Contents

About the Authors vii
Acknowledgments xi
Introduction to the Second Edition xxiii

Part I Introduction to Exploitation: Linux on x86

Chapter 1 Before You Begin 3
Basic Concepts 3
Memory Management 4
Assembly 6
Recognizing C and C++ Code Constructs in Assembly 7
Conclusion 10

Chapter 2 Stack Overflows 11
Buffers 12
The Stack 13
Functions and the Stack 15
Overflowing Buffers on the Stack 18
Controlling EIP 22
An Interesting Diversion 23
Using an Exploit to Get Root Privileges 25
The Address Problem 27
The NOP Method 33
Defeating a Non-Executable Stack 35
Return to libc 35
Conclusion 39
## Contents

**Chapter 3** Shellcode
- Understanding System Calls 41
- Writing Shellcode for the exit() Syscall 44
- Injectable Shellcode 48
- Spawning a Shell 50
- Conclusion 59

**Chapter 4** Introduction to Format String Bugs
- Prerequisites 61
- What Is a Format String? 61
- What Is a Format String Bug? 63
- Format String Exploits 68
  - Crashing Services 69
  - Information Leakage 70
- Controlling Execution for Exploitation 75
- Why Did This Happen? 84
- Format String Technique Roundup 85
- Conclusion 88

**Chapter 5** Introduction to Heap Overflows
- What Is a Heap? 89
  - How a Heap Works 91
- Finding Heap Overflows 91
  - Basic Heap Overflows 93
  - Intermediate Heap Overflows 98
  - Advanced Heap Overflow Exploitation 105
- Conclusion 107

**Part II** Other Platforms—Windows, Solaris, OS/X, and Cisco

**Chapter 6** The Wild World of Windows
- How Does Windows Differ from Linux? 111
  - Win32 API and PE-COFF 112
- Heaps 114
  - Threading 115
- The Genius and Idiocy of the Distributed Common Object Model and DCE-RPC 116
  - Recon 118
  - Exploitation 120
  - Tokens and Impersonation 120
  - Exception Handling under Win32 122
- Debugging Windows 124
  - Bugs in Win32 124
  - Writing Windows Shellcode 125
    - A Hacker’s Guide to the Win32 API 126
    - A Windows Family Tree from the Hacker’s Perspective 126
  - Conclusion 127
### Chapter 7  **Windows Shellcode**

- Syntax and Filters  
  - Setting Up  
    - Parsing the PEB  
    - Heapoverflow.c Analysis  
    - Searching with Windows Exception Handling  
    - Popping a Shell  
    - Why You Should Never Pop a Shell on Windows  
  - Conclusion  

- 129

### Chapter 8  **Windows Overflows**

- Stack-Based Buffer Overflows  
- Frame-Based Exception Handlers  
- Abusing Frame-Based Exception Handling on Windows 2003 Server  
  - A Final Note about Frame-Based Handler Overwrites  
- Stack Protection and Windows 2003 Server  
- Heap-Based Buffer Overflows  
  - The Process Heap  
    - Dynamic Heaps  
    - Working with the Heap  
    - How the Heap Works  
  - Exploiting Heap-Based Overflows  
    - Overwrite Pointer to RtlEnterCriticalSection in the PEB  
    - Overwrite Pointer to Unhandled Exception Filter  
    - Repairing the Heap  
    - Other Aspects of Heap-Based Overflows  
    - Wrapping Up the Heap  
  - Other Overflows  
    - .data Section Overflows  
    - TEB/PEB Overflows  
- Exploiting Buffer Overflows and Non-Executable Stacks  
- Conclusion  

- 155

### Chapter 9  **Overcoming Filters**

- Writing Exploits for Use with an Alphanumeric Filter  
- Writing Exploits for Use with a Unicode Filter  
  - What Is Unicode?  
  - Converting from ASCII to Unicode  
- Exploiting Unicode-Based Vulnerabilities  
  - The Available Instruction Set in Unicode Exploits  
- The Venetian Method  
  - An ASCII Venetian Implementation  
- Decoder and Decoding  
  - The Decoder Code  
  - Getting a Fix on the Buffer Address  
- Conclusion  

- 205
# Contents

**Chapter 10  Introduction to Solaris Exploitation**  
Introduction to the SPARC Architecture 224  
Registers and Register Windows 224  
The Delay Slot 227  
Synthetic Instructions 228  
Solaris/SPARC Shellcode Basics 228  
Self-Location Determination and SPARC Shellcode 228  
Simple SPARC exec Shellcode 229  
Useful System Calls on Solaris 230  
NOP and Padding Instructions 231  
Solaris/SPARC Stack Frame Introduction 231  
Stack-Based Overflow Methodologies 232  
Arbitrary Size Overflow 232  
Register Windows and Stack Overflow Complications 233  
Other Complicating Factors 233  
Possible Solutions 234  
Off-By-One Stack Overflow Vulnerabilities 234  
Shellcode Locations 235  
Stack Overflow Exploitation In Action 236  
The Vulnerable Program 236  
The Exploit 238  
Heap-Based Overflows on Solaris/SPARC 241  
Solaris System V Heap Introduction 242  
Heap Tree Structure 242  
Basic Exploit Methodology (t_delete) 263  
Standard Heap Overflow Limitations 266  
Targets for Overwrite 267  
Other Heap-Related Vulnerabilities 270  
Off-by-One Overflows 270  
Double Free Vulnerabilities 270  
Arbitrary Free Vulnerabilities 271  
Heap Overflow Example 271  
The Vulnerable Program 272  
Other Solaris Exploitation Techniques 276  
Static Data Overflows 276  
Bypassing the Non-Executable Stack Protection 276  
Conclusion 277

**Chapter 11  Advanced Solaris Exploitation**  
Single Stepping the Dynamic Linker 281  
Various Style Tricks for Solaris SPARC Heap Overflows 296  
Advanced Solaris/SPARC Shellcode 299  
Conclusion 311
Chapter 12  OS X Shellcode 313
OS X Is Just BSD, Right? 314
Is OS X Open Source? 314
OS X for the Unix-aware 315
Password Cracking 316
OS X PowerPC Shellcode 316
OS X Intel Shellcode 324
Example Shellcode 326
ret2libc 327
ret2str(l)cpy 329
OS X Cross-Platform Shellcode 332
OS X Heap Exploitation 333
Bug Hunting on OS X 335
Some Interesting Bugs 335
Essential Reading for OS X Exploits 337
Conclusion 338

Chapter 13  Cisco IOS Exploitation 339
An Overview of Cisco IOS 339
   Hardware Platforms 340
   Software Packages 340
   IOS System Architecture 343
Vulnerabilities in Cisco IOS 346
   Protocol Parsing Code 347
   Services on the Router 347
   Security Features 348
   The Command-Line Interface 348
Reverse Engineering IOS 349
   Taking the Images Apart 349
   Differing IOS Images 350
   Runtime Analysis 351
Exploiting Cisco IOS 357
   Stack Overflows 357
   Heap Overflows 359
   Shellcodes 364
Conclusion 373

Chapter 14  Protection Mechanisms 375
Protections 375
   Non-Executable Stack 376
   W^X (EitherWritable or Executable) Memory 381
   Stack Data Protection 388
   AAAS: ASCII Armored Address Space 394
   ASLR: Address Space Layout Randomization 396
   Heap Protections 399
   Windows SEH Protections 407
   Other Protections 411
## Part III Vulnerability Discovery

**Chapter 15 Establishing a Working Environment**

### What You Need for Reference

- gcc
- gdb
- NASM
- WinDbg
- OllyDbg
- Visual C++
- Python

### What You Need for Code

- gcc
- gdb

### What You Need for Investigation

- Useful Custom Scripts/Tools
- All Platforms
- Unix
- Windows

### What You Need to Know

- Paper Archives

### Optimizing Shellcode Development

- Plan the Exploit
- Write the Shellcode in Inline Assembler
- Maintain a Shellcode Library
- Make It Continue Nicely
- Make the Exploit Stable
- Make It Steal the Connection

### Conclusion

**Chapter 16 Fault Injection**

### Design Overview

- Input Generation
- Fault Injection
- Modification Engines
- Fault Delivery
- Nagel Algorithm
- Timing
- Heuristics
- Stateless versus State-Based Protocols

### Fault Monitoring

- Using a Debugger
- FaultMon
## Contents

Putting It Together 458
Conclusion 459

### Chapter 17  The Art of Fuzzing 461

- General Theory of Fuzzing 461
- Static Analysis versus Fuzzing 466
- Fuzzing Is Scalable 466
- Weaknesses in Fuzzers 468
- Modeling Arbitrary Network Protocols 469
- Other Fuzzer Possibilities 469
  - Bit Flipping 469
  - Modifying Open Source Programs 470
  - Fuzzing with Dynamic Analysis 470
- SPIKE 471
  - What Is a Spike? 471
  - Why Use the SPIKE Data Structure to Model Network Protocols? 472
- Other Fuzzers 480
Conclusion 480

### Chapter 18  Source Code Auditing: Finding Vulnerabilities in C-Based Languages 481

- Tools 482
  - Cscope 482
  - Ctags 483
  - Editors 483
  - Cbrowser 484
- Automated Source Code Analysis Tools 484
- Methodology 485
  - Top-Down (Specific) Approach 485
  - Bottom-Up Approach 485
  - Selective Approach 485
- Vulnerability Classes 486
  - Generic Logic Errors 486
  - (Almost) Extinct Bug Classes 487
  - Format Strings 487
  - Generic Incorrect Bounds-Checking 488
  - Loop Constructs 489
  - Off-by-One Vulnerabilities 490
  - Non-Null Termination Issues 492
  - Skipping Null-Termination Issues 493
  - Signed Comparison Vulnerabilities 494
  - Integer-Related Vulnerabilities 495
  - Different-Sized Integer Conversions 497
  - Double Free Vulnerabilities 498
  - Out-of-Scope Memory Usage Vulnerabilities 499
  - Uninitialized Variable Usage 499
  - Use After Free Vulnerabilities 500
  - Multithreaded Issues and Re-Entrant Safe Code 500
# Contents

Beyond Recognition: A Real Vulnerability versus a Bug  501
Conclusion  501

**Chapter 19** Instrumented Investigation: A Manual Approach  503
Philosophy  503
Oracle extproc Overflow  504
Common Architectural Failures  508
Problems Happen at Boundaries  508
Problems Happen When Data Is Translated  509
Problems Cluster in Areas of Asymmetry  511
Problems Occur When Authentication and Authorization Are Confused  512
Problems Occur in the Dumbest Places  512
Bypassing Input Validation and Attack Detection  513
Stripping Bad Data  513
Using Alternate Encodings  514
Using File-Handling Features  515
Evading Attack Signatures  517
Defeating Length Limitations  517
Windows 2000 SNMP DOS  520
Finding DOS Attacks  521
SQL-UDP  522
Conclusion  523

**Chapter 20** Tracing for Vulnerabilities  525
Overview  526
A Vulnerable Program  527
Component Design  529
Building VulnTrace  538
Using VulnTrace  543
Advanced Techniques  546
Conclusion  548

**Chapter 21** Binary Auditing: Hacking Closed Source Software  549
Binary versus Source-Code Auditing: The Obvious Differences  550
IDA Pro—The Tool of the Trade  550
Features: A Quick Crash Course  551
Debugging Symbols  552
Binary Auditing Introduction  552
Stack Frames  552
Calling Conventions  554
Compiler-Generated Code  556
memcpy-Like Code Constructs  560
strlen-Like Code Constructs  560
C++ Code Constructs  561
The this Pointer  561
Reconstructing Class Definitions  562
vtables  562
Quick but Useful Tidbits  563
### Contents

<table>
<thead>
<tr>
<th>Manual Binary Analysis</th>
<th>563</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quick Examination of Library Calls</td>
<td>564</td>
</tr>
<tr>
<td>Suspicious Loops and Write Instructions</td>
<td>564</td>
</tr>
<tr>
<td>Higher-Level Understanding and Logic Bugs</td>
<td>565</td>
</tr>
<tr>
<td>Graphical Analysis of Binaries</td>
<td>566</td>
</tr>
<tr>
<td>Manual Decompilation</td>
<td>566</td>
</tr>
<tr>
<td>Binary Vulnerability Examples</td>
<td>566</td>
</tr>
<tr>
<td>Microsoft SQL Server Bugs</td>
<td>566</td>
</tr>
<tr>
<td>LSD’s RPC-DCOM Vulnerability</td>
<td>567</td>
</tr>
<tr>
<td>IIS WebDAV Vulnerability</td>
<td>568</td>
</tr>
<tr>
<td>Conclusion</td>
<td>570</td>
</tr>
</tbody>
</table>

### Part IV Advanced Materials

#### Chapter 22 Alternative Payload Strategies

<table>
<thead>
<tr>
<th>Modifying the Program</th>
<th>573</th>
</tr>
</thead>
<tbody>
<tr>
<td>The SQL Server 3-Byte Patch</td>
<td>574</td>
</tr>
<tr>
<td>The MySQL 1-Bit Patch</td>
<td>575</td>
</tr>
<tr>
<td>OpenSSH RSA Authentication Patch</td>
<td>578</td>
</tr>
<tr>
<td>Other Runtime Patching Ideas</td>
<td>580</td>
</tr>
<tr>
<td>GPG 1.2.2 Randomness Patch</td>
<td>581</td>
</tr>
<tr>
<td>Upload and Run (or Proglet Server)</td>
<td>583</td>
</tr>
<tr>
<td>Syscall Proxies</td>
<td>584</td>
</tr>
<tr>
<td>Problems with Syscall Proxies</td>
<td>587</td>
</tr>
<tr>
<td>Conclusion</td>
<td>596</td>
</tr>
</tbody>
</table>

#### Chapter 23 Writing Exploits that Work in the Wild

<table>
<thead>
<tr>
<th>Factors in Unreliability</th>
<th>597</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magic Numbers</td>
<td>597</td>
</tr>
<tr>
<td>Versioning</td>
<td>598</td>
</tr>
<tr>
<td>Shellcode Problems</td>
<td>599</td>
</tr>
<tr>
<td>Countermeasures</td>
<td>601</td>
</tr>
<tr>
<td>Preparation</td>
<td>602</td>
</tr>
<tr>
<td>Brute Forcing</td>
<td>602</td>
</tr>
<tr>
<td>Local Exploits</td>
<td>603</td>
</tr>
<tr>
<td>OS/Application Fingerprinting</td>
<td>603</td>
</tr>
<tr>
<td>Information Leaks</td>
<td>605</td>
</tr>
<tr>
<td>Conclusion</td>
<td>606</td>
</tr>
</tbody>
</table>

#### Chapter 24 Attacking Database Software

<table>
<thead>
<tr>
<th>Network Layer Attacks</th>
<th>607</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Layer Attacks</td>
<td>610</td>
</tr>
<tr>
<td>Running Operating System Commands</td>
<td>618</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>619</td>
</tr>
<tr>
<td>Oracle</td>
<td>620</td>
</tr>
<tr>
<td>IBM DB2</td>
<td>621</td>
</tr>
<tr>
<td>Exploiting Overruns at the SQL Level</td>
<td>622</td>
</tr>
<tr>
<td>SQL Functions</td>
<td>623</td>
</tr>
<tr>
<td>Conclusion</td>
<td>625</td>
</tr>
</tbody>
</table>
## Contents

### Chapter 25 Unix Kernel Overflows
- Kernel Vulnerability Types
- 0day Kernel Vulnerabilities
  - OpenBSD exec_ibcs2_coff_prep_zmagic() Stack Overflow
  - The Vulnerability
- Solaris vfs_getvfssw() Loadable Kernel Module
  - Traversal Vulnerability
  - The sysfs() System Call
  - The mount() System Call
- Conclusion

### Chapter 26 Exploiting Unix Kernel Vulnerabilities
- The exec_ibcs2_coff_prep_zmagic() Vulnerability
  - Calculating Offsets and Breakpoints
  - Overwriting the Return Address and Redirecting Execution
  - Locating the Process Descriptor (or the Proc Structure)
  - Kernel Mode Payload Creation
  - Returning Back from Kernel Payload
  - Getting root (uid=0)
- Solaris vfs_getvfssw() