

Contents

Understanding the Current Legal Climate

This book will teach you techniques that, if used in the wrong way, will get you in trouble with the law. Me saying this is like a driving instructor saying, “I’m going to teach you how to drive; if you drive badly, you might run someone over.” In both cases, any harm done would be *your* fault.

Tools & Traps...

Want to Check that Firewall?

There are an incredible number of freeware tools available to you for beginning your checks of vulnerability. I have a couple of favorites that allow for quick probes and checks of information about various IP addresses:


- SuperScan, from Foundstone Corporation: www.foundstone.com/knowledge/free_tools.html
- Sam Spade, from SamSpade.org: www.samspade.org.

Foreword v 1.5	xxix
Foreword v 1.0	xxxiii
Chapter 1 How To Hack	1
Introduction	2
What We Mean by “Hack”	2
Why Hack?	3
Knowing What To Expect in the Rest of This Book	4
Understanding the Current Legal Climate	6
Summary	8
Frequently Asked Questions	8
Chapter 2 The Laws of Security	11
Introduction	12
Knowing the Laws of Security	12
Client-Side Security Doesn’t Work	14
You Cannot Securely Exchange Encryption	
Keys without a Shared Piece of Information	15
Malicious Code Cannot Be	
100 Percent Protected against	18
Any Malicious Code Can Be Completely	
Morphed to Bypass Signature Detection	20
Firewalls Cannot Protect	
You 100 Percent from Attack	22
Social Engineering	24
Attacking Exposed Servers	24
Attacking the Firewall Directly	26
Client-Side Holes	26
Any IDS Can Be Evaded	27
Secret Cryptographic Algorithms Are Not Secure	28
If a Key Is Not Required, You Do Not Have	
Encryption—You Have Encoding	30
Passwords Cannot Be Securely Stored on	
the Client Unless There Is Another Password	
to Protect Them	32
In Order for a System to Begin to Be	
Considered Secure, It Must Undergo	
an Independent Security Audit	35
Security through Obscurity Does Not Work	37

Summary	39
Solutions Fast Track	39
Frequently Asked Questions	42

Chapter 3 Classes of Attack 45

Introduction	46
Identifying and Understanding the Classes of Attack	46
Denial of Service	47
Local Vector Denial of Service	47
Network Vector Denial of Service	50
Information Leakage	56
Service Information Leakage	56
Protocol Information Leakage	58
Leaky by Design	60
Leaky Web Servers	60
A Hypothetical Scenario	61
Why Be Concerned with Information Leakage?	61
Regular File Access	62
Permissions	62
Symbolic Link Attacks	63
Misinformation	65
Standard Intrusion Procedure	67
Special File/Database Access	69
Attacks against Special Files	69
Attacks against Databases	70
Remote Arbitrary Code Execution	72
The Attack	73
Code Execution Limitations	74
Elevation of Privileges	74
Remote Privilege Elevation	75
Identifying Methods of Testing for Vulnerabilities	77
Proof of Concept	77
Exploit Code	78
Automated Security Tools	79
Versioning	79
Standard Research Techniques	80
Whois	81
Domain Name System	86
Nmap	89
Web Indexing	90

 There are seven classes of attacks: denial of service (DoS), information leakage, regular file access, misinformation, special file/database access, remote arbitrary code execution, and elevation of privileges.

Summary	93
Solutions Fast Track	95
Frequently Asked Questions	96

Q: Is decompiling and other reverse engineering legal?

A: In the United States, reverse engineering may soon be illegal. The Digital Millennium Copyright Act includes a provision designed to prevent the circumvention of technological measures that control access to copyrighted works. Source code can be copyrighted, and therefore makes the reverse engineering of copyrighted code illegal.

