Contents

Preface **xv**

About the Author **xix**

Acknowledgments **xxi**

I Software Security 1

1 Set-UID Programs **3**

1.1 The Need for Privileged Programs 4
1.1.1 The Password Dilemma 4
1.1.2 Different Types of Privileged Programs 5
1.2 The `Set-UID` Mechanism 6
1.2.1 A Superman Story 6
1.2.2 How It Works 6
1.2.3 An Example of `Set-UID` Program 7
1.2.4 How to Ensure Its Security 8
1.2.5 The `Set-GID` Mechanism 8
1.3 What Can Go Wrong: What Happened to Superman 9
1.4 Attack Surfaces of `Set-UID` Programs 10
1.4.1 User Inputs: Explicit Inputs 10
1.4.2 System Inputs 11
1.4.3 Environment Variables: Hidden Inputs 11
1.4.4 Capability Leaking 12
1.5 Invoking Other Programs 14
1.5.1 Unsafe Approach: Using `system()` 14
1.5.2 Safe Approach: Using `execve()` 17
1.5.3 Invoking External Commands in Other Languages 18
1.5.4 Lessons Learned: Principle of Isolation 19
1.6 Principle of Least Privilege 19
1.7 Summary 20

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 31
2.3.1 Static and Dynamic Linking 32
2.3.2 Case Study: LD_PRELOAD and LD_LIBRARY_PATH 33
2.3.3 Case Study: OS X Dynamic Linker 36
2.4 Attack via External Program 36
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 39
2.5.1 Case Study - Locale in UNIX 39
2.6 Application Code 40
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 45
3.1 Background: Shell Functions 46
3.2 The Shellshock Vulnerability 48
3.2.1 Vulnerable Version of `bash` 48
3.2.2 The Shellshock Bug 48
3.2.3 Mistake in the Bash Source Code 49
3.2.4 Exploiting the Shellshock vulnerability 50
3.3 Shellshock Attack on `Set-UID` Programs 50
3.4 Shellshock Attack on CGI Programs 52
3.4.1 Experiment Environment Setup 52
3.4.2 How Web Server Invokes CGI Programs 52
3.4.3 How Attacker Sends Data to `Bash` 54
3.4.4 Launching the Shellshock Attack 54
3.4.5 Creating Reverse Shell 56
3.5 Remote Attack on PHP 58
3.6 Summary 59

4 Buffer Overflow Attack 61
4.1 Program Memory Layout 62
4.2 Stack and Function Invocation 63
4.2.1 Stack Memory Layout 63
4.2.2 Frame Pointer 64
4.3 Stack Buffer-Overflow Attack 65
4.3.1 Copy Data to Buffer 66
4.3.2 Buffer Overflow 67
4.3.3 Exploiting a Buffer Overflow Vulnerability 68
4.4 Setup for Our Experiment 69
4.4.1 Disable Address Randomization 70
4.4.2 Vulnerable Program 70
4.5 Conduct Buffer-Overflow Attack 71

4.5.1 Finding the Address of the Injected Code 71
4.5.2 Improving Chances of Guessing 72
4.5.3 Finding the Address Without Guessing 73
4.5.4 Constructing the Input File 74
4.6 Attacks with Unknown Address and Buffer Size 77
4.6.1 Knowing the Range of Buffer Size 77
4.6.2 Knowing the Range of the Buffer Address 77
4.6.3 A General Solution 78
4.7 Writing a Shellcode 80
4.7.1 Writing Malicious Code Using C 80
4.7.2 Writing a Shellcode: Main Idea 81
4.7.3 Explanation of a Shellcode Example 81
4.8 Countermeasures: Overview 84
4.9 Address Randomization 86
4.9.1 Address Randomization on Linux 87
4.9.2 Effectiveness of Address Randomization 88
4.10 StackGuard 89
4.10.1 The Observation and the Idea 90
4.10.2 Manually Adding Code to Function 90
4.10.3 StackGuard Implementation in `gcc` 92
4.11 Defeating the Countermeasure in `bash` and `dash` 94
4.12 Summary 96

5 Return-to-libc Attack and ROP **99**
5.1 Introduction: Non-Executable Stack 100
5.2 The Attack Experiment: Setup 101
5.3 Launch the Return-to-libc Attack: Part I 103
5.3.1 Task A: Find the Address of the `system()` Function 103
5.3.2 Task B: Find the Address of the String `"/bin/sh"` 104
5.4 Launch the Return-to-libc Attack: Part II 105
5.4.1 Function Prologue 106
5.4.2 Function Epilogue 107
5.4.3 Function Prologue and Epilogue Example 108
5.4.4 Perform Task C 109
5.4.5 Construct Malicious Input 110
5.4.6 Launch the Attack 111
5.5 Return-Oriented Programming 112
5.5.1 Experiment Setup 112
5.5.2 Tracking the values of the `esp` and `ebp` registers 114
5.5.3 Chaining Function Calls Without Arguments 115
5.5.4 Chaining Function Calls With Arguments: Skipping Prologue 117
5.5.5 Chaining Function Calls With Arguments: via `leave` and `ret` 120
5.5.6 Chaining Function Calls With Zero in the Argument 124
5.5.7 Use the Chaining Technique to Get Root Shell 125
5.5.8 Further Generalization: Return-Oriented Programming 127
5.6 Summary 128

6 Format String Vulnerability 129
6.1 Functions with Variable Number of Arguments . . . 130
6.1.1 How to Access Optional Arguments . . . 130
6.1.2 How `printf()` Accesses Optional Arguments . . . 132
6.2 Format String with Missing Optional Arguments . . . 133
6.3 Vulnerable Program and Experiment Setup . . . 135
6.4 Exploiting the Format String Vulnerability . . . 136
6.4.1 Attack 1: Crash Program . . . 136
6.4.2 Attack 2: Print out Data on the Stack . . . 137
6.4.3 Attack 3: Change the Program's Data in the Memory . . . 137
6.4.4 Attack 4: Change the Program's Data to a Specific Value . . . 139
6.4.5 Attack 4 (Continuation): A Much Faster Approach . . . 140
6.5 Code Injection Attack using Format String Vulnerability . . . 142
6.5.1 The Revised Vulnerable Program . . . 142
6.5.2 The Attack Strategy . . . 144
6.5.3 The Attack Program . . . 145
6.5.4 Reducing the Size of Format String . . . 147
6.6 Countermeasures . . . 149
6.6.1 Developer . . . 149
6.6.2 Compiler . . . 149
6.6.3 Address Randomization . . . 150
6.7 Relationship with the Buffer-Overflow Attack . . . 150
6.8 Summary . . . 151

7 Race Condition Vulnerability 153
7.1 The General Race Condition Problem . . . 154
7.2 Race Condition Vulnerability . . . 155
7.3 Experiment Setup . . . 157
7.4 Exploiting Race Condition Vulnerabilities . . . 158
7.4.1 Choose a Target File . . . 158
7.4.2 Launch Attack . . . 159
7.4.3 Monitor the Result . . . 160
7.4.4 Running the Exploit . . . 161
7.5 Countermeasures . . . 162
7.5.1 Atomic Operation . . . 162
7.5.2 Repeating Check and Use . . . 163
7.5.3 Sticky Symlink Protection . . . 164
7.5.4 Principle of Least Privilege . . . 165
7.6 Summary . . . 167

8 Dirty COW 169
8.1 Memory Mapping using `mmap()` . . . 170
8.2 `MAP_SHARED`, `MAP_PRIVATE` and Copy On Write . . . 171
8.3 Discard the Copied Memory . . . 173
8.4 Mapping Read-Only Files . . . 173
8.5 The Dirty COW Vulnerability . . . 175
8.6 Exploiting the Dirty COW Vulnerability . . . 176
8.6.1 Selecting `/etc/passwd` as Target File . . . 177

8.6.2 Set Up the Memory Mapping and Threads . . . 177
8.6.3 The `write` Thread . . . 178
8.6.4 The `madvise` Thread . . . 179
8.6.5 The Attack Result . . . 179
8.7 Summary . . . 180

9 Reverse Shell **181**
9.1 Introduction . . . 182
9.2 File Descriptor and Redirection . . . 182
9.2.1 File Descriptor . . . 182
9.2.2 Standard IO Devices . . . 184
9.2.3 Redirection . . . 185
9.2.4 How To Implement Redirection . . . 186
9.3 Redirecting Input/Output to a TCP Connection . . . 187
9.3.1 Redirecting Output to a TCP Connection . . . 187
9.3.2 Redirecting Input to a TCP Connection . . . 188
9.3.3 Redirecting to TCP Connection From Shell . . . 189
9.4 Reverse Shell . . . 190
9.4.1 Redirecting the Standard Output . . . 190
9.4.2 Redirecting the Standard Input . . . 190
9.4.3 Redirecting the Standard Error . . . 192
9.4.4 Code Injection . . . 192
9.5 Summary . . . 193

II Web Security **195**

10 Cross Site Request Forgery **197**
10.1 Cross-Site Requests and Its Problems . . . 198
10.2 Cross-Site Request Forgery Attack . . . 199
10.3 CSRF Attacks on HTTP GET Services . . . 200
10.3.1 HTTP GET and POST Services . . . 200
10.3.2 The Basic Idea of CSRF Attacks . . . 201
10.3.3 Attack on Elgg's Add-friend Service . . . 201
10.4 CSRF Attacks on HTTP POST Services . . . 203
10.4.1 Constructing a POST Request Using JavaScript . . . 203
10.4.2 Attack on Elgg's Edit-Profile Service . . . 204
10.5 Countermeasures . . . 206
10.5.1 Using the `referer` Header . . . 207
10.5.2 Same-Site Cookies . . . 207
10.5.3 Secret Token . . . 207
10.5.4 Case Study: `Elgg`'s Countermeasures . . . 208
10.6 Summary . . . 208

11 Cross-Site Scripting Attack **211**
11.1 The Cross-Site Scripting Attack . . . 212
11.1.1 Non-persistent (Reflected) XSS Attack . . . 213
11.1.2 Persistent XSS Attack . . . 214

11.1.3 What damage can XSS cause? 214
11.2 XSS Attacks in Action 215
11.2.1 Prelude: Injecting JavaScript Code 215
11.2.2 Use XSS Attacks to Befriend with Others 216
11.2.3 Use XSS Attacks to Change Other People's Profiles 219
11.3 Achieving Self-Propagation 221
11.3.1 Creating a Self-Propagating XSS Worm: the DOM Approach 222
11.3.2 Create a Self-Propagating Worm: the Link Approach 224
11.4 Preventing XSS attacks 225
11.4.1 Getting Rid of Code from User Inputs 225
11.4.2 Defeating XSS Attacks using Content Security Policy 226
11.4.3 Experimenting with Content Security Policy 228
11.5 Summary 230

12 SQL Injection Attack **233**
12.1 A Brief Tutorial of SQL 234
12.1.1 Log in to MySQL 234
12.1.2 Create a Database 234
12.1.3 CREATE a Table 234
12.1.4 INSERT a Row 235
12.1.5 The SELECT Statement 235
12.1.6 `WHERE` Clause 236
12.1.7 UPDATE SQL Statement 237
12.1.8 Comments in SQL Statements 237
12.2 Interacting with Database in Web Application 238
12.2.1 Getting Data from User 238
12.2.2 Getting Data From Database 239
12.3 Launching SQL Injection Attacks 241
12.3.1 Attack Using `cURL` 242
12.3.2 Modify Database 242
12.3.3 Multiple SQL Statements 243
12.4 The Fundamental Cause 244
12.5 Countermeasures 247
12.5.1 Filtering and Encoding Data 247
12.5.2 Prepared Statement 247
12.6 Summary 249

III Hardware Security **251**

13 Meltdown Attack **253**
13.1 Introduction and Analogy 254
13.1.1 Analogy: The Microsoft Brainteaser Question 254
13.1.2 Stealing A Secret 255
13.1.3 Side Channels 255
13.2 Side Channel Attacks via CPU Cache 256
13.2.1 Time Difference When Accessing Cache v.s Memory 256
13.2.2 Using CPU Cache as a Side Channel 258

13.3 The Room Holding Secret: The Kernel . . . 261
13.3.1 Secret Data in Kernel Space . . . 261
13.3.2 The Guard: Preventing Direct Access to Kernel Memory . . . 263
13.3.3 Avoid Getting Killed: Handling Error/Exceptions in C . . . 263
13.4 Passing the Guard: Out-of-Order Execution by CPU . . . 265
13.5 The Meltdown Attack . . . 268
13.5.1 A Naive Approach . . . 268
13.5.2 Improve the Attack by Getting the Secret Data Cached . . . 269
13.5.3 Improve the Attack Using Assembly Code . . . 269
13.5.4 Improve the Attack Using Statistic Approach . . . 270
13.6 Countermeasures . . . 272
13.7 Summary . . . 273

14 Spectre Attack **275**
14.1 Introduction . . . 276
14.2 Out-of-Order Execution and Branch Prediction . . . 276
14.2.1 An Experiment . . . 277
14.2.2 Experiment Results . . . 279
14.3 The Spectre Attack . . . 279
14.3.1 The Setup for the Experiment . . . 280
14.3.2 The Program Used in the Experiment . . . 281
14.4 Improve the Attack Using Statistic Approach . . . 283
14.5 Spectre Variant and Mitigation . . . 286
14.6 Summary . . . 286

IV Cryptography **287**

15 Secret-Key Encryption **289**
15.1 Introduction . . . 290
15.2 Substitution Cipher . . . 290
15.2.1 Monoalphabetic Substitution Cipher . . . 290
15.2.2 Breaking Monoalphabetic Substitution Cipher . . . 291
15.2.3 Polyalphabetic Substitution Cipher . . . 294
15.2.4 The Enigma Machine . . . 295
15.3 DES and AES Encryption Algorithms . . . 297
15.3.1 DES: Data Encryption Standard . . . 297
15.3.2 AES: Advanced Encryption Standard . . . 298
15.4 Encryption Modes . . . 298
15.4.1 Encryption Modes . . . 299
15.4.2 Electronic Codebook (ECB) Mode . . . 300
15.4.3 Cipher Block Chaining (CBC) Mode . . . 300
15.4.4 Cipher Feedback (CFB) Mode . . . 302
15.4.5 Output Feedback (OFB) Mode . . . 303
15.4.6 Counter (CTR) Mode . . . 304
15.4.7 Modes for Authenticated Encryption . . . 305
15.4.8 Padding . . . 306
15.5 Initialization Vector and Common Mistakes . . . 307

15.5.1 Common Mistake: Use the Same IV 307
15.5.2 Common Mistake: Use a Predictable IV 310
15.6 Programming using Cryptography APIs 313
15.7 Authenticated Encryption and the GCM Mode 315
15.7.1 The GCM Mode 316
15.7.2 Programming using the GCM Mode 317
15.8 Summary 318

16 One-Way Hash Function **319**
16.1 Introduction 320
16.2 Concept and Properties 320
16.2.1 Cryptographic Properties 320
16.2.2 Replay the Number Game 321
16.3 Algorithms and Programs 321
16.3.1 The MD (Message Digest) Series 322
16.3.2 The SHA (Secure Hash Algorithm) Series 322
16.3.3 How Hash Algorithm Works 323
16.3.4 One-Way Hash Commands 323
16.3.5 Computing One-Way Hash in Programs 324
16.3.6 Performance of One-Way Hash Functions 326
16.4 Applications of One-Way Hash Functions 326
16.4.1 Integrity Verification 327
16.4.2 Committing a Secret Without Telling It 327
16.4.3 Password Verification 328
16.4.4 Trusted Timestamping 330
16.5 Message Authentication Code (MAC) 331
16.5.1 Constructing MAC and Potential Attacks 332
16.5.2 Launching the Length Extension Attack 333
16.5.3 Case Study: Length Extension Attack on Flickr 336
16.5.4 The Keyed-Hash MAC (HMAC) Algorithm 336
16.6 Blockchain and Bitcoins 337
16.6.1 Hash Chain and Blockchain 337
16.6.2 Make Chaining Difficult 338
16.6.3 Adding Incentives and Bitcoin 340
16.7 Hash Collision Attacks 341
16.7.1 Security Impact of Collision Attacks 341
16.7.2 Generating Two Different Files with the Same MD5 Hash 342
16.7.3 Generating Two Programs with the Same MD5 Hash 344
16.7.4 Making the Two Programs Behave Differently 347
16.7.5 Hash-Colliding X.509 Certificates 349
16.8 Summary 350

17 Public Key Cryptography **351**
17.1 Introduction 352
17.2 Diffie-Hellman Key Exchange 352
17.2.1 Diffie-Hellman Key Exchange 353
17.2.2 Turn DH Key Exchange into a Public-Key Encryption Algorithm . . . 354
17.3 The RSA Algorithm 355

17.3.1 Math Background: Modulo Operation . . . 356
17.3.2 Math Background: Euler's Theorem . . . 356
17.3.3 Math Background: Extended Euclidean Algorithm . . . 357
17.3.4 The RSA Algorithm . . . 358
17.3.5 Exercise: Small Number . . . 359
17.3.6 Exercise: Large Number . . . 360
17.3.7 Performance . . . 362
17.3.8 Hybrid Encryption . . . 363
17.3.9 Other Public-Key Encryption Algorithms . . . 363
17.4 Using OpenSSL Tools to Conduct RSA Operations . . . 364
17.4.1 Generating RSA keys . . . 364
17.4.2 Extracting the public key . . . 365
17.4.3 Encryption and Decryption . . . 366
17.5 Paddings for RSA . . . 366
17.5.1 Attacks Against Textbook RSA . . . 367
17.5.2 Paddings: PKCS#1 v1.5 and OAEP . . . 367
17.6 Digital Signature . . . 368
17.6.1 Digital Signature using RSA . . . 369
17.6.2 DSA and Other Digital Signature Algorithms . . . 371
17.7 Programming using Public-Key Cryptography APIs . . . 371
17.7.1 Key Generation . . . 372
17.7.2 Encryption and Decryption . . . 372
17.7.3 Digital Signature . . . 374
17.8 Applications . . . 376
17.8.1 Authentication . . . 376
17.8.2 HTTPS and TLS/SSL . . . 378
17.8.3 Chip Technology Used in Credit Cards . . . 378
17.9 Blockchain and Bitcoins . . . 381
17.10Summary and Further Learning . . . 381

18 Public Key Infrastructure **383**
18.1 Attack on Public Key Cryptography . . . 384
18.1.1 Man-in-the-Middle (MITM) Attack . . . 384
18.1.2 Defeating MITM Attacks . . . 385
18.1.3 Public Key Infrastructure . . . 385
18.2 Public Key Certificates . . . 386
18.2.1 X.509 Digital Certificate . . . 386
18.2.2 Get Certificate from a Real Server . . . 387
18.3 Certificate Authority (CA) . . . 388
18.3.1 Being a CA . . . 389
18.3.2 Getting X.509 Certificate from CA . . . 390
18.3.3 Deploying Public Key Certificate in Web Server . . . 393
18.3.4 Apache Setup for HTTPS . . . 394
18.4 Root and Intermediate Certificate Authorities . . . 395
18.4.1 Root CAs and Self-Signed Certificate . . . 395
18.4.2 Intermediate CAs and Chain of Trust . . . 396
18.4.3 Creating Certificates for Intermediate CA . . . 397
18.4.4 Apache Setup . . . 398

18.4.5 Trusted CAs in the Real World . . . 398
18.5 How PKI Defeats the MITM Attack . . . 399
18.5.1 Attacker Forwards the Authentic Certificate . . . 399
18.5.2 Attacker Creates a Fake Certificate . . . 399
18.5.3 Attackers Send Their Own Certificates . . . 400
18.5.4 The Man-In-The-Middle Proxy . . . 401
18.6 Attacks on the Public-Key Infrastructure . . . 402
18.6.1 Attack on CA's Verification Process . . . 403
18.6.2 Attack on CA's Signing Process . . . 403
18.6.3 Attacks on the Algorithms . . . 404
18.6.4 Attacks on User Confirmation . . . 405
18.7 Types of Digital Certificates . . . 405
18.7.1 Domain Validated Certificates (DV) . . . 406
18.7.2 Organizational Validated Certificates (OV) . . . 406
18.7.3 Extended Validated Certificates (EV) . . . 407
18.8 Summary . . . 407