

Computer Security

A Hands-on Approach

Wenliang Du
Syracuse University

Contents

Preface	xv
About the Author	xix
Acknowledgments	xxi
I Software Security	1
1 Set-UID Programs	5
1.1 The Need for Privileged Programs	6
1.1.1 The Password Dilemma	6
1.1.2 Different Types of Privileged Programs	7
1.2 The Set-UID Mechanism	8
1.2.1 A Superman Story	8
1.2.2 How It Works	8
1.2.3 An Example of Set-UID Program	9
1.2.4 How to Ensure Its Security	10
1.2.5 The Set-GID Mechanism	10
1.3 What Can Go Wrong: What Happened to Superman	11
1.4 Attack Surfaces of Set-UID Programs	12
1.4.1 User Inputs: Explicit Inputs	12
1.4.2 System Inputs	13
1.4.3 Environment Variables: Hidden Inputs	13
1.4.4 Capability Leaking	14
1.5 Invoking Other Programs	16
1.5.1 Unsafe Approach: Using <code>system()</code>	16
1.5.2 Safe Approach: Using <code>execve()</code>	18
1.5.3 Invoking External Commands in Other Languages	19
1.5.4 Lessons Learned: Principle of Isolation	20
1.6 Principle of Least Privilege	20
1.7 Summary	21
2 Environment Variables and Attacks	23
2.1 Environment Variables	24
2.1.1 How to Access Environment Variables	24
2.1.2 How a Process Gets Its Environment Variables	25

2.1.3	Memory Location for Environment Variables	26
2.1.4	Shell Variables and Environment Variables	27
2.2	Attack Surface	30
2.3	Attacks via Dynamic Linker	32
2.3.1	Static and Dynamic Linking	32
2.3.2	Case Study: LD_PRELOAD and LD_LIBRARY_PATH	34
2.3.3	Case Study: OS X Dynamic Linker	36
2.4	Attack via External Program	37
2.4.1	Two Typical Ways to Invoke External Programs	37
2.4.2	Case Study: the PATH environment variable	37
2.4.3	Reduce Attack Surface	38
2.5	Attack via Library	38
2.5.1	Case Study - Locale in UNIX	39
2.6	Application Code	39
2.6.1	Case Study - Using getenv() in Application Code	40
2.7	Set-UID Approach versus Service Approach	41
2.8	Summary	42
3	Shellshock Attack	43
3.1	Background: Shell Functions	44
3.2	The Shellshock Vulnerability	46
3.2.1	The Shellshock Bug	46
3.2.2	Mistake in the Bash Source Code	46
3.2.3	Exploiting the Shellshock vulnerability	48
3.3	Shellshock Attack on Set-UID Programs	48
3.4	Shellshock Attack on CGI Programs	49
3.4.1	Experiment Environment Setup	50
3.4.2	How Web Server Invokes CGI Programs	50
3.4.3	How Attacker Sends Data to Bash	52
3.4.4	Launching the Shellshock Attack	52
3.4.5	Creating Reverse Shell	53
3.5	Remote Attack on PHP	55
3.6	Summary	56
4	Buffer Overflow Attack	57
4.1	Program Memory Layout	58
4.2	Stack and Function Invocation	59
4.2.1	Stack Memory Layout	60
4.2.2	Frame Pointer	60
4.3	Stack Buffer-Overflow Attack	62
4.3.1	Copy Data to Buffer	62
4.3.2	Buffer Overflow	63
4.3.3	Exploiting a Buffer Overflow Vulnerability	64
4.4	Setup for Our Experiment	66
4.4.1	Disable Address Randomization	66
4.4.2	Vulnerable Program	66
4.5	Conduct Buffer-Overflow Attack	67
4.5.1	Finding the Address of the Injected Code	68

4.5.2	Improving Chances of Guessing	69
4.5.3	Finding the Address Without Guessing	69
4.5.4	Constructing the Input File	71
4.6	Writing a Shellcode	73
4.6.1	Writing Malicious Code Using C	73
4.6.2	Writing a Shellcode: Main Idea	74
4.6.3	Explanation of a Shellcode Example	74
4.7	Countermeasures: Overview	77
4.8	Address Randomization	79
4.8.1	Address Randomization on Linux	79
4.8.2	Effectiveness of Address Randomization	81
4.9	StackGuard	82
4.9.1	The Observation and the Idea	82
4.9.2	Manually Adding Code to Function	83
4.9.3	StackGuard Implementation in gcc	84
4.10	Summary	87
5	Return-to-libc Attack	89
5.1	Introduction	90
5.2	The Attack Experiment: Setup	91
5.3	Launch the Return-to-libc Attack: Part I	93
5.3.1	Task A: Find the Address of the <code>system()</code> Function	93
5.3.2	Task B: Find the Address of the String <code>"/bin/sh"</code>	94
5.4	Launch the Return-to-libc Attack: Part II	95
5.4.1	Function Prologue	96
5.4.2	Function Epilogue	97
5.4.3	Function Prologue and Epilogue Example	97
5.4.4	Perform Task C	98
5.4.5	Construct Malicious Input	99
5.4.6	Launch the Attack	100
5.5	Summary	101
6	Format String Vulnerability	103
6.1	Functions with Variable Number of Arguments	104
6.1.1	How to Access Optional Arguments	104
6.1.2	How <code>printf()</code> Accesses Optional Arguments	106
6.2	Format String with Missing Optional Arguments	107
6.3	Vulnerable Program and Experiment Setup	109
6.4	Exploiting the Format String Vulnerability	110
6.4.1	Attack 1: Crash Program	110
6.4.2	Attack 2: Print out Data on the Stack	111
6.4.3	Attack 3: Change the Program's Data in the Memory	111
6.4.4	Attack 4: Change the Program's Data to a Specific Value	113
6.4.5	Attack 4 (Continuation): A Much Faster Approach	114
6.4.6	Attack 5: Inject Malicious Code	116
6.4.7	Reducing the Size of Format String	118
6.5	Countermeasures	119
6.5.1	Developer	119

6.5.2	Compiler	119
6.5.3	Address Randomization	120
6.6	Summary	120
7	Race Condition Vulnerability	123
7.1	The General Race Condition Problem	124
7.2	Race Condition Vulnerability	125
7.3	Experiment Setup	127
7.4	Exploiting Race Condition Vulnerabilities	128
7.4.1	Choose a Target File	128
7.4.2	Launch Attack	129
7.4.3	Monitor the Result	130
7.4.4	Running the Exploit	130
7.5	Countermeasures	131
7.5.1	Atomic Operation	131
7.5.2	Repeating Check and Use	132
7.5.3	Sticky Symlink Protection	133
7.5.4	Principle of Least Privilege	135
7.6	Summary	136
8	Dirty COW	137
8.1	Memory Mapping using <code>mmap()</code>	138
8.2	<code>MAP_SHARED</code> , <code>MAP_PRIVATE</code> and Copy On Write	139
8.3	Discard the Copied Memory	141
8.4	Mapping Read-Only Files	141
8.5	The Dirty COW Vulnerability	143
8.6	Exploiting the Dirty COW Vulnerability	144
8.6.1	Selecting <code>/etc/passwd</code> as Target File	145
8.6.2	Set Up the Memory Mapping and Threads	145
8.6.3	The <code>write</code> Thread	146
8.6.4	The <code>madvise</code> Thread	147
8.6.5	The Attack Result	147
8.7	Summary	148
II	Web Security	149
9	Cross Site Request Forgery	153
9.1	Cross-Site Requests and Its Problems	154
9.2	Cross-Site Request Forgery Attack	155
9.3	CSRF Attacks on HTTP GET Services	156
9.3.1	HTTP GET and POST Services	156
9.3.2	The Basic Idea of CSRF Attacks	157
9.3.3	Attack on Elgg's Add-friend Service	157
9.4	CSRF Attacks on HTTP POST Services	159
9.4.1	Constructing a POST Request Using JavaScript	159
9.4.2	Attack on Elgg's Edit-Profile Service	160
9.5	Countermeasures	162

9.5.1	Using the referer Header	163
9.5.2	Same-Site Cookies	163
9.5.3	Secret Token	163
9.5.4	Case Study: Elgg's Countermeasures	164
9.6	Summary	166
10	Cross-Site Scripting Attack	167
10.1	The Cross-Site Scripting Attack	168
10.1.1	Non-persistent (Reflected) XSS Attack	168
10.1.2	Persistent XSS Attack	170
10.1.3	What damage can XSS cause?	170
10.2	XSS Attacks in Action	171
10.2.1	Prelude: Injecting JavaScript Code	171
10.2.2	Use XSS Attacks to Befriend with Others	172
10.2.3	Use XSS Attacks to Change Other People's Profiles	175
10.3	Achieving Self-Propagation	177
10.3.1	Creating a Self-Propagating XSS Worm: the DOM Approach	178
10.3.2	Create a Self-Propagating Worm: the Link Approach	180
10.4	Preventing XSS attacks	181
10.5	Summary	182
11	SQL Injection Attack	183
11.1	A Brief Tutorial of SQL	184
11.1.1	Log in to MySQL	184
11.1.2	Create a Database	184
11.1.3	CREATE a Table	184
11.1.4	INSERT a Row	185
11.1.5	The SELECT Statement	185
11.1.6	WHERE Clause	186
11.1.7	UPDATE SQL Statement	187
11.1.8	Comments in SQL Statements	187
11.2	Interacting with Database in Web Application	188
11.2.1	Getting Data from User	188
11.2.2	Getting Data From Database	189
11.3	Launching SQL Injection Attacks	191
11.3.1	Attack Using cURL	192
11.3.2	Modify Database	192
11.3.3	Multiple SQL Statements	193
11.4	The Fundamental Cause	194
11.5	Countermeasures	197
11.5.1	Filtering and Encoding Data	197
11.5.2	Prepared Statement	197
11.6	Summary	199

III Network Security	201
12 Packet Sniffing and Spoofing	205
12.1 How Packets Are Received	206
12.1.1 Network Interface Card (NIC)	206
12.1.2 BSD Packet Filter (BPF)	207
12.2 Packet Sniffing	208
12.2.1 Receiving Packets Using Sockets	208
12.2.2 Packet Sniffing using Raw Sockets	209
12.2.3 Packet Sniffing Using the pcap API	211
12.2.4 Processing Captured Packet	212
12.3 Packet Spoofing	215
12.3.1 Sending Normal Packets Using Socket	215
12.3.2 Sending Spoofed Packets Using Raw Sockets	216
12.3.3 Constructing ICMP Packets	218
12.3.4 Constructing UDP Packets	219
12.4 Snooping: Sniffing and Spoofing	221
12.5 Endianness	223
12.6 Calculating Checksum	224
12.7 Summary	226
13 Attacks on the TCP Protocol	227
13.1 How the TCP Protocol Works	228
13.1.1 TCP Client Program	228
13.1.2 TCP Server Program	229
13.1.3 Data Transmission: Under the Hood	232
13.1.4 TCP Header	233
13.2 SYN Flooding Attack	234
13.2.1 TCP Three-Way Handshake Protocol	234
13.2.2 The SYN Flooding Attack	235
13.2.3 Launching the SYN Flooding Attack	236
13.2.4 Launching SYN Flooding Attacks Using Our Own Code	238
13.2.5 Countermeasure	239
13.3 TCP Reset Attack	240
13.3.1 Closing TCP Connections	240
13.3.2 How the Attack Works	241
13.3.3 Launching the TCP Reset Attack: Setup	241
13.3.4 TCP Reset Attack on Telnet connections	242
13.3.5 TCP Reset Attack on SSH connections	244
13.3.6 TCP Reset Attack on Video-Streaming Connections	244
13.4 TCP Session Hijacking Attack	245
13.4.1 TCP Session and Session Hijacking	246
13.4.2 Launching the TCP Session Hijacking Attack	247
13.4.3 What Happens to the Hijacked TCP Connection	249
13.4.4 Causing More Damage	251
13.4.5 Creating Reverse Shell	251
13.5 Summary	252

14 Firewall	255
14.1 Introduction	256
14.2 Types of Firewalls	257
14.2.1 Packet Filter	257
14.2.2 Stateful Firewall	258
14.2.3 Application/Proxy Firewall	258
14.3 Building a Simple Firewall using Netfilter	258
14.3.1 Writing Loadable Kernel Modules	259
14.3.2 Compiling Kernel Modules	260
14.3.3 Installing Kernel Modules	260
14.4 Netfilter	261
14.4.1 netfilter Hooks for IPv4	262
14.4.2 Implementing a Simple Packet Filter Firewall	262
14.5 The iptables Firewall in Linux	265
14.5.1 The structure of the iptables Firewall	265
14.5.2 Traversing Chains and Rule Matching	266
14.5.3 iptables Extensions	267
14.5.4 Building a Simple Firewall	268
14.6 Stateful Firewall using Connection Tracking	270
14.6.1 Stateful Firewall	270
14.6.2 The Connection Tracking Framework in Linux	271
14.6.3 Example: Set up a Stateful Firewall	272
14.7 Application/Proxy Firewall and Web Proxy	272
14.8 Evading Firewalls	273
14.8.1 Using SSH Tunneling to Evade Firewalls	273
14.8.2 Dynamic Port Forwarding	275
14.8.3 Using VPN to Evade Firewall	276
14.9 Summary	276
15 Domain Name System (DNS) and Attacks	279
15.1 DNS Hierarchy, Zones, and Servers	280
15.1.1 DNS Domain Hierarchy	280
15.1.2 DNS Zone	281
15.1.3 Authoritative Name Servers	282
15.1.4 The Organization of Zones on the Internet	282
15.2 DNS Query Process	283
15.2.1 Local DNS Files	284
15.2.2 Local DNS Server and the Iterative Query Process	285
15.3 Set Up DNS Server and Experiment Environment	287
15.3.1 Configure the User Machine	287
15.3.2 Configure the Local DNS server	288
15.3.3 Set Up Zones in the Local DNS Server	290
15.4 DNS Attacks: Overview	292
15.5 Local DNS Cache Poisoning Attack	294
15.6 Remote DNS Cache Poisoning Attack	296
15.6.1 The Kaminsky Attack	297
15.6.2 Construct the IP and UDP headers of DNS reply	299
15.6.3 Construct the DNS Header and Payload	300

15.6.4	Result Verification	301
15.7	Reply Forgery Attacks from Malicious DNS Servers	302
15.7.1	Fake Data in the Additional Section	302
15.7.2	Fake Data in the Authority Section	302
15.7.3	Using Both Sections	303
15.7.4	Fake Data in the Answer Section: for Reverse DNS Lookup	304
15.8	Protection Against DNS Cache Poisoning Attacks	306
15.8.1	DNSSEC	306
15.8.2	TLS/SSL Solution	306
15.9	Denial of Service Attacks on DNS Servers	308
15.9.1	Attacks on the Root and TLD Servers	308
15.9.2	Attacks on Nameservers of a Particular Domain	309
15.10	Summary	310
16	Virtual Private Network	311
16.1	Introduction	312
16.1.1	Virtual Private Network	312
16.1.2	How a Virtual Private Network Works	314
16.2	An Overview of How TLS/SSL VPN Works	315
16.2.1	Establishing A TLS/SSL Tunnel	316
16.2.2	Forwarding IP packets	316
16.2.3	Releasing IP Packets	317
16.3	How TLS/SSL VPN Works: Details	318
16.3.1	Virtual Network Interfaces	318
16.3.2	Creating a TUN Interface	319
16.3.3	Routing Packets to a TUN Interface	321
16.3.4	Reading and Writing Operations on the TUN Interface	322
16.3.5	Forwarding Packets via the Tunnel	322
16.3.6	Packet's Return Trip	323
16.4	Building a VPN	323
16.4.1	Establish the Tunnel	324
16.4.2	Monitoring File Descriptors	326
16.4.3	From TUN To Tunnel	326
16.4.4	From Tunnel to TUN	327
16.4.5	Bring Everything Together	327
16.5	Setting Up a VPN	328
16.5.1	Network Configuration	328
16.5.2	Testing VPN	330
16.6	Using VPN to Bypass Egress Firewall	332
16.6.1	Network Setup	332
16.6.2	Setting Up VPN to Bypass Firewall	334
16.7	Summary	335
17	The Heartbleed Bug and Attack	337
17.1	Background: the Heartbeat Protocol	338
17.2	Launch the Heartbleed Attack	340
17.2.1	Attack Environment and Setup	340
17.2.2	Launch an Attack	341

17.3	Fixing the Heartbleed Bug	343
17.4	Summary	343
18	Public Key Infrastructure	345
18.1	Attack on Public Key Cryptography	346
18.1.1	Man-in-the-Middle (MITM) Attack	346
18.1.2	Defeating MITM Attacks	347
18.1.3	Public Key Infrastructure	347
18.2	Public Key Certificates	348
18.2.1	X.509 Digital Certificate	348
18.2.2	Get Certificate from a Real Server	349
18.3	Certificate Authority (CA)	350
18.3.1	Being a CA	351
18.3.2	Getting X.509 Certificate from CA	352
18.3.3	Deploying Public Key Certificate in Web Server	354
18.3.4	Apache Setup for HTTPS	356
18.4	Root and Intermediate Certificate Authorities	356
18.4.1	Root CAs and Self-Signed Certificate	356
18.4.2	Intermediate CAs and Chain of Trust	357
18.4.3	Creating Certificates for Intermediate CA	358
18.4.4	Apache Setup	359
18.4.5	Trusted CAs in the Real World	359
18.5	How PKI Defeats the MITM Attack	360
18.5.1	Attacker Forwards the Authentic Certificate	361
18.5.2	Attacker Creates a Fake Certificate	361
18.5.3	Attacker Sends Its Own Certificate	361
18.5.4	The Man-In-The-Middle Proxy	362
18.6	Attacks on the Public-Key Infrastructure	363
18.6.1	Attack on CA's Verification Process	364
18.6.2	Attack on CA's Signing Process	365
18.6.3	Attacks on the Algorithms	365
18.6.4	Attacks on User Confirmation	366
18.7	Types of Digital Certificates	367
18.7.1	Domain Validated Certificates (DV)	367
18.7.2	Organizational Validated Certificates (OV)	368
18.7.3	Extended Validated Certificates (EV)	368
18.8	Summary	369
19	Transport Layer Security	371
19.1	Overview of TLS	372
19.2	TLS Handshake	373
19.2.1	Overview of the TLS Handshake Protocol	373
19.2.2	Certificate Verification	375
19.2.3	Key Generation and Exchange	375
19.3	TLS Data Transmission	377
19.3.1	Sending Data with TLS Record Protocol	377
19.3.2	Receiving Data with TLS Record Protocol	378
19.4	TLS Programming: A Client Program	379

19.4.1	The Overall Picture	379
19.4.2	TLS Initialization	380
19.4.3	TCP Connection Setup	381
19.4.4	TLS Handshake	382
19.4.5	Application Data Transmission	383
19.4.6	Set Up the Certificate Folder	383
19.4.7	The Complete Client Code	385
19.5	Verifying Server's Hostname	386
19.5.1	Modified Client Code	386
19.5.2	An Experiment: Man-In-The-Middle Attack	387
19.5.3	Hostname Checking	389
19.6	TLS Programming: the Server Side	390
19.6.1	TLS Setup	390
19.6.2	TCP Setup	392
19.6.3	TLS Handshake	392
19.6.4	TLS Data Transmission	393
19.6.5	Testing	394
19.7	Summary	394