Chapter 4 Methodology 99

Introduction	100
Understanding Vulnerability Research	
Methodologies	100
Source Code Research	101
Searching For Error-Prone Functions	101
Line-By-Line Review	102
Discovery Through Difference	102
Binary Research	104
Tracing Binaries	104
Debuggers	105
Guideline-Based Auditing	105
Sniffers	105
The Importance of Source Code Reviews	106
Searching Error-Prone Functions	106
Buffer Overflows	106
Input Validation Bugs	110
Race Conditions	112
Reverse Engineering Techniques	113
Disassemblers, Decompilers, and Debuggers	120
Black Box Testing	125
Chips	126
Summary	128
Solutions Fast Track	129
Frequently Asked Questions	130

Recursive Grepning

According to Ryan Tennant's (Argoth) Solaris Infrequently Asked Obscure Questions (IAOQ) at <http://shells.devunix.org/~argoth/iaoq>, a recursive *grep* can be performed using the following command:

```
/usr/bin/find . |
/usr/bin/xargs
/usr/bin/grep PATTERN
```

Chapter 5 Diffing 131

Introduction	132
What Is Diffing?	132
Why Diff?	135
Looking to the Source Code	136
Going for the Gold: A Gaming Example	139
Exploring Diff Tools	143
Using File-Comparison Tools	143
Using the <i>fc</i> Tool	143
Using the <i>diff</i> Command	145
Working with Hex Editors	146
Hackman	147
[N] Curses Hexedit	148
Hex Workshop	149

Utilizing File System Monitoring Tools	150
Doing It The Hard Way: Manual	
Comparison	150
Comparing File Attributes	151
Using the Archive Attribute	153
Examining Checksums and Hashes	154
Finding Other Tools	155
Troubleshooting	157
Problems with Checksums and Hashes	157
Problems with Compression and Encryption	159
Summary	160
Solutions Fast Track	161
Frequently Asked Questions	162
Chapter 6 Cryptography	165
Introduction	166
Understanding Cryptography Concepts	166
History	167
Encryption Key Types	167
Learning about Standard Cryptographic	
Algorithms	169
Understanding Symmetric Algorithms	170
DES	170
AES (Rijndael)	172
IDEA	173
Understanding Asymmetric Algorithms	174
Diffie-Hellman	174
RSA	176
Understanding Brute Force	177
Brute Force Basics	177
Using Brute Force to Obtain Passwords	178
L0phtcrack	180
Crack	181
John the Ripper	182
Knowing When Real Algorithms	
Are Being Used Improperly	183
Bad Key Exchanges	183
Hashing Pieces Separately	184
Using a Short Password to Generate	
a Long Key	185
Improperly Stored Private or Secret Keys	186
Understanding Amateur Cryptography Attempts	188
Classifying the Ciphertext	189

John the Ripper

John the Ripper is another password-cracking program, but it differs from Crack in that it is available in UNIX, DOS, and Win32 editions. Crack is great for older systems using crypt(), but John the Ripper is better for newer systems using MD5 and similar password formats.

Frequency Analysis	189
Ciphertext Relative Length Analysis	190
Similar Plaintext Analysis	190
Monoalphabetic Ciphers	191
Other Ways to Hide Information	191
XOR	191
UUEncode	195
Base64	195
Compression	197
Summary	199
Solutions Fast Track	200
Frequently Asked Questions	202

Chapter 7 Unexpected Input **205**

Introduction	206
Understanding Why Unexpected Data Is Dangerous	206
Finding Situations Involving Unexpected Data	208
Local Applications and Utilities	208
HTTP/HTML	208
Unexpected Data in SQL Queries	211
Application Authentication	215
Disguising the Obvious	220
Using Techniques to Find and Eliminate Vulnerabilities	221
Black-Box Testing	222
Discovering Network and System Problems	225
Use the Source	226
Untaint Data by Filtering It	227
Escaping Characters Is Not Always Enough	227
Perl	228
Cold Fusion/Cold Fusion	
Markup Language (CFML)	229
ASP	229
PHP	230
Protecting Your SQL Queries	231
Silently Removing versus Alerting on Bad Data	232
Invalid Input Function	232
Token Substitution	233
Utilizing the Available Safety Features in Your Programming Language	233

Understanding Why Unexpected Data Is Dangerous

- ☑ Almost all applications interact with the user, and thus take data from them.
- ☑ An application can't assume that the user is playing by the rules.
- ☑ The application has to be wary of buffer overflows, logic alteration, and the validity of data passed to system functions.

Perl	233
PHP	235
ColdFusion/ColdFusion Markup Language	235
ASP	236
MySQL	237
Using Tools to Handle Unexpected Data	237
Web Sleuth	237
CGIAudit	237
RATS	237
Flawfinder	238
Retina	238
Hailstorm	238
Pudding	238
Summary	239
Solutions Fast Track	239
Frequently Asked Questions	242

Damage & Defense...

Understanding Assembly Language

There are a few specific pieces of assembly language knowledge that are necessary to understand the stack. One thing that is required is to understand the normal usage of *registers* in a stack:

- **EIP** The extended instruction pointer.
- **ESP** The extended stack pointer.
- **EBP** The extended base pointer.

Chapter 8 Buffer Overflow 243

Introduction	244
Understanding the Stack	244
The Code	246
Disassembly	247
The Stack Dump	248
Oddities and the Stack	249
Understanding the Stack Frame	249
Introduction to the Stack Frame	250
Passing Arguments to a Function:	
A Sample Program	250
The Disassembly	251
The Stack Dumps	254
Stack Frames and Calling Syntaxes	256
Learning about Buffer Overflows	257
A Simple Uncontrolled Overflow:	
A Sample Program	259
The Disassembly	260
The Stack Dumps	262
Creating Your First Overflow	263
Creating a Program with an Exploitable Overflow	264
Writing the Overflowable Code	264
Disassembling the Overflowable Code	265
Stack Dump after the Overflow	267
Performing the Exploit	267

General Exploit Concepts	268
Buffer Injection Techniques	268
Methods to Execute Payload	269
Designing Payload	281
Performing the Exploit on Linux	282
Performing the Exploit on Windows NT	293
Learning Advanced Overflow Techniques	303
Input Filtering	303
Incomplete Overflows and Data Corruption	304
Stack Based Function Pointer Overwrite	306
Heap Overflows	306
Corrupting a Function Pointer	307
Trespassing the Heap	307
Advanced Payload Design	310
Using What You Already Have	310
Dynamic Loading New Libraries	311
Eggshell Payloads	313
Summary	314
Solutions Fast Track	314
Frequently Asked Questions	317



Q: How can I eliminate or minimize the risk of unknown format string vulnerabilities in programs on my system?

A: A good start is having a sane security policy. Rely on the least-privileges model, ensure that only the most necessary utilities are installed setuid and can be run only by members of a trusted group. Disable or block access to all services that are not completely necessary.

Chapter 9 Format Strings **319**

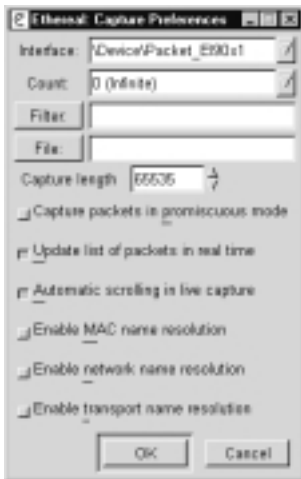
Introduction	320
Understanding Format String Vulnerabilities	322
Why and Where Do Format String Vulnerabilities Exist?	326
How Can They Be Fixed?	327
How Format String Vulnerabilities Are Exploited	328
Denial of Service	329
Reading Memory	329
Writing to Memory	330
How Format String Exploits Work	332
Constructing Values	333
What to Overwrite	335
Overwriting Return Addresses	335
Overwriting Global Offset Table Entries and Other Function Pointers	335
Examining a Vulnerable Program	336
Testing with a Random Format String	340
Writing a Format String Exploit	344

Summary 356
 Solutions Fast Track 356
 Frequently Asked Questions 358

Chapter 10 Sniffing 361

Introduction 362
 What Is Sniffing? 362
 How Does It Work? 362
 What to Sniff? 363
 Obtaining Authentication Information 363
 Monitoring Telnet (Port 23) 364
 Monitoring FTP (Port 21) 364
 Monitoring POP (Port 110) 365
 Monitoring IMAP (Port 143) 365
 Monitoring NNTP (Port 119) 366
 Monitoring rexec (Port 512) 366
 Monitoring rlogin (Port 513) 367
 Monitoring X11 (Port 6000+) 368
 Monitoring NFS File Handles 368
 Capturing Windows NT Authentication Information 369
 Capturing Other Network Traffic 370
 Monitoring SMTP (Port 25) 370
 Monitoring HTTP (Port 80) 370

Ethereal Capture Preferences



Popular Sniffing Software 371
 Ethereal 371
 Network Associates Sniffer Pro 372
 NT Network Monitor 374
 WildPackets 375
 TCPDump 376
 dsniff 377
 Ettercap 380
 Esniff.c 380
 Sniffit 381
 Carnivore 382
 Additional Resources 385
 Advanced Sniffing Techniques 385
 Man-in-the-Middle (MITM) Attacks 385
 Cracking 386
 Switch Tricks 386
 ARP Spoofing 386
 MAC Flooding 387
 Routing Games 388

Exploring Operating System APIs	388
Linux	388
BSD	392
libpcap	392
Windows	395
Taking Protective Measures	395
Providing Encryption	395
Secure Shell (SSH)	396
Secure Sockets Layers (SSL)	397
PGP and S/MIME	397
Switching	398
Employing Detection Techniques	398
Local Detection	398
Network Detection	399
DNS Lookups	399
Latency	399
Driver Bugs	400
AntiSniff	400
Network Monitor	400
Summary	401
Solutions Fast Track	402
Frequently Asked Questions	404
Chapter 11 Session Hijacking	407
Introduction	408
Understanding Session Hijacking	408
TCP Session Hijacking	410
TCP Session Hijacking with Packet	
Blocking	411
Route Table Modification	411
ARP Attacks	414
UDP Hijacking	415
Examining the Available Tools	416
Juggernaut	416
Hunt	420
Ettercap	425
SMBRelay	430
Storm Watchers	430
ACK Storms	431
Playing MITM for Encrypted Communications	433
Man-in-the-Middle Attacks	434
Dsniff	435
Other Hijacking	436

Understanding Session Hijacking

- The point of hijacking a connection is to steal trust.
- Hijacking is a race scenario: Can the attacker get an appropriate response packet in before the legitimate server or client can?
- Attackers can remotely modify routing tables to redirect packets or get a system into the routing path between two hosts.

Summary	438
Solutions Fast Track	438
Frequently Asked Questions	440

Chapter 12 Spoofing: Attacks on Trusted Identity **443**

Introduction	444
What It Means to Spoof	444
Spoofing Is Identity Forgery	444
Spoofing Is an Active Attack	
against Identity Checking Procedures	445
Spoofing Is Possible at All	
Layers of Communication	445
Spoofing Is Always Intentional	446
Spoofing May Be Blind or Informed,	
but Usually Involves Only Partial	
Credentials	447
Spoofing Is Not the Same Thing as Betrayal	448
Spoofing Is Not Necessarily Malicious	448
Spoofing Is Nothing New	449
Background Theory	449
The Importance of Identity	450
The Evolution of Trust	451
Asymmetric Signatures between Human	
Beings	451
Establishing Identity within Computer	
Networks	453
Return to Sender	454
In the Beginning, There Was...	
a Transmission	455
Capability Challenges	457
Ability to Transmit: "Can It Talk	
to Me?"	457
Ability to Respond: "Can It Respond	
to Me?"	459
Ability to Encode: "Can It Speak My	
Language?"	463
Ability to Prove a Shared Secret:	
"Does It Share a Secret with Me?"	465
Ability to Prove a Private Keypair:	
"Can I Recognize Your Voice?"	467

Tools & Traps...

Perfect Forward Secrecy: SSL's Dirty Little Secret

The dirty little secret of SSL is that, unlike SSH and unnecessarily like standard PGP, its standard modes are *not* perfectly forward secure. This means that an attacker can lie in wait, sniffing encrypted traffic at its leisure for as long as it desires, until one day it breaks in and steals the SSL private key used by the SSL engine (which is extractable from all but the most custom hardware).

Ability to Prove an Identity Keypair: “Is Its Identity Independently Represented in My Keypair?”	468
Configuration Methodologies: Building a Trusted Capability Index	470
Local Configurations vs. Central Configurations	470
Desktop Spoofs	471
The Plague of Auto-Updating Applications	471
Impacts of Spoofs	473
Subtle Spoofs and Economic Sabotage	474
Flattery Will Get You Nowhere	474
Subtlety Will Get You Everywhere	476
Selective Failure for Selecting Recovery	476
Bait and Switch: Spoofing the Presence of SSL Itself	478
Down and Dirty: Engineering Spoofing Systems	486
Spitting into the Wind: Building a Skeleton Router in Userspace	486
Designing the Nonexistent: The Network Card That Didn’t Exist but Responded Anyway	487
Implementation: DoxRoute, Section by Section	488
Bring Out the Halon: Spoofing Connectivity Through Asymmetric Firewalls	510
Symmetric Outgoing TCP: A Highly Experimental Framework for Handshake-Only TCP Connection Brokering	511
Summary	518
Solution Fast Track	519
Frequently Asked Questions	523
Chapter 13 Tunneling	527
Introduction	528
Strategic Constraints of Tunnel Design	530
Privacy: “Where Is My Traffic Going?”	532
Routability: “Where Can This Go Through?”	532
Deployability: “How Painful Is This to Get Up and Running?”	533
Flexibility: “What Can We Use This for, Anyway?”	534

