness, and facilitate commerce. They've worked with software, stand-alone hardware, and everything in between. They've broken their share of algorithms, but they can almost always find attacks that bypass the algorithms altogether. Counterpane Systems don't have to try every possible key or even find flaws in the algorithms. They exploit errors in design, errors in implementation, and errors in installation. Sometimes they invent a new trick to break a system, but most of the time they exploit the same old mistakes that designers make over and over again. The article conveys some of the lessons this company has learned.

[Schn1999] Bruce Schneier, Mudge, and David Wagner. Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2). In *CQRE, Dusseldorf*, pages 192–203. Springer-Verlag, October 1999. Found on the Internet at: http://www.counterpane.com/pptpv2-paper.html.

The Point-to-Point Tunneling Protocol (PPTP) is used to secure PPP connections over TCP/IP link. In response to [SM98],[36] Microsoft released extensions to the PPTP authentication mechanism (MS-CHAP), called MS-CHAPv2. We present an overview of the changes in the authentication and encryption-key generation portions of MS-CHAPv2, and assess the improvements and remaining weaknesses in Microsoft's PPTP implementation.

[Schn2000] Bruce Schneier. *Secrets & Lies: Digital Security in a Networked World*. Wiley Computer Publishing, 2000.

[36] [Schn1998a].

Welcome to *thebusinessworld.com*. It's digital: Information is more readily accessible than ever. It's inescapably connected: businesses are increasingly — if not totally — dependent on digital communications. But our passion for technology has a price: *increased exposure to security threats*. Companies around the world need to understand the risks associated with doing business electronically. The answer starts here.

Information security expert Bruce Schneier explains what everyone in business needs to know about security in order to survive and be competitive. Pragmatic, interesting, and humorous, Schneier exposes the digital world and realities of our networked society. He examines the entire system, from the reasons for technical insecurities to the minds behind malicious attacks. You'll be guided through the security war zone, and learn how to understand and arm yourself against the threats of our connected world.

There are no quick fixes for digital security. And with the number of security vulnerabilities, breaches, and digital disaster increasing over time, it's vital that you learn how to manage the vulnerabilities and protect our data in this networked world. You need to understand who the attackers are, what they want, and how to deal with the threats they represent. In *Secrets and Lies*, you'll learn about security technologies and product capabilities, as well as their limitations. And you'll find out how to respond given the landscape of your system and the limitations of your business.

With its accessible style, this practical guide covers:

- The digital threats and attacks that you must understand
- The security products and processes currently available
- The limitations of technology
- The steps involved in product testing to discover security flaws
- The technologies to watch for over the next couple of years
- Risk assessment in your company
- The implementation of security policies and countermeasures

Secrets and Lies offers the expert guidance you'll need to make the right choices about securing your digital self.

[Schu1995] E. Eugene Schultz and Thomas A. Longstaff. Internet Sniffer Attacks. In *18th National Information System Security Conference*, October 10–13, 1995.

Shared media networks (i.e., ethernets, FDDI, token ring networks, and so forth) are vulnerable to "sniffer" or "promiscuous monitoring attacks" in which can be captured with authorization at intermediate points during transmission. For well over a year, Internet attackers have used network sniffers to obtain login IDs and passwords to compromise large numbers of Internet capable host machines as well as gateway machines operated by regional Internet service providers. This paper analyzes how these attacks have occurred and discusses the damage that resulted. The attacks are part of a new trend toward use of network mechanisms rather than the more elementary host-based approaches. Whereas the data in the TCP/IP packets have traditionally been the target of promiscuous monitoring attacks, the control information contained in these packets is increasing becoming the target. Furthermore, network intruders are concentrated more on exploiting network mechanisms than on weaknesses in individual systems.

Traditional security measures are no longer adequate to protect against current attack methods. Newer measures, such as using one-time passwords are regularly checking network interfaces to determine whether they are in promiscuous mode, are becoming increasingly necessary.

[Schu1997] Christoph L. Schuba, Ivan V. Krsul, Markus G. Kuhn, Eugene H. Spafford, Aurobindo Sundaram, and Diego Zamboni. Analysis of a Denial of Service Attack on TCP. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 208–223. COAST Laboratory, Department of Computer Sciences, Purdue University, IEEE, 1997.

This paper analyzes a network-based denial of service attack for IP (Internet Protocol) based networks. It is popularly called *SYN flooding*. It works by an attacker sending many TCP *(Transmission Control Protocol)* connection requests with spoofed source addresses to a victim's machine. Each request causes the targeted host

to instantiate data structures out of a limited pool of resources. Once the target host's resources are exhausted, no more incoming TCP connections can be established, thus denying further legitimate access.

The paper contributes a detailed analysis of the SYN flooding attack and a discussion of existing and proposed countermeasures. Furthermore, we introduce a new solution approach, explain its design, and evaluate its performance. Our approach offers protection against SYN flooding for all hosts connected to the same local area network, independent of their operating system or networking stack implementation. It is highly portable, configurable, extensible, and requires neither special hardware, nor modification in routers or protected end systems.

[Schw1996] Winn Schwartau. *Information Warfare: Cyberterrorism: Protecting Your Personal Security in the Electronic Age*. Thunder's Mouth Press, 632 Broadway, 7th Floor New York, NY 10012, second edition, 1996.

Information Warfare costs the United States an estimated $100 to 300$ billion per year through... Industrial Espionage, Hackers and Cyberpunks, Malicious Software and Viruses, Data Eavesdropping, Code Breaking and Chipping, Attacks on Personal Privacy, HERF Guns, EMP/T Bombs and Magnetic Weaponry, Binary Schizophrenia.

[Seel1989a] Donn Seeley. A Tour of the Worm. In *Winter USENIX Conference*, pages 287–304, Department of Computer Science, University of Utah, 1989. USENIX.

On the evening of November 2, 1988, a self-replicating program was released upon the Internet[37]. This program (*a worm*) invaded VAX and Sun-3 computers running versions of Berkeley UNIX, and used their resources to attack still more computers[38][39]. Within the space of hours this program had spread across the U.S., infecting hundreds or thousands of computers and making many of them unusable due to the burden of its activity. This paper provides a chronology for the outbreak and presents a detailed description of the internals of the worm, based on a C version produced by decompiling.

[Seel1989b] Donn Seeley. Password Cracking: A Game of Wits. *Communications of the ACM*, 32(6):700–703, June 1989.

The following report has been gleaned from "A Tour of the Worm," an in-depth account of the November Internet infection. The author found the worm's crypt algorithm a frustrating, yet engaging, puzzle.

[Sena1996] United States Senate. Security in Cyberspace. U.S. Government Printing Office, Hearing before the Permanent Subcommittee on Investigations of the Committee on Governmental Affairs, United States Senate, One Hundred Fourth Congress, Second Session, S. Hrg. 104—701, May 22, June 5, 25, and July 16, 1996.

Prepared Statement of Senator Roth, Chairman. This morning, the Subcommittee will begin the first of a series of hearings on security in cyberspace. This Subcommittee has had a long tradition of investigating emerging threats to our Nation's security. Today we turn to a topic which is perhaps less tangible, but, but

[37] The Internet is a logical network made up of many physical networks, all running the IP class of network protocols.

[38] VAX and Sun-3 are models of computers built by Digital Equipment Corp. and Sun Microsystems Inc, respectively. UNIX is a Registered Bell of AT&T Trademark Laboratories.

[39] "Registered Bell of AT&T Trademark Laboratories" probably had two words (Bell and Trademark) reversed. It should probably read "Registered Trademark of AT&T Bell Laboratories."

just as serious — the security of our computers....

Over the years, we have seen a dramatic evolution in computer technology, but the basic challenge has remained the same: How do we safeguard our valuable information resources and systems.

Today, computers have become essential to the transacting of our Nation's daily business. Everything from telephones to transportation, power networks, our financial system, emergency services, and our national defense depends upon computers. Together, these components, networks, and systems make up the national information infrastructure. Now, more than ever, our military and other critical government personnel rely upon these networks and systems to maintain our national security.

Computer technology has enabled the United States to become the most advanced nation in cyberspace. However, this very strength also makes us uniquely vulnerable....

Unfortunately, mutual trust and cooperation are not enough to ensure that, in this increasingly interconnected world, our computer networks remain safe from unauthorized intruders. With the ever rising number of people connecting to, and "surfing" the Internet, we may soon find ourselves in perilous waters if we do not take precautions to protect our computer networks and the sensitive information they hold....

In order to stop intruders, we need to understand the nature of the threat. Information and intelligence collected form victims of computer intrusions can help both government and private industry understand who these perpetrators are; how they are breaking in; what damage they are causing; and what their motives might be. Whether a hacker is a curious teenage or a foreign spy, cyber trespassing, thievery, and tampering puts the integrity of our data and systems at risk.

As the saying goes: An ounce of prevention is worth a pound of cure. Our information infrastructure is too important to neglect. Defending our computer systems against infiltration is perhaps the most cost-effective way to deal with this problem. By identifying our vulnerabilities now in a controlled environment, we can take precautions to protect this fundamental asset before we suffer a catastrophic and expensive loss. The protection of our computer networks and the information contained in those systems should be of vital concern to all Americans.

[Sena1998] United States Senate. Weak Computer Security in Government: Is the Public at Risk? U.S. Government Printing Office, Hearing before the Committee on Governmental Affairs, United States Senate, One Hundred Fifth Congress, Second Session, S. Hrg. 105—609, May 19, 1998.

Chairman Thompson. The Governmental Affairs Commmittee today is holding the first of a series of hearings on the security of Federal computer systems... It seems that the more technologically advanced we become, the more vulnerable we become. Today's hearings will address the darker side of the information revolution while exploring how we can better protect governmental information....

In today's hearings, we will discuss these challenges and we will hear that the nature of this challenge comes from the fact that our Nation's underlying information infrastructure is riddled with vulnerabilities which represent severe security flaws and severe risk to our Nation's security, public safety, and personal privacy.

While hacker attacks receive much media attention, even more worrisome are the attacks that go unknown. The nature of attacks in the information age seems to allow a malicious individual or group to inflict extensive damage from the comfort and safety of their own home. We must ask whether we are becoming so dependent on communications links and electronic microprocessors that a determined adversary or terrorist could possibly shut down Federal Government operations or damage the economy simply by attacking our computers. At risk are systems that control power distribution and utilities, phones, air traffic, stock exchanges, the Federal Reserve, and taxpayers' credit and medical records.

Unfortunately, government agencies are ill prepared to address this situation. We as a nation cannot wait for the Pearl Harbor of the information age. We must increase our vigilance to attack this problem before we are hit with a surprise attack.

Our witnesses today have substantial knowledge about what the problems really are and can recommend solutions. First, Dr.,Peter Neumann, a recognized private sector expert on computer security, will provide the Committee with an overview of information security issues and testify on the systemic security problems in the government's computer systems.

Then we will hear from L0pht, seven members of a hacker think tank who identify security weaknesses in computer systems in an effort to persuade companies to design more secure systems. L0pht members will testify about specific weaknesses which enable hackers to exploit the Nation's information infrastructure and government information....

[Shoc1982] John F. Shoch and Jon A. Hupp. The "Worm" Programs — Early Experience with a Distributed Computation. *Communications of the ACM*, 25(3):172–180, 1982.

The "worm" programs were an experiment in the development of distributed computations: programs that span machine boundaries and also replicate themselves in idle machines. A "worm" is composed of multiple "segments," each running on a different machine. The underlying worm maintenance mechanisms are responsible for maintaining the worm — finding free machines when needed and replicating the program for each additional segment. These techniques were successfully used to support several real applications, ranging from a simple multimachine test program to a more sophisticated real-time animation system harnessing multiple machines.

[Sieb1986] Ulrich Sieber. *The International Handbook on Computer Crime: Computer-related Economic Crime and the Infringements of Privacy*. John Wiley & Sons, Chichester, New York, Brisbane, Toronto, Singapore, 1986.

In recent years the operation and security of computer systems have become of crucial importance for business and public administration. Computers are now used extensively to administer monetary transactions, prepare balance sheets, control production, hold confidential information and direct air control and defence systems. With the rapid growth of new technology there has been a steady rise in the level of criminal offences involving DP [Data Processing] systems in the USA, Western Europe, Japan and even some socialist states. Due to the increasing dependence on computers, this problem of computer-related crime represents an existential threat not only to individual companies but to the economy as a whole. It is therefore the subject of international concern and has led to an intensive discussion about computer crime and computer abuse in all Western industrial countries.

To evaluate and to overcome computer-related crime there is a need for international co-operation in the fields of criminological research, the clarification and reform of prevailing legal provisions, development of security measures and prosecution of computer crime. This book provides an international and comparative survey of these problems. It gives a comprehensive criminological analysis of computer crime; looks at the present legal situation in Western countries; discusses the new proposals for reforming the systems and the forthcoming computer legislation; analyses the security strategies to prevent computer crime in the future and describes the difficulties of prosecuting computer crime.

[Snow2000] Andrew P. Snow, Upkar Varshney, and Alisha D. Malloy. Reliability and Survivability of Wireless and Mobile Networks. *Computer*, 33(7):49–55, July 2000.

As wireless and mobile services grow, weaknesses in network infrastructures become clearer. Providers must now consider ways to decrease the number of network failures and to cope with failures when they do occur.

[Solo1998] James D. Solomon. *Mobile IP: The Internet Unplugged*. Prentice Hall Series in Computer Networking and Distributed Systems. PTR Prentice Hall, Upper Saddle River, New Jersey 07458, 1998. Radia Perlman, editor.

The complete guide to developing, using, and profiting from Mobile IP networks. Mobile IP brings together two of the world's most powerful technology trends: the Internet and mobile communications. Whether you're planning to develop, deploy, utilize, or invest in Mobile IP networks, this book delivers the up-to-date information you need — with clarity and insight. Discover:

- What problems Mobile IP is designed to solve, and how it solves them
- How to use Mobile IP in real-world intranet and Internet-wide applications
- How to manage the security issues associated with Mobile IP
- Business models for delivering commercial Mobile IP services

- Which technical issues still need work — and possible solutions

In **Mobile IP: The Internet Unplugged**, the co-chair of the Mobile IP Working Group offers an insider's view of critical Mobile IP concepts like agent discovery, registration, and IP encapsulation. He presents detailed coverage of Mobile IP security, including the role of key management, encryption, authentication, integrity checking, and non-repudiation. Finally, he presents a compelling vision of the future, where the benefits of standards-based mobile data are available everywhere.

[Spaf1989a] Eugene H. Spafford. Crisis and Aftermath. *Communications of the ACM*, 32(6):678–687, June 1989.

Last November the Internet was infected with a worm program that eventually spread to thousands of machine, disrupted normal activities and Internet connectivity for many days. The following article examines just how this worm operated.

[Spaf1989b] Eugene H. Spafford. Some Musings on Ethics and Computer Break-Ins. In *USENIX Winter Conference*, Department of Computer Sciences, Purdue University, W. Lafayette, IN 47907-2004, 1989. USENIX.

In November and December, the computing community experienced the release of the Internet Worm, computer break-ins at Lawrence Livermore National Labs, and the temporary disconnection of the Milnet because of computer break-ins on a machine belonging to the Mitre Corporation. These incidents have led to many discussions about responsibility and ethics. Many of these discussions, particularly in forums such as the Usenet, have become heated without leading to any commonly-accepted conclusions.

This paper addresses some of these points. The intent is to summarize a few of the principal arguments supporting various positions and to argue some points of particular merit. At the end, references are given to material that may help provide background material for readers seeking further information.

Included in this discussion are the questions of whether individuals breaking into our machines are doing us a favor, and whether those individuals should in any way be encouraged. The paper concludes with some observations about the importance of the discussion, and the need to reach a consensus in the computer profession, if not in society as a whole.

[Spaf1989c] Eugene H. Spafford. The Internet Worm Program: An Analysis. *Computer Communication Review*, 19(1):17–57, January 1989. Reprint of Purdue Technical Report CSD-TR-823.

On the evening of 2 November 1988, someone infected the Internet with a *worm* program. The program exploited flaws in utility programs in systems based on BSD-derived versions of UNIX. The flaws allowed the program to break into those machines and copy itself, thus *infecting* those systems. This program eventually spread to thousands of machines, and disrupted normal activities and Internet connectivity for many days.

This report gives a detailed description of the components of the worm program — data and functions. It is based on study of **two** completely independent reverse-compilations of the worm and a version disassembled to VAX assembly language. Almost no source code is given in the paper because of current concerns about the state of the "immune system" of Internet hosts, but the description should be detailed enough to allow the reader to understand the behavior of the program.

This paper contains a review of the security flaws exploited by the worm program, and gives some recommendations on how to eliminate or mitigate their future use. The report also includes an analysis of the coding style and methods used by the author(s) of the worm, and draws some conclusions about his abilities and intent.

[Stal1995] William Stallings. *Network and Internetwork Security Principles and Practice*. Prentice Hall, Englewood Cliffs, New Jersey 07632, 1995.

Network and Internetwork Security covers network security technology, the standards that are being developed for security in an internetworking environment, and the practical issues involved in developing security applications. The first part of the book is a tutorial on and survey of network security technology. Each of the basic building blocks of network security, including conventional and public-key cryptography, authentication, and digital signatures, is covered. In addition the first part explores methods for countering hackers and viruses. The second part of the book is devoted to a thorough discussion of important network security applications, including PGP, PEM, Kerberos, and SNMPv2 security.

[Stev1994] W. Richard Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley Professional Computing Series. Addison Wesley, One Jacob Way / Reading, Massachusetts 01867, 1994. See also Wright and Stevens, Vol. 2.

TCP/IP Illustrated is a complete and detailed guide to the entire TCP/IP protocol suite — with an important difference from other books on the subject. Rather than just describing what the RFCs say the protocol suite should do, this unique book uses a popular diagnostic tool so you may actually watch the protocols in action.

By forcing various conditions to occur — such as connection establishment, timeout and retransmission, and fragmentation — and then displaying the results, TCP/IP Illustrated gives you a much greater understanding of these concepts than words alone could provide. Whether you are new to TCP/IP or you have read other books on the subject, you will come away with an increased understanding of how and why TCP/IP works the way it does, as well as enhanced skill at developing applications that run over TCP/IP.

With this unique approach, TCP/IP Illustrated presents the structure and function of TCP/IP from the link layer up through the network, transport, and application layers. You will learn about the protocols that belong to each of these layers and how they operate under numerous implementations, including Sun OS 4.1.3, Solaris 2.2, System V Release 4, BSD/386TM, AIX 3.2.2, and 4.4BSD.

In TCP/IP Illustrated you will find the most thorough coverage of TCP available - 8 entire chapters. You will also find coverage of the newest TCP/IP features, including multicasting, path MTU discovery, and long fat pipes.

[Stra1984] Jr. Detmar W. Straub and Cathy Spatz Widom. Deviancy by Bits and Bytes: Computer Abusers and Control Measures. In *Proceedings of the 2nd IFIP International Conference on Computer Security*, 1984.

The phenomena of computer crime and abuse include a wide spectrum of activities. The motivations of computer criminals or abusers also range from ignorance and misunderstanding to the intentional and purposeful malfeasance of career criminals. Drawing upon the detailed case histories of Parker and others, this paper proposes a 4-part taxonomy of computer abusers which focuses on criminogenic motivations. The proposed taxonomy can serve as a framework for theories and theory-testing and demonstrates the markedly different countermeasures called for by each type of motivation. In terms of general control measures, it is shown that deterrence of abuse should receive the greatest attention from those responsible for the security role in the organization.

[Syve1994] Paul Syverson. A Taxonomy of Replay Attacks. In *Proceedings of the 7th Computer Security Foundations Workshop*, pages 187–191. IEEE, 1994.

This paper presents a taxonomy of replay attacks on cryptographic protocols in terms of message origin and destination. The taxonomy is independent of any method used to analyze or prevent such attacks. It is also complete in the sense that any replay attack is composed entirely of elements classified by the taxonomy. The classification of attacks is illustrated using both new and previously known attacks on protocols. The taxonomy is also used to discuss the appropriateness of particular countermeasures and protocol analysis methods to particular kinds of replays.

[Thie1997] Richard Thieme. Zen and the Art of Hacking. *Internet Underground*, 2(4):26–33, April 1997.

Don't call them hackers, call them *Homo sapiens hackii* — human beings who are "back-engineered" by their symbiotic relationship with computer networks to frame reality in ways shaped by that interaction. They're not a new species, but they're a new variety, and just like the pod people in the *Invasion of the Body Snatchers*, they're everywhere. But how can separate [sic] the real thing from the pretenders?

Looks, jargon and hard-guy handles are too easy to imitate. Besides, real hackers blend in well with their surroundings — that's the point of social engineering, after all — and hide in large corporations, high-tech startups and IT departments, and in intelligence, security and law enforcement.
Some don't even use computers very much.

[Thom1984] Ken Thompson. Reflections on Trusting Trust. *Communications of the ACM*, 27(8):761–763, August 1984.

To what extent should one trust a statement that a program is free of Trojan horses? Perhaps it is more important to trust the people who wrote the software.

[Tzu1963] Sun Tzu. *The Art of War*. Oxford University Press, London, 1963. Translated and with an introduction by Samuel B. Griffith with a forward by B.H. Liddell Hart. Originally written c. 400–320 B.C.

Sun Tzu's essays on 'The Art of War' form the earliest of known treatises on the subject, but have never been surpassed in comprehensiveness and depth of understanding. They might be termed the concentrated essence of wisdom on the conduct of war. Among all the military thinkers of the past, only Clausewitz is comparable, and even he is more 'dated' than Sun Tzu, and in part antiquated, although he was writing more than two thousand years later. Sun Tzu has clearer vision, more profound insight, and eternal freshness....

In brief, Sun Tzu was the best short introduction to the study of warfare, and no less valuable for constant reference in extending study of the subject.

[VanD1985] J.A. VanDuyn. *The Human Factor in Computer Crime*. Petrocelli Books, Inc., 1985. Primary author is Van Duyn.[40]

To appreciate the extent to which human factor plays a part in computer crime, we have to be aware of its role in crime deterrence, prevention, detection, and risk assessment on the physical, hardware, software, and personnel security levels.

Considering the rather recent appreciation of the last security category, perhaps it isn't odd that at most DP [Data Processing] facilities while great attention is afforded to physical, hardware, and software security measures and controls, little or no thought is given to personnel security. Yet, without effective personnel security, meaning a security program which is designed to foster the most effective deterrent against computer crime: *job satisfaction*, the most sophisticated hardware and software security systems are worthless.

This does not mean to imply that the first three computer security categories are not important insofar as computer crime countermeasures are concerned. Far from it.

Nevertheless, unless the human factor — an element often ignored by management an neglected by high technology technicians — is considered in every aspect of computer security, the business or government organization and its DP operations are highly vulnerable to computer crime, as the case histories offered in this book indicate.

[40]Last name is actually Van Duyn, but to get it listed correctly in BibTeX it was listed without a space. My apologies to the author.

[Vand1999] Jim Vanderwalker. A Superhighway Like the Internet, 1999. May have been published a few years earlier. No date on essay; date was when found on Internet at http://www.cs.ucdavis.edu/∼bishop/humor.

"Think of the Internet as a highway."

There it is again. Some clueless fool talking about the "Information Superhighway." They don't know didley about the net. It's nothing like a superhighway. That's a rotten metaphor.

Suppose the metaphor ran in the other direction. Suppose the highways were like the net ...

A highway hundreds of lanes wide. Most with pitfalls for potholes. Privately operated bridges and overpasses. No highway patrol. A couple of rent-a-cops on bicycles with broken whistles. 500 member vigilante posses with nuclear weapons. A minimum of 237 on ramps at every intersection. No signs. Wanna get to Ensenada? Holler out the window at a passing truck to ask directions. Ad hoc traffic laws. Some lanes would vote to make use by a single-occupant-vehicle a capital offense on Monday through Friday between 7:00 and 9:00. Other lanes would just shoot you without a trial for talking on a car phone.

AOL would be a giant diesel-smoking bus with hundreds of ebola victims on board throwing dead wombats and rotten cabbage at the other cars, most of which have been assembled at home from kits. Some are built around 2.5 horsepower lawnmower engines with a top speed of nine miles an hour. Others burn nitrogylcerin and idle at 120.

No license plates. World War II bomber nose art instead. Terrifying paintings of huge teeth or vampire eagles. Bumper mounted machine guns. Flip somebody the finger on this highway and get a white phosphorus grenade up your tailpipe. Flatbed trucks cruise around with anti-aircraft missile batteries to shoot down the traffic helicopter. Little kids on tricycles with squirtguns filled with hydrochloric acid switch lanes without warning.

NO OFFRAMPS. None.

Now that's the way to run an Interstate Highway system.

[vanE1985] Wim vanEck. Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk? *Computers & Security*, 4(4):269–286, December 1985. [41]

This paper describes the results of research into the possibility of "eavesdropping" on video display units, by picking up and decoding the electromagnetic interference produced by this type of equipment. During the research project, which started in January, 1983, it became more and more clear that this type of information theft can be committed very easily using a normal TV receiver.

[Vars2000] Upkar Varshney. Recent Advances in Wireless Networking. *Computer*, 33(6):100–103, June 2000.

After discussing advances in wired networking in a previous column ("Recent Advances in Wired networking," *Computer*, April 2000, pp. 107–109), I now turn to advances in wireless networking. Wireless networks can include everything from cellular, personal communications system (PCS), and Group System for Mobile communications (GSM) networking to wireless LANs, satellite-based networks, and fixed wireless networks. Many of these technologies have experienced significant growth lately because of an increasingly mobile workforce and accelerating user acceptance.

Although voice and short messaging drove the early successes of wireless networks, data and more sophisticated applications are driving advancements today. Significant progress has also been made in middleware development and standardization. These factors — along with demand for higher bandwidth and global roaming — will continue to push the standardization and the near-future deployment of third-generation wireless networks using terrestrial and satellite components....

[41] Last name is actually van Eck, but to get it listed correctly in BibTeX, it was listed without a space. My apologies to the author.

[Vene1992] Wietse Venema. TCP WRAPPER Network monitoring, acess control, and booby traps. In *USENIX UNIX Security Symposium III*, September 14–16 1992. Baltimore, MD.

This paper presents a simple tool to monitor and control incoming network traffic. The tool has been successfully used for shielding off systems and for detection of cracker activity. It has no impact on legal computer users, and does not require any change to existing systems software or configuration files. The tool has been installed world-wide on numerous UNIX systems without any source code change.

[Vene1996] Wietse Venema. Murphy's Law and Computer Security. In *Proceedings of the 6th USENIX Security Symposium Focusing on Applications of Cryptography*, pages 187–193, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710, July 22–25 1996.

This paper discusses lessons learned from a selection of computer security problems that have surfaced in the recent past, and that are likely to show up again in the future. Examples are taken from security advisories and from unpublished loopholes in the author's own work.

[vonN1961] John vonNeumann. *Various Techniques Used in Connection With Random Digits*, volume V (Design of Computers, Theory or Automata and Numerical Analysis) of *John von Neumann Collected Works; A.H. Taub, General Editor*, pages 768–770. Pergamon Press, Oxford, 1961. Summary written by George E. Forsythe; originally in J. Res. Nat. Bus. Stand. Appl. Math. Series 3:36–38 (1951).[42]

In manual computing methods today random numbers are probably being satisfactorily obtained from tables. When random numbers are to be used in fast machines, numbers will usually be needed faster. More significant is the fact that, because longer sequences will be used, one is likely to have more elaborate requirements about what constitutes a satisfactory sequence of random numbers. There are two classes of questions connected with high-speed computation about which I should like to make some remarks: (A) How can one produce a sequence of random decimal digits — a sequence where each digit appears with probability one-tenth and where consecutive ones are independent of each other in all combinations? (B) How can one produce random real numbers according to an assigned probability distributed law?

[Wagn1983] Neal R. Wagner. Fingerprinting. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 18–22, 1983.

This paper presents a general discussion of the use of fingerprints, especially fingerprinted data. Fingerprinted is classified in four orthogonal ways, and some illustrative examples are given. The basis for a statistical analysis of altered fingerprints is presented, along with an example simulation. The possibility of more subtle fingerprints is discussed.

[42]Last name is actually von Neumann, but to get it listed correctly it was listed without a space. My apologies to the author.

[Wake1990] John F. Wakerly. *Digital Design: Principles and Practice.* Prentice Hall Series in Computer Engineering; Edward J. McCluskey, Series Editor. Prentice Hall, Englewood Cliffs, N.J. 07632, 1990.

This new book by one of the most highly respected and successful digital system designers and authors offers an authoritative introduction to basic principles of digital design *and* a practical guide to techniques used in both board-level and VLSI systems.

Distinguishing features include:

- Development of structured design techniques for real applications using MSI functions, PLDs, ROMs, RAMs, and state machines
- An emphasis on electrical circuit properties of logic elements to assist the reader in understanding the "non-digital" behavior of digital circuits in real designs
- Focus on modern digital design and debugging environments, including documentation practices and CAE tools.
- Over 400 illustrations with a functional use of color.

[Wall2001] Paul Wallich. Amiga: The Computer That Wouldn't Die. *IEEE Spectrum*, 38(3):40–46, March 2001.

Having outlived a number of its corporate owners, and spurred on by its passionate users, the Amiga computer is going back to market again — but this time, in very different forms.

[Walt1998] Edward Waltz. *Information Warfare: Principles and Operations.* Artech House, 1998.

The forms of cooperation, competition, conflict, and warfare in this world are changing as information technology is changing the way that we observe, understand, decide, and communicate. These changes are impacting every aspect of personal, corporate, and national security. This book provides a systems-level introduction of the means by which information technology is changing conflict and warfare....

Following an introduction to this emerging area, the book is divided into two major parts, describing the major components of information warfare.

Part I describes the basis for information-based warfare (IBW), beginning with the information sciences that define information (Chapter 2), and technologies to create knowledge from data (Chapter 3). Chapter 4 describes the means to apply these technologies to achieve dominant battlespace awareness and knowledge, the goal of IBW.

Part II details information operations (IO) that attack opponents' information systems and defend a national's own systems. We describe first information warfare (IW) policy, strategy, and tactics (Chapter 5), then IO operations (Chapters 6 and 7) before detailing methods for offense (Chapter 8) and defense (Chapter 9). An overview of core, enable, and emerging technologies in this area is provided in the conclusion (Chapter 10).

[Weis1973] Clark Weissman. System Security Analysis / Certification Methodology and Results. Technical Report SP-3728, System Development Corporation, October 8, 1973.

While limited protection is present in contemporary systems, intelligent, motivated users have little difficulty obtaining unauthorized data and program access by clever exploitation of operating system security flaws.

This presentation outlines a comprehensive security analysis and certification method, developed and successfully applied by System Development Corporation (SDC), that determines system vulnerability by detecting security flaws not found in conventional system testing.

[Weis1995] Clark Weissman. Penetration Testing. In Marshall D. Abrams, Sushil Jajodia, and Harold J. Podell, editors, *Information Security: An Integrated Collection of Essays*, chapter 11, pages 269—296. IEEE Computer Society Press, 1995.

Near flawless penetration testing is a requirement for high-rated secure systems — those rated above B1 based on the Trusted Computer System Evaluation Criteria (TCSEC) and its Trusted Network and Database Interpretations (TNI and TDI). Unlike security functional testing, which demonstrates correct behavior of the product's advertised security controls, penetration testing is a form of stress testing which exposes weaknesses — that is, flaws — in the *trusted computing base* (TCB). This essay describes the Flaw Hypothesis Methodology (FHM), the earliest comprehensive and widely used method for conducting penetrations testing. It reviews motivation for penetration testing and penetration test planning, which establishes the goals, ground rules, and resources available for testing. The TCSEC defines "flaw" as "an error of commission, omission, or oversight in a system that allows protection mechanisms to be bypassed." This essay provides an overview of FHM and its analogy to a heuristic-based strategy game.

[Whit1978] Thomas Whiteside. *Computer Capers: Tales of Electronic Thievery, Embezzlement, and Fraud*. Thomas Y. Crowell, New York, 1978.

Crime usually does its ingenious best to keep pace with technology, so it is probably inevitable that some of the more interesting specialized manifestations of criminality to emerge recently have had to do with deliberate misuses of the computer.

[Whit1996] Gregory B. White, Eric A. Fisch, and Udo W. Pooch. *Computer System and Network Security*. Computer Engineering Series, Udo W. Pooch series editor. CRC Press, 1996.

Computer System and Network Security provides the reader with a basic understanding of the issues involved in the security of computer systems and networks. Introductory in nature, this important new book covers all aspects related to the growing field of computer security. Such complete coverage in a single text has previously been unavailable, and college professors and students, as well as professionals responsible for system security, will find this unique book a valuable source of information, either as a textbook or as a general reference.

Computer System and Network Security discusses existing and potential threats to computer systems and networks and outlines the basic actions that are generally taken to protect them. The first two chapters of the text introduce the reader to the field of computer security, covering fundamental issues and objectives. The next several chapters describe security models, authentication issues, access control, intrusion detection, and damage control. Later chapters address network and database security and systems/networks connected to wide-area networks and internetworks. Other topics include firewalls, cryptography, malicious software, and security standards. The book includes case studies with information about incidents involving computer security, illustrating the problems and potential damage that can be caused when security fails.

This unique reference/textbook covers all aspects of computer and network security, filling an obvious gap in the existing literature.

[Wilk1981] A. L. Wilkinson, D. H. Anderson, D. P. Chang, Lee Hock Hin, A. J. Mayo, I. T. Viney, R. Williams, and W. Wright. A Penetration Analysis of a Burroughs Large System. *Operating Systems Review*, 15(1):14–25, 1981.

A penetration analysis of a Burroughs Large System is described and a fundamental flaw allowing total penetration is discussed. Specific aspects examined include File Security, Resource Limits, Accounting Methods, and Disruptions.

[Will1995] Jeffrey R. Williams, Marv Schaefer, and Douglas J. Landoll. Pretty Good Assurance. In *New Security Paradigms Workshop*, pages 82–89, Arca Systems, Inc. 8229 Boone Blvd., Suite 610 Vienna, VA 22182, August 22–25 1995. ACM SIGSAC.

This paper describes the need for pretty good assurance: clearly stated claims about the security properties of systems, accompanied by evidence that explains in clear terms why we should believe that these claims are substantiated. Several different types of threats are identified are their relationships to assurance are explored. The developer's role in creating an assurance argument is distinguished from the user's role consuming assurance. Finally, some thoughts on the future are presented.

[Wink1996] Ira S. Winkler. Case Study of Industrial Espionage through Social Engineering. In *19th National Information Systems Security Conference*, volume 1, pages 306–312, 1996.

The Federal Bureau of Investigation estimates that U.S.,Corporations lose $100 Billion annually due to industrial espionage. While many people believe that the espionage is committed by well financed organizations that can only be stopped by national agencies, that is very incorrect. Industrial espionage usually exploits simple and very preventable vulnerabilities to produce tremendous results. By focusing on comprehensive security, and not just technical security, information security professionals can significantly hamper adversary attempts to steal their organization's information assets. The presentation that describes this paper presents a case study of an actual industrial espionage attack against a large U.S.,corporation.

[Wood1990] Nancy K. Woodfield. An Approach for Evaluating the Security of an Air Force Type Network. In *Fifth Annual Computer Security Applications Conference*, pages 53–62, Piscataway, NJ, 1990. IEEE.

This paper discusses an approach for assessing the security, AFR 205-16 sensitive/unclassified and Trusted Network Interpretation C2 level, of an Air Force type network environment. This approach was used to evaluate a "target" network. The evaluation also looked at the security impacts of incorporating computer applications and systems into an integrated computer network. Also discussed are GOSIP, X.25, Closed User Groups and OSI.

[Wrig1995] Gary R. Wright and W. Richard Stevens. *TCP/IP Illustrated, Volume 2: The Implementation*. Addison-Wesley Professional Computing Series. Addison Wesley, One Jacob Way / Reading, Massachusetts 01867, 1995. See also Stevens, Vol. 1.

TCP/IP Illustrated, an ongoing series covering the many facets of TCP/IP, brings a highly-effective visual approach to learning about this networking protocol suite.

TCP/IP Illustrated, Volume 2 contains a thorough explanation of how TCP/IP protocols are implemented. There isn't a more practical or up-to-date book — this volume is the only one to cover the de facto standard implementation from the 4.4BSD-Lite release, the foundation for TCP/IP implementations run daily on hundreds of thousands of systems worldwide.

Combining 500 illustrations with 15,000 lines of real working code, *TCP/IP Illustrated, Volume 2* uses a teach-by-example approach to help you master TCP/IP implementation. You will learn about such topics as the relationship between the sockets API and the protocol suite, and the differences between a host implementation and a router. In addition, the book covers the newest features of the 4.4BSD-Lite release, including

multicasting, long fat pipe support, window scale, timestamp options, protection against wrapped sequence numbers, and many other topics.

Comprehensive in scope, based on a working standard, and thoroughly illustrated, this book is an indispensable resource for anyone working with TCP/IP.

[Youn1979] W.R. Young. Advanced Mobile Phone Service: Introduction, Background, and Objectives. *Bell Systems Technical Journal*, 58:1–14, January 1979.

This paper introduces a series of papers that describe in detail the Bell System's Advanced Mobile Phone Service (AMPS). It presents a brief history of mobile radio, highlighting the important events and legal decisions that preceded development of the AMPS system. The cellular system concept that has been embodied in AMPS makes large-scale mobile radio service affordable to a sizable segment of the public. This concept calls for dividing transmission areas into "cells" to handle radio traffic, and, as traffic grows, subdividing those cells into smaller segments without increasing radio spectrum. This paper outlines AMPS objectives and sets the stage for more detailed articles on its evolution, its design and testing, and maintenance considerations.

[Youn1987] William D. Young and John McHugh. Coding For a Believable Specification to Implementation Mapping. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, pages 140–148, 1987.

One criterion for "Beyond A1" certification according to the DoD *Trusted Computer Systems Evaluation Criteria* will be the code-level verification. We argue that, while verification at the actual code level may be infeasible for large secure systems, it is possible to push the verification to a low level of abstraction and then map the specification in an intuitive manner to the source code. Providing a suitable mapping requires adhering to a strict discipline on both the specification and code sides. We discuss the issues involved in this problem, particularizing the discussion to a mapping from Gypsy specifications to C code.

[Zhou1999] Lidong Zhou and Zygmunt J. Haas. Securing Ad Hoc Networks. *IEEE Network*, 13(6):24–30, November/December 1999.

Ad hoc networks are a new wireless networking paradigm for mobile hosts. Unlike traditional mobile wireless networks, ad hoc networks do not rely on any fixed infrastructure. Instead, hosts rely on each other to keep the network connected. Military tactical and other security-sensitive operations are still the main applications of ad hoc networks, although there is a trend to adopt ad hoc networks for commercial uses due to their unique properties. One main challenge in the design of these networks is their vulnerability to security attacks. In this article, we study the threats an ad hoc network faces and the security goals to be achieved. We identify the new challenges and opportunities posed by this new networking environment and explore new approaches to secure its communication. In particular, we take advantage of the inherent redundancy in ad hoc networks — multiple routes between nodes — to defend routing against denial-of-service attacks. We also use replication and new cryptographic schemes, such as threshold cryptography, to build a highly secure and highly available key management service, which forms the core of our security framework.

[Zimm1991] D. Zimmerman. The Finger User Information Protocol. Request For Comments (RFC) 1288, December 1991.

This memo describes the Finger user information protocol. This is a simple protocol which provides an interface to a remote user information program.

Based on RFC 742, a description of the original Finger protocol, this memo attempts to clarify the expected communication between the two ends of a Finger connection. It also tries not to invalidate the many existing implementations or add unnecessary restrictions to the original protocol definition.

This edition corrects and clarifies RFC 1196.

Vita

Daniel Lowry Lough was born on October 31, 1971 in Fredericksburg, Virginia. After graduating from Stafford Senior High School in 1989, he moved to Blacksburg, Virginia to attend Virginia Tech. Daniel received his B.S. in Computer Engineering in 1994, and he also passed the Fundamentals of Engineering (FE) exam in the same year. He received a M.S. in Electrical Engineering in 1997 from Virginia Tech under the requirements of the then-forthcoming M.S. in Computer Engineering. In January 1997, Daniel was awarded a Bradley Fellowship (later Hekimian Bradley Fellowship) from the Bradley Department of Electrical Engineering at Virginia Tech to complete his Ph.D. degree. He met his wife Rebecca at Virginia Tech in the spring of 1997, and they were married two years later. Rebecca is currently completing her Ph.D. in Horticulture at North Carolina State University in Raleigh, North Carolina. Daniel has lived in Raleigh for the last two years completing this dissertation. This summer he will start a job in the Washington, D.C. area.

Daniel L. Lough
daniel@lough.org
http://www.lough.org

Daniel Lowry Lough

daniel@lough.org
www.lough.org

OBJECTIVE: Computer Security Engineer working under a Professional Engineer (PE) on issues in critical infrastructure protection, information warfare, secure protocol development, or wireless security

EDUCATION:

Ph.D., Computer Engineering, May 2001
Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, VA
Dissertation: A Taxonomy of Computer Attacks with Applications to Wireless Networks
GPA: *4.0 / 4.0*

M.S., Electrical Engineering, May 1997
Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, VA
Project and Report: Proposed Next Generation of a Real Time Bit Error Rate Simulator
GPA: *4.0 / 4.0*

B.S., Computer Engineering, May 1994
Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, VA
Minor: Computer Science

Related Courses:

Computer Network Security	Network Application Design
Computer Security Fundamentals	Computer and Network Architecture
Foundations of Cryptography	Rapid Prototyping of Computing Machinery
Advanced Internetwork Architectures	Microprocessor System Design

Ph.D. Project, August 1997 – Present
- Determined existence of common set of similar attacks over past thirty years
- Merged past operating system attacks into common taxonomy

M.S. Project, August 1995 – May 1996
- Determined that implementation of patented BERSIM (Bit Error Rate SIMulator) sold by Virginia Tech Intellectual Properties contained errors (U.S. Patent 5,233,628)
- Documented area of hardware that needed to be redesigned

Major Graduate Project, August – December 1995
- Designed 16-bit RISC general purpose microprocessor
- Implemented microprocessor on Xilinx FPGA platform

Major Undergraduate Project, August 1993 – May 1994
- Led team of ten students to design and create mobile robot for IEEE contest
- Designed and created microcontroller board to control mobile robot

Engineer Intern, Fundamentals of Engineering Examination passed, July 1994
Virginia Tech, *Blacksburg, VA*

CLEARANCE: **Secret Security Clearance**, *Naval Surface Warfare Center, Dahlgren, VA*, 1991 – 1996

COMPUTER SKILLS: Security, TCP/IP, IPv6, Mobile IP, Design with FPGAs, Windows, UNIX, MS Word, LATEX, C, VHDL, various Assembly languages

EXPERIENCE:

Instructor of Computer Engineering, June – August 1997, January – May 1998
Bradley Department of Electrical and Computer Engineering, *Virginia Tech*, *Blacksburg, VA*
- Developed course materials for senior level class on VHDL
- Instructed 70 undergraduate and graduate students in two sections of course
- Lectured on VHDL, design process, and simulating hardware in VHDL

Substitute Instructor of Computer Engineering (part time), 1998 – 1999
Bradley Department of Electrical and Computer Engineering, *Virginia Tech*, *Blacksburg, VA*
- Computer Security Fundamentals (graduate level)
- Modeling with Hardware Description Languages (graduate level)
- Computer Network Engineering (undergraduate level)
- Digital Design I (undergraduate level)

Graduate Research Assistant, January – May 1996
Mobile and Portable Radio Research Group, *Virginia Tech, Blacksburg, VA*
- Researched techniques to update commercial computer product BERSIM
- Assisted users of MPRG products in solving technical problems

Student Trainee, Electronics Engineer (Graduate Co-op), Summer, Christmas 1994 – 1996
Naval Surface Warfare Center, Dahlgren Division, *Dahlgren, VA*
- Identified problems with system running on real time OS
- Configured World Wide Web server for department
- Created HTML pages with basic security for departmental World Wide Web server
- Wrote Interface Design Document for connection between development systems

Engineering Aid, Summer, Christmas 1991 – 1994
Naval Surface Warfare Center, Dahlgren Division, Dahlgren, VA
- Acted as system administrator on self-configured Sun 630MP network server
- Assisted in setup of VMEbus system and Ada real time OS on MVME167 boards
- Installed ethernet interface between Sun IPCs and 68040 development system

REFEREED PUBLICATIONS:

The Next Generation of the Internet: Aspects of the Internet Protocol Version 6
David C. Lee, Daniel L. Lough, Scott F. Midkiff, Nathaniel J. Davis, IV, and Phillip E. Benchoff, *IEEE Network*, Vol. 12, No. 1, pp. 28 – 33, January/February 1998

The Internet Protocol Version 6: The Future Workhorse of the Information Highway
David C. Lee and Daniel L. Lough, *IEEE Potentials*, pp. 11 – 12, April/May 1998

NON-REFEREED PUBLICATIONS:

A Short Tutorial on Wireless LANs and IEEE 802.11
Daniel L. Lough, T. Keith Blankenship, and Kevin J. Krizman
looking .forward, a supplement to IEEE Computer, Vol. 5, No. 2, pp. 9 – 12, Summer 1997

TECHNICAL PRESENTATIONS:

A Taxonomy of Computer Attacks, work in progress, 16 May 2000
Daniel L. Lough, Nathaniel J. Davis, IV, and Randy C. Marchany
21st IEEE Symposium on Security and Privacy

The History and Future of Computer Attack Taxonomies, work in progress, 26 August 1999
8th USENIX Security Symposium
abstract printed in *;login: The USENIX Association Magazine*, p. 22, November 1999
Daniel L. Lough, Nathaniel J. Davis, IV, and Randy C. Marchany

IPv6: Who / What / When / Where / How / Why, 10 July 1999
Daniel L. Lough, *DefCon VII*

The History and Future of Computer Attack Taxonomies, work in progress, 11 May 1999
20th IEEE Symposium on Security and Privacy
Daniel L. Lough, Nathaniel J. Davis, IV, and Randy C. Marchany

Security of Wireless Technology, 2 August 1998
Daniel L. Lough, *DefCon VI*

MANUSCRIPTS REVIEWED:

Security Engineering: A Comprehensive Guide to Building Dependable Distributed Systems, Ross J. Anderson, Wiley Computer Publishing, 2001

LEADERSHIP POSITIONS HELD:

- **President**, Eta Kappa Nu, Beta Lambda Chapter (Electrical Engineering Honor Society), *Virginia Tech, Blacksburg, VA*, 1994 – 1995
- **Chair**, Student Advisory Council, Bradley Department of Electrical Engineering, *Virginia Tech, Blacksburg, VA* 1994-1995
- **Team Project Leader**, IEEE Mobile Robot Competition, *Virginia Tech Team, Blacksburg, VA*, August 1993 – May 1994
- **Student Representative**, Student Advisory Committee to the Vice President for Finance and Treasurer, University Committee, *Virginia Tech, Blacksburg, VA*, 1997 – 1999
- **Student Representative**, Electrical Engineering Undergraduate Advisory Committee, Department Committee, Bradley Department of Electrical Engineering, *Virginia Tech, Blacksburg, VA*, 1994 – 1995
- **Student Representative**, Commission on Graduate Studies and Policies, *Virginia Tech, Blacksburg, VA*, 1997 – 1998
- **Delegate**, Graduate Student Assembly, *Virginia Tech, Blacksburg, VA*, January – May 1996, August 1997 – May 1998
- **Representative**, Student Government Association House of Representatives, *Virginia Tech, Blacksburg, VA*, August 1997 – May 1998

HONORS AND AWARDS:

- **Hekimian Bradley Fellow**, four years funding of Ph.D., August 1997 – May 2001 *Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA*
- **Bradley Fellow**, semester of funding of M.S., January – May 1997 *Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA*
- **Travel Fund Grant**, Graduate Student Assembly, *Virginia Tech, Blacksburg, VA*, December 1999
- **USENIX Travel Grant**, 8th Security Symposium, *Washington DC,* September 1999
- **Travel Fund Grant**, Graduate Student Assembly, *Virginia Tech, Blacksburg, VA*, December 1998
- **Tau Beta Pi**, Virginia Beta Chapter (Engineering Honor Society), *Virginia Tech, Blacksburg, VA*, 1995 – Present
- **Eta Kappa Nu**, Beta Lambda Chapter (Electrical Engineering Honor Society), *Virginia Tech, Blacksburg, VA*, 1992 – Present
- **Phi Kappa Phi**, Virginia Tech Chapter (Honor Society), *Blacksburg, VA*, 1996 – Present
- **Omicron Delta Kappa**, Alpha Omicron Circle (Leadership Honor Society), *Blacksburg, VA*, 1999 – Present

PROFESSIONAL MEMBERSHIPS:

- **Institute of Electrical and Electronic Engineers**, Student Member, 1992 – Present
- **Computer Society, IEEE**, Student Member, 1992 – Present
- **Association of Computing Machinery**, Student Member, 1994 – Present
- **USENIX Computer Association**, Student Member, 1997 – Present
- **National Society of Professional Engineers**, 1999 – Present
- **Student Advisory Council**, Electrical Engineering, Virginia Tech, 1994 – 1996