

Computer & Internet Security

A Hands-on Approach

Second Edition

Wenliang Du

Syracuse University

Contents

Preface	xxi
About the Author	xxvii
Acknowledgments	xxix
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>	19
1.5.3 Invoking External Commands in Other Languages	20
1.5.4 Lessons Learned: Principle of Isolation	21
1.6 Principle of Least Privilege	21
1.7 Summary	22
2 Environment Variables and Attacks	25
2.1 Environment Variables	26
2.1.1 How to Access Environment Variables	26
2.1.2 How a Process Gets Its Environment Variables	27

2.1.3	Memory Location for Environment Variables	28
2.1.4	Shell Variables and Environment Variables	29
2.2	Attack Surface	32
2.3	Attacks via Dynamic Linker	33
2.3.1	Static and Dynamic Linking	34
2.3.2	Case Study: LD.PRELOAD and LD.LIBRARY.PATH	35
2.3.3	Case Study: OS X Dynamic Linker	38
2.4	Attack via External Program	38
2.4.1	Two Typical Ways to Invoke External Programs	39
2.4.2	Case Study: the PATH environment variable	39
2.4.3	Reduce Attack Surface	40
2.5	Attack via Library	41
2.5.1	Case Study - Locale in UNIX	41
2.6	Application Code	42
2.6.1	Case Study - Using <code>getenv()</code> in Application Code	42
2.7	Set-UID Approach versus Service Approach	43
2.8	Summary	44
3	Shellshock Attack	47
3.1	Background: Shell Functions	48
3.2	The Shellshock Vulnerability	50
3.2.1	Vulnerable Version of <code>bash</code>	50
3.2.2	The Shellshock Bug	50
3.2.3	Mistake in the Bash Source Code	51
3.2.4	Exploiting the Shellshock vulnerability	52
3.3	Shellshock Attack on Set-UID Programs	52
3.4	Shellshock Attack on CGI Programs	54
3.4.1	Experiment Environment Setup	54
3.4.2	How Web Server Invokes CGI Programs	54
3.4.3	How Attacker Sends Data to Bash	56
3.4.4	Launching the Shellshock Attack	56
3.4.5	Creating Reverse Shell	58
3.5	Remote Attack on PHP	60
3.6	Summary	61
4	Buffer Overflow Attack	63
4.1	Program Memory Layout	64
4.2	Stack and Function Invocation	65
4.2.1	Stack Memory Layout	65
4.2.2	Frame Pointer	66
4.3	Stack Buffer-Overflow Attack	67
4.3.1	Copy Data to Buffer	68
4.3.2	Buffer Overflow	69
4.3.3	Exploiting a Buffer Overflow Vulnerability	70
4.4	Setup for Our Experiment	71
4.4.1	Disable Address Randomization	72
4.4.2	Vulnerable Program	72
4.5	Conduct Buffer-Overflow Attack	73

4.5.1	Finding the Address of the Injected Code	73
4.5.2	Improving Chances of Guessing	74
4.5.3	Finding the Address Without Guessing	75
4.5.4	Constructing the Input File	76
4.6	Attacks with Unknown Address and Buffer Size	79
4.6.1	Knowing the Range of Buffer Size	79
4.6.2	Knowing the Range of the Buffer Address	79
4.6.3	A General Solution	80
4.7	Writing a Shellcode	82
4.7.1	Writing Malicious Code Using C	82
4.7.2	Writing a Shellcode: Main Idea	83
4.7.3	Explanation of a Shellcode Example	83
4.8	Countermeasures: Overview	86
4.9	Address Randomization	88
4.9.1	Address Randomization on Linux	89
4.9.2	Effectiveness of Address Randomization	90
4.10	StackGuard	91
4.10.1	The Observation and the Idea	92
4.10.2	Manually Adding Code to Function	92
4.10.3	StackGuard Implementation in gcc	94
4.11	Defeating the Countermeasure in <code>bash</code> and <code>dash</code>	96
4.12	Summary	98
5	Return-to-libc Attack and ROP	101
5.1	Introduction: Non-Executable Stack	102
5.2	The Attack Experiment: Setup	103
5.3	Launch the Return-to-libc Attack: Part I	105
5.3.1	Task A: Find the Address of the <code>system()</code> Function	105
5.3.2	Task B: Find the Address of the String <code>"/bin/sh"</code>	106
5.4	Launch the Return-to-libc Attack: Part II	107
5.4.1	Function Prologue	108
5.4.2	Function Epilogue	109
5.4.3	Function Prologue and Epilogue Example	110
5.4.4	Perform Task C	111
5.4.5	Construct Malicious Input	112
5.4.6	Launch the Attack	113
5.5	Return-Oriented Programming	114
5.5.1	Experiment Setup	114
5.5.2	Tracking the values of the <code>esp</code> and <code>ebp</code> registers	116
5.5.3	Chaining Function Calls Without Arguments	117
5.5.4	Chaining Function Calls With Arguments: Skipping Prologue	119
5.5.5	Chaining Function Calls With Arguments: via <code>leave</code> and <code>ret</code>	122
5.5.6	Chaining Function Calls With Zero in the Argument	126
5.5.7	Use the Chaining Technique to Get Root Shell	127
5.5.8	Further Generalization: Return-Oriented Programming	129
5.6	Summary	130

6	Format String Vulnerability	131
6.1	Functions with Variable Number of Arguments	132
6.1.1	How to Access Optional Arguments	132
6.1.2	How <code>printf()</code> Accesses Optional Arguments	134
6.2	Format String with Missing Optional Arguments	135
6.3	Vulnerable Program and Experiment Setup	137
6.4	Exploiting the Format String Vulnerability	138
6.4.1	Attack 1: Crash Program	138
6.4.2	Attack 2: Print out Data on the Stack	139
6.4.3	Attack 3: Change the Program's Data in the Memory	139
6.4.4	Attack 4: Change the Program's Data to a Specific Value	141
6.4.5	Attack 4 (Continuation): A Much Faster Approach	142
6.5	Code Injection Attack using Format String Vulnerability	144
6.5.1	The Revised Vulnerable Program	144
6.5.2	The Attack Strategy	146
6.5.3	The Attack Program	147
6.5.4	Reducing the Size of Format String	149
6.6	Countermeasures	151
6.6.1	Developer	151
6.6.2	Compiler	151
6.6.3	Address Randomization	152
6.7	Relationship with the Buffer-Overflow Attack	152
6.8	Summary	153
7	Race Condition Vulnerability	155
7.1	The General Race Condition Problem	156
7.2	Race Condition Vulnerability	157
7.3	Experiment Setup	159
7.4	Exploiting Race Condition Vulnerabilities	160
7.4.1	Choose a Target File	160
7.4.2	Launch Attack	161
7.4.3	Monitor the Result	162
7.4.4	Running the Exploit	163
7.5	Countermeasures	164
7.5.1	Atomic Operation	164
7.5.2	Repeating Check and Use	165
7.5.3	Sticky Symlink Protection	166
7.5.4	Principle of Least Privilege	167
7.6	Summary	169
8	Dirty COW	171
8.1	Memory Mapping using <code>mmap()</code>	172
8.2	<code>MAP_SHARED</code> , <code>MAP_PRIVATE</code> and Copy On Write	173
8.3	Discard the Copied Memory	175
8.4	Mapping Read-Only Files	175
8.5	The Dirty COW Vulnerability	177
8.6	Exploiting the Dirty COW Vulnerability	178
8.6.1	Selecting <code>/etc/passwd</code> as Target File	179

8.6.2	Set Up the Memory Mapping and Threads	179
8.6.3	The <code>write</code> Thread	180
8.6.4	The <code>madvise</code> Thread	181
8.6.5	The Attack Result	181
8.7	Summary	182
9	Reverse Shell	183
9.1	Introduction	184
9.2	File Descriptor and Redirection	184
9.2.1	File Descriptor	184
9.2.2	Standard IO Devices	186
9.2.3	Redirection	187
9.2.4	How To Implement Redirection	188
9.3	Redirecting Input/Output to a TCP Connection	189
9.3.1	Redirecting Output to a TCP Connection	189
9.3.2	Redirecting Input to a TCP Connection	190
9.3.3	Redirecting to TCP Connection From Shell	191
9.4	Reverse Shell	192
9.4.1	Redirecting the Standard Output	192
9.4.2	Redirecting the Standard Input	192
9.4.3	Redirecting the Standard Error	194
9.4.4	Code Injection	194
9.5	Summary	195
II	Web Security	197
10	Cross Site Request Forgery	201
10.1	Cross-Site Requests and Its Problems	202
10.2	Cross-Site Request Forgery Attack	203
10.3	CSRF Attacks on HTTP GET Services	204
10.3.1	HTTP GET and POST Services	204
10.3.2	The Basic Idea of CSRF Attacks	205
10.3.3	Attack on Elgg's Add-friend Service	205
10.4	CSRF Attacks on HTTP POST Services	207
10.4.1	Constructing a POST Request Using JavaScript	207
10.4.2	Attack on Elgg's Edit-Profile Service	208
10.5	Countermeasures	210
10.5.1	Using the <code>referer</code> Header	211
10.5.2	Same-Site Cookies	211
10.5.3	Secret Token	211
10.5.4	Case Study: Elgg's Countermeasures	212
10.6	Summary	212
11	Cross-Site Scripting Attack	215
11.1	The Cross-Site Scripting Attack	216
11.1.1	Non-persistent (Reflected) XSS Attack	217
11.1.2	Persistent XSS Attack	218

11.1.3	What damage can XSS cause?	218
11.2	XSS Attacks in Action	219
11.2.1	Prelude: Injecting JavaScript Code	219
11.2.2	Use XSS Attacks to Befriend with Others	220
11.2.3	Use XSS Attacks to Change Other People's Profiles	223
11.3	Achieving Self-Propagation	225
11.3.1	Creating a Self-Propagating XSS Worm: the DOM Approach	226
11.3.2	Create a Self-Propagating Worm: the Link Approach	228
11.4	Preventing XSS attacks	229
11.4.1	Getting Rid of Code from User Inputs	229
11.4.2	Defeating XSS Attacks using Content Security Policy	230
11.4.3	Experimenting with Content Security Policy	232
11.5	Summary	234
12	SQL Injection Attack	237
12.1	A Brief Tutorial of SQL	238
12.1.1	Log in to MySQL	238
12.1.2	Create a Database	238
12.1.3	CREATE a Table	238
12.1.4	INSERT a Row	239
12.1.5	The SELECT Statement	239
12.1.6	WHERE Clause	240
12.1.7	UPDATE SQL Statement	241
12.1.8	Comments in SQL Statements	241
12.2	Interacting with Database in Web Application	242
12.2.1	Getting Data from User	242
12.2.2	Getting Data From Database	243
12.3	Launching SQL Injection Attacks	245
12.3.1	Attack Using cURL	246
12.3.2	Modify Database	246
12.3.3	Multiple SQL Statements	247
12.4	The Fundamental Cause	248
12.5	Countermeasures	251
12.5.1	Filtering and Encoding Data	251
12.5.2	Prepared Statement	251
12.6	Summary	253
III	Hardware Security	255
13	Meltdown Attack	259
13.1	Introduction and Analogy	260
13.1.1	Analogy: The Microsoft Brainteaser Question	260
13.1.2	Stealing A Secret	261
13.1.3	Side Channels	261
13.2	Side Channel Attacks via CPU Cache	262
13.2.1	Time Difference When Accessing Cache v.s Memory	262
13.2.2	Using CPU Cache as a Side Channel	264

13.3	The Room Holding Secret: The Kernel	267
13.3.1	Secret Data in Kernel Space	267
13.3.2	The Guard: Preventing Direct Access to Kernel Memory	269
13.3.3	Avoid Getting Killed: Handling Error/Exceptions in C	269
13.4	Passing the Guard: Out-of-Order Execution by CPU	271
13.5	The Meltdown Attack	274
13.5.1	A Naïve Approach	274
13.5.2	Improve the Attack by Getting the Secret Data Cached	275
13.5.3	Improve the Attack Using Assembly Code	275
13.5.4	Improve the Attack Using Statistic Approach	276
13.6	Countermeasures	278
13.7	Summary	279
14	Spectre Attack	281
14.1	Introduction	282
14.2	Out-of-Order Execution and Branch Prediction	282
14.2.1	An Experiment	283
14.2.2	Experiment Results	285
14.3	The Spectre Attack	285
14.3.1	The Setup for the Experiment	286
14.3.2	The Program Used in the Experiment	287
14.4	Improve the Attack Using Statistic Approach	289
14.5	Spectre Variant and Mitigation	292
14.6	Summary	292
IV	Network Security	293
15	Packet Sniffing and Spoofing	297
15.1	How Packets Are Received	298
15.1.1	Network Interface Card (NIC)	298
15.1.2	BSD Packet Filter (BPF)	299
15.2	Packet Sniffing	300
15.2.1	Receiving Packets Using Sockets	300
15.2.2	Packet Sniffing using Raw Sockets	301
15.2.3	Packet Sniffing Using the pcap API	303
15.2.4	Processing Captured Packet	304
15.3	Packet Spoofing	307
15.3.1	Sending Normal Packets Using Socket	308
15.3.2	Sending Spoofed Packets Using Raw Sockets	309
15.3.3	Constructing ICMP Packets	311
15.3.4	Constructing UDP Packets	312
15.4	Sniffing and Then Spoofing	314
15.5	Sniffing and Spoofing Using Python and Scapy	316
15.5.1	Installing Scapy	316
15.5.2	A Simple Example	316
15.5.3	Packet Sniffing	317
15.5.4	Spoofing ICMP Packets	318