Quality: “How Painful Will This System Be to Maintain?”	537
Designing End-to-End Tunneling Systems	537
Drilling Tunnels Using SSH	538
Security Analysis: OpenSSH 3.02	539
Setting Up OpenSSH	541
Open Sesame: Authentication	543
Basic Access: Authentication by Password	543
Transparent Access: Authentication by Private Key	544
Server to Client Authentication	544
Client to Server Authentication	545
Command Forwarding: Direct	
Execution for Scripts and Pipes	550
Port Forwarding: Accessing Resources on Remote Networks	556
Local Port Forwards	557
Dynamic Port Forwards	560
Internet Explorer 6: Making the Web Safe for Work	561
Speak Freely: Instant Messaging over SSH	564
That’s a Wrap: Encapsulating Arbitrary Win32 Apps within the Dynamic Forwarder	566
Summoning Virgil: Using Dante’s Socksify to Wrap UNIX Applications	567
Remote Port Forwards	569
When in Rome: Traversing the Recalcitrant Network	571
Crossing the Bridge: Accessing Proxies through ProxyCommands	571
No Habla HTTP? Permuting thy Traffic	575
Show Your Badge: Restricted Bastion Authentication	576
Bringing the Mountain: Exporting SSHD Access	579
Echoes in a Foreign Tongue: Cross-Connecting Mutually Firewalled Hosts	581
Not In Denver, Not Dead: Now What? Standard File Transfer over SSH	584

**Primary questions for
privacy of
communications
include the following:**

- Can anyone else monitor the traffic within this tunnel? Read access, addressed by encryption.
- Can anyone else modify the traffic within this tunnel, or surreptitiously gain access to it? Write access, addressed primarily through authentication.

Incremental File Transfer over SSH	586
CD Burning over SSH	589
Acoustic Tubing: Audio	
Distribution over TCP and SSH	593
Summary	598
Solutions Fast Track	600
Frequently Asked Questions	606

Chapter 14 Hardware Hacking 609

Introduction	610
Understanding Hardware Hacking	610
Opening the Device: Housing	
and Mechanical Attacks	611
Types of Tamper Mechanisms	613
Tamper Resistance	615
Tamper Evidence	615
Tamper Detection	615
Tamper Response	617
External Interfaces	618
Protocol Analysis	620
Electromagnetic Interference	
and Electrostatic Discharge	623
Analyzing the Product Internals: Electrical	
Circuit Attacks	624
Reverse-engineering the Device	624
Basic Techniques: Common Attacks	627
Device Packaging	627
Memory Retrieval	628
Timing Attacks	629
Advanced Techniques: Epoxy	
Removal and IC Delidding	630
Silicon Die Analysis	631
Cryptanalysis and Obfuscation Methods	632
What Tools Do I Need?	634
Starter Kit	634
Advanced Kit	635
Example: Hacking the iButton Authentication	
Token	637
Experimenting with the Device	638
Reverse-engineering the “Random”	
Response	639
Example: Hacking the NetStructure 7110	
E-commerce Accelerator	642

Understanding Hardware Hacking



Hardware hacking is done for the following reasons:

- General analysis of the product to determine common security weaknesses and attacks
- Access to the internal circuit without evidence of device tampering
- Retrieval of any internal or secret data components
- Cloning of the device
- Retrieving memory contents
- Elevation of privilege

Opening the Device	642
Retrieving the Filesystem	642
Reverse-engineering the Password Generator	646
Summary	648
Solutions Fast Track	649
Frequently Asked Questions	652

Chapter 15 Viruses, Trojan Horses, and Worms 655

Introduction	656
How Do Viruses, Trojans Horses, and Worms Differ?	656
Viruses	656
Worms	657
Macro Virus	658
Trojan Horses	659
Hoaxes	660
Anatomy of a Virus	660
Propagation	660
Payload	662
Other Tricks of the Trade	663
Dealing with Cross-platform Issues	664
Java	664
Macro Viruses	665
Recompilation	665
Shockwave Flash	665
Proof that We Need to Worry	665
The Morris Worm	666
ADMw0rm	666
Melissa and I Love You	666
Sadmin Worm	673
Code Red Worms	674
Nimda Worm	675
Creating Your Own Malware	677
New Delivery Methods	678
Faster Propagation Methods	679
Other Thoughts on Creating New Malware	679
How to Secure Against Malicious Software	680
Anti-Virus Software	681
Updates and Patches	683
Web Browser Security	683
Anti-Virus Research	683

A “worm” is a program that can run independently, will consume the resources of its host from within in order to maintain itself, and can propagate a complete working version of itself on to other machines.

Summary	685
Solutions Fast Track	685
Frequently Asked Questions	687

Chapter 16 IDS Evasion **689**

Introduction	690
Understanding How Signature-Based IDSs Work	690
Judging False Positives and Negatives	693
Alert Flooding	693
Using Packet Level Evasion	694
IP Options	696
Time-To-Live Attacks	696
IP Fragmentation	697
TCP Header	698
TCP Synchronization	699
TCB Creation	699
Stream Reassembly	700
TCB Teardown	701
Using Fragrouter and Congestant	701
Countermeasures	704
Using Application Protocol Level Evasion	705
Security as an Afterthought	705
Evading a Match	706
Alternate Data Encodings	706
Web Attack Techniques	707
Method Matching	708
Directory and File Referencing	708
Countermeasures	709
Using Code Morphing Evasion	709
Summary	713
Solutions Fast Track	714
Frequently Asked Questions	716

Tools & Traps...

Baiting with Honeynets

Recently, there has been an upsurge in the use of honeynets as a defensive tool. A *honeynet* is a system that is deployed with the intended purpose of being compromised. These are hyper defensive tools that can be implemented at any location inside a network. The current best known configuration type for these tools is where two systems are deployed, one for the bait, the other configured to log all traffic.

Chapter 17 Automated Security Review and Attack Tools **719**

Introduction	720
Learning about Automated Tools	720
Exploring the Commercial Tools	725
CyberCop Scanner	728
Internet Security Systems (ISS)	
Internet Scanner	728
BindView's BV-Control for Internet Security	729
eEye Retina	729

**Vulnerability Scanners
by Number**

Product	Vulnerability Count
ISS Internet Scanner	976
NAI CyberCop Scanner	830
BV Control for Internet Security	900
Harris STAT Scanner	1,200
Symantec NetRecon	600
eEye Retina	820

**Deciding How Much
Detail to Publish**

- Take great care in deciding whether or not you want to provide exploit code with your NSF report.
- You must be prepared to take a slight risk when reporting security flaws. You could end up facing the vendor's wrath.
- Be extra cautious in describing any security flaw that requires the circumvention of a vendor's copyright protection mechanisms.

Other Products	729
Exploring the Free Tools	730
Nessus	730
Security Administrators	
Integrated Network Tool (SAINT)	731
Security Administrators Research	
Assistant (SARA)	732
ShadowScan	732
Nmap and NmapNT	732
Whisker	733
VLAD the Scanner	733
Other Resources	734
Using Automated Tools for Penetration Testing	734
Testing with the Commercial Tools	734
Testing the Free Tools	739
Knowing When Tools Are Not Enough	743
The New Face of Vulnerability Testing	744
Summary	745
Solutions Fast Track	745
Frequently Asked Questions	746

Chapter 18 Reporting Security Problems 749

Introduction	750
Understanding Why Security	
Problems Need to Be Reported	750
Full Disclosure	752
Determining When and to	
Whom to Report the Problem	755
Whom to Report Security Problems to?	755
How to Report a Security Problem	
to a Vendor	758
Deciding How Much Detail to Publish	759
Publishing Exploit Code	759
Problems	760
Repercussions from Vendors	760
Reporting Errors	762
Risk to the Public	762
Summary	763
Solutions Fast Track	763
Frequently Asked Questions	765

Index**767**