15.5.5	Spoofing UDP Packets	318
15.5.6	Sniffing and Then Spoofing	319
15.5.7	Sending and Receiving Packets	319
15.6	Spoofing Packets Using a Hybrid Approach	320
15.6.1	A Hybrid Approach	320
15.6.2	Constructing Packet Template Using Scapy	321
15.6.3	Modifying and Sending Packets Using C	321
15.7	Endianness	323
15.8	Calculating Checksum	324
15.9	Summary	326
16	Attacks on the TCP Protocol	327
16.1	How the TCP Protocol Works	328
16.1.1	TCP Client Program	328
16.1.2	TCP Server Program	329
16.1.3	Data Transmission: Under the Hood	332
16.1.4	TCP Header	333
16.2	SYN Flooding Attack	334
16.2.1	TCP Three-Way Handshake Protocol	334
16.2.2	The SYN Flooding Attack	335
16.2.3	Launching the SYN Flooding Attack	336
16.2.4	Launching SYN Flooding Attacks Using C Code	338
16.2.5	Countermeasure	340
16.3	TCP Reset Attack	341
16.3.1	Closing TCP Connections	341
16.3.2	How the Attack Works	342
16.3.3	Launching the TCP Reset Attack: Setup	342
16.3.4	TCP Reset Attack on Telnet connections	343
16.3.5	TCP Reset Attack on SSH connections	344
16.3.6	TCP Reset Attack on Video-Streaming Connections	345
16.4	TCP Session Hijacking Attack	347
16.4.1	TCP Session and Session Hijacking	347
16.4.2	Launching TCP Session Hijacking Attack	348
16.4.3	What Happens to the Hijacked TCP Connection	351
16.4.4	Causing More Damage	351
16.4.5	Creating Reverse Shell	352
16.5	Summary	354
17	Firewall	355
17.1	Introduction	356
17.2	Types of Firewalls	357
17.2.1	Packet Filter	357
17.2.2	Stateful Firewall	358
17.2.3	Application/Proxy Firewall	358
17.3	Building a Simple Firewall using Netfilter	358
17.3.1	Writing Loadable Kernel Modules	359
17.3.2	Compiling Kernel Modules	360
17.3.3	Installing Kernel Modules	360

17.4	Netfilter	361
17.4.1	netfilter Hooks for IPv4	362
17.4.2	Implementing a Simple Packet Filter Firewall	362
17.5	The iptables Firewall in Linux	365
17.5.1	The structure of the iptables Firewall	365
17.5.2	Traversing Chains and Rule Matching	366
17.5.3	iptables Extensions	367
17.5.4	Building a Simple Firewall	368
17.6	Stateful Firewall using Connection Tracking	371
17.6.1	Stateful Firewall	371
17.6.2	The Connection Tracking Framework in Linux	372
17.6.3	Example: Set up a Stateful Firewall	372
17.7	Application/Proxy Firewall and Web Proxy	373
17.8	Evading Firewalls	374
17.8.1	Using SSH Tunneling to Evade Firewalls	374
17.8.2	Dynamic Port Forwarding	375
17.8.3	Reverse SSH Tunneling	377
17.8.4	Using VPN to Evade Firewall	377
17.9	Summary	377
18	Domain Name System (DNS) and Attacks	379
18.1	DNS Hierarchy, Zones, and Servers	380
18.1.1	DNS Domain Hierarchy	380
18.1.2	DNS Zone	381
18.1.3	Authoritative Name Servers	382
18.1.4	The Organization of Zones on the Internet	382
18.2	DNS Query Process	384
18.2.1	Local DNS Files	384
18.2.2	Local DNS Server and the Iterative Query Process	385
18.3	Set Up DNS Server and Experiment Environment	387
18.3.1	Configure the User Machine	388
18.3.2	Configure the Local DNS server	388
18.3.3	Set Up Zones in the Local DNS Server	390
18.4	Constructing DNS Request and Reply Using Scapy	392
18.4.1	DNS Header	392
18.4.2	DNS Records	393
18.4.3	Example 1: Sending a DNS Query	394
18.4.4	Example 2: Implement a Simple DNS Server	395
18.5	DNS Attacks: Overview	397
18.6	Local DNS Cache Poisoning Attack	398
18.6.1	Launch DNS Cache Poisoning Attack	399
18.6.2	Targeting the Authority Section	401
18.7	Remote DNS Cache Poisoning Attack	402
18.7.1	The Kaminsky Attack	403
18.7.2	Construct the IP and UDP headers of DNS reply	405
18.7.3	Construct the DNS Header and Payload	406
18.7.4	Result Verification	408
18.8	Reply Forgery Attacks from Malicious DNS Servers	409

18.8.1	Fake Data in the Additional Section	409
18.8.2	Fake Data in the Authority Section	411
18.8.3	Fake Data in Both Authority and Additional Sections	412
18.8.4	Fake Data in the Answer Section	413
18.8.5	Fake Answer in Reverse DNS Lookup	413
18.9	DNS Rebinding Attack	415
18.9.1	How DNS Rebinding Attack Works	415
18.9.2	Attack Environment Setup	417
18.9.3	Set Up the User Machine	418
18.9.4	Emulating a Vulnerable IoT Device's Web Server	418
18.9.5	Set Up the Web Server on Attacker Computer	419
18.9.6	Setting Up the Malicious DNS Server	421
18.9.7	Launching the Attack	422
18.9.8	Defending Against DNS Rebinding Attack	424
18.10	Protection Against DNS Spoofing Attacks	424
18.10.1	DNSSEC	424
18.10.2	TLS/SSL Solution	425
18.11	Denial of Service Attacks on DNS Servers	426
18.11.1	Attacks on the Root and TLD Servers	426
18.11.2	Attacks on Nameservers of a Particular Domain	427
18.12	Summary	428
19	Virtual Private Network	429
19.1	Introduction	430
19.1.1	Virtual Private Network	430
19.1.2	How a Virtual Private Network Works	432
19.2	An Overview of How TLS/SSL VPN Works	433
19.2.1	Establishing A TLS/SSL Tunnel	434
19.2.2	Forwarding IP packets	434
19.2.3	Releasing IP Packets	435
19.3	How TLS/SSL VPN Works: Details	436
19.3.1	Virtual Network Interfaces	436
19.3.2	Creating a TUN Interface	437
19.3.3	Routing Packets to a TUN Interface	439
19.3.4	Reading and Writing Operations on the TUN Interface	440
19.3.5	Forwarding Packets via the Tunnel	441
19.3.6	Packet's Return Trip	441
19.4	Building a VPN	441
19.4.1	Establish the Tunnel	442
19.4.2	Monitoring File Descriptors	444
19.4.3	From TUN To Tunnel	444
19.4.4	From Tunnel to TUN	445
19.4.5	Bring Everything Together	445
19.5	Setting Up a VPN	446
19.5.1	Network Configuration	446
19.5.2	Configure VPN Server	448
19.5.3	Configure VPN Client	448
19.5.4	Configure Host V	448

19.6	Testing VPN	449
19.6.1	Fing Test	449
19.6.2	Telnet Test	450
19.7	Using VPN to Bypass Egress Firewall	451
19.7.1	Network Setup	451
19.7.2	Setting Up VPN to Bypass Firewall	452
19.8	Summary	453
20	The Heartbleed Bug and Attack	455
20.1	Background: the Heartbeat Protocol	456
20.2	Launch the Heartbleed Attack	458
20.2.1	Attack Environment and Setup	458
20.2.2	Launch an Attack	459
20.3	Fixing the Heartbleed Bug	461
20.4	Summary	461
V	Cryptography	463
21	Secret-Key Encryption	467
21.1	Introduction	468
21.2	Substitution Cipher	468
21.2.1	Monoalphabetic Substitution Cipher	468
21.2.2	Breaking Monoalphabetic Substitution Cipher	469
21.2.3	Polyalphabetic Substitution Cipher	472
21.2.4	The Enigma Machine	473
21.3	DES and AES Encryption Algorithms	475
21.3.1	DES: Data Encryption Standard	475
21.3.2	AES: Advanced Encryption Standard	476
21.4	Encryption Modes	476
21.4.1	Encryption Modes	477
21.4.2	Electronic Codebook (ECB) Mode	478
21.4.3	Cipher Block Chaining (CBC) Mode	478
21.4.4	Cipher Feedback (CFB) Mode	480
21.4.5	Output Feedback (OFB) Mode	481
21.4.6	Counter (CTR) Mode	482
21.4.7	Modes for Authenticated Encryption	483
21.4.8	Padding	484
21.5	Initialization Vector and Common Mistakes	485
21.5.1	Common Mistake: Use the Same IV	485
21.5.2	Common Mistake: Use a Predictable IV	488
21.6	Programming using Cryptography APIs	491
21.7	Authenticated Encryption and the GCM Mode	493
21.7.1	The GCM Mode	494
21.7.2	Programming using the GCM Mode	495
21.8	Summary	496

22 One-Way Hash Function	497
22.1 Introduction	498
22.2 Concept and Properties	498
22.2.1 Cryptographic Properties	498
22.2.2 Replay the Number Game	499
22.3 Algorithms and Programs	499
22.3.1 The MD (Message Digest) Series	500
22.3.2 The SHA (Secure Hash Algorithm) Series	500
22.3.3 How Hash Algorithm Works	501
22.3.4 One-Way Hash Commands	501
22.3.5 Computing One-Way Hash in Programs	502
22.3.6 Performance of One-Way Hash Functions	504
22.4 Applications of One-Way Hash Functions	504
22.4.1 Integrity Verification	505
22.4.2 Committing a Secret Without Telling It	505
22.4.3 Password Verification	506
22.4.4 Trusted Timestamping	508
22.5 Message Authentication Code (MAC)	509
22.5.1 Constructing MAC and Potential Attacks	510
22.5.2 Launching the Length Extension Attack	511
22.5.3 Case Study: Length Extension Attack on Flickr	514
22.5.4 The Keyed-Hash MAC (HMAC) Algorithm	514
22.6 Blockchain and Bitcoins	515
22.6.1 Hash Chain and Blockchain	515
22.6.2 Make Chaining Difficult	516
22.6.3 Adding Incentives and Bitcoin	518
22.7 Hash Collision Attacks	519
22.7.1 Security Impact of Collision Attacks	519
22.7.2 Generating Two Different Files with the Same MD5 Hash	520
22.7.3 Generating Two Programs with the Same MD5 Hash	522
22.7.4 Making the Two Programs Behave Differently	525
22.7.5 Hash-Colliding X.509 Certificates	527
22.8 Summary	528
23 Public Key Cryptography	529
23.1 Introduction	530
23.2 Diffie-Hellman Key Exchange	530
23.2.1 Diffie-Hellman Key Exchange	531
23.2.2 Turn DH Key Exchange into a Public-Key Encryption Algorithm	532
23.3 The RSA Algorithm	533
23.3.1 Math Background: Modulo Operation	534
23.3.2 Math Background: Euler's Theorem	534
23.3.3 Math Background: Extended Euclidean Algorithm	535
23.3.4 The RSA Algorithm	536
23.3.5 Exercise: Small Number	537
23.3.6 Exercise: Large Number	538
23.3.7 Performance	540
23.3.8 Hybrid Encryption	541

23.3.9 Other Public-Key Encryption Algorithms	541
23.4 Using OpenSSL Tools to Conduct RSA Operations	542
23.4.1 Generating RSA keys	542
23.4.2 Extracting the public key	543
23.4.3 Encryption and Decryption	544
23.5 Paddings for RSA	544
23.5.1 Attacks Against Textbook RSA	545
23.5.2 Paddings: PKCS#1 v1.5 and OAEP	545
23.6 Digital Signature	546
23.6.1 Digital Signature using RSA	547
23.6.2 DSA and Other Digital Signature Algorithms	549
23.7 Programming using Public-Key Cryptography APIs	549
23.7.1 Key Generation	550
23.7.2 Encryption and Decryption	550
23.7.3 Digital Signature	552
23.8 Applications	554
23.8.1 Authentication	554
23.8.2 HTTPS and TLS/SSL	556
23.8.3 Chip Technology Used in Credit Cards	556
23.9 Blockchain and Bitcoins	558
23.10 Summary and Further Learning	558
24 Public Key Infrastructure	561
24.1 Attack on Public Key Cryptography	562
24.1.1 Man-in-the-Middle (MITM) Attack	562
24.1.2 Defeating MITM Attacks	563
24.1.3 Public Key Infrastructure	563
24.2 Public Key Certificates	564
24.2.1 X.509 Digital Certificate	564
24.2.2 Get Certificate from a Real Server	565
24.3 Certificate Authority (CA)	566
24.3.1 Being a CA	567
24.3.2 Getting X.509 Certificate from CA	568
24.3.3 Deploying Public Key Certificate in Web Server	571
24.3.4 Apache Setup for HTTPS	572
24.4 Root and Intermediate Certificate Authorities	573
24.4.1 Root CAs and Self-Signed Certificate	573
24.4.2 Intermediate CAs and Chain of Trust	574
24.4.3 Creating Certificates for Intermediate CA	575
24.4.4 Apache Setup	576
24.4.5 Trusted CAs in the Real World	576
24.5 How PKI Defeats the MITM Attack	577
24.5.1 Attacker Forwards the Authentic Certificate	577
24.5.2 Attacker Creates a Fake Certificate	577
24.5.3 Attackers Send Their Own Certificates	578
24.5.4 The Man-In-The-Middle Proxy	579
24.6 Attacks on the Public-Key Infrastructure	580
24.6.1 Attack on CA's Verification Process	581

24.6.2	Attack on CA's Signing Process	581
24.6.3	Attacks on the Algorithms	582
24.6.4	Attacks on User Confirmation	583
24.7	Types of Digital Certificates	583
24.7.1	Domain Validated Certificates (DV)	584
24.7.2	Organizational Validated Certificates (OV)	584
24.7.3	Extended Validated Certificates (EV)	585
24.8	Summary	585
25	Transport Layer Security	587
25.1	Overview of TLS	588
25.2	TLS Handshake	589
25.2.1	Overview of the TLS Handshake Protocol	589
25.2.2	Certificate Verification	591
25.2.3	Key Generation and Exchange	591
25.3	TLS Data Transmission	593
25.3.1	Sending Data with TLS Record Protocol	593
25.3.2	Receiving Data with TLS Record Protocol	594
25.4	TLS Programming: A Client Program	595
25.4.1	The Overall Picture	596
25.4.2	TLS Initialization	596
25.4.3	TCP Connection Setup	598
25.4.4	TLS Handshake	598
25.4.5	Application Data Transmission	599
25.4.6	Set Up the Certificate Folder	600
25.4.7	The Complete Client Code	601
25.5	Verifying Server's Hostname	602
25.5.1	Modified Client Code	602
25.5.2	An Experiment: Man-In-The-Middle Attack	604
25.5.3	Hostname Checking	605
25.6	TLS Programming: the Server Side	607
25.6.1	TLS Setup	607
25.6.2	TCP Setup	609
25.6.3	TLS Handshake	609
25.6.4	TLS Data Transmission	611
25.6.5	Testing	611
25.7	Summary	612
26	Bitcoin and Blockchain	615
26.1	History	616
26.2	Cryptography Foundation and Bitcoin Address	617
26.2.1	Generating Private and Public Keys	617
26.2.2	Turning Hash Value Into Bitcoin Address	619
26.2.3	Wallet	622
26.3	Transactions	622
26.3.1	The "Safe" Analogy	623
26.3.2	An Example	624
26.3.3	Input	625

26.3.4	Output	626
26.4	Unlocking the Output of a Transaction	627
26.4.1	Some Fun but Non-standard Locks	628
26.4.2	Pay-to-Pubkey-Hash Type (P2PH)	630
26.4.3	Pay-to-Multisig (P2MS)	631
26.4.4	Pay-to-ScriptHash (P2SH)	632
26.4.5	P2SH Example: Multi-Signature	633
26.4.6	Case Study: A Real Transaction	634
26.4.7	Propagation of Transactions	636
26.5	Blockchain and Mining	636
26.5.1	Generating Blocks	636
26.5.2	Rewarding	637
26.5.3	Transaction and Merkle Tree	638
26.5.4	Brunching and Reaching Consensus	639
26.5.5	Double Spending and Majority of Hash Power	641
26.5.6	Case Study: Users with Majority of Hash Power	642
26.6	Summary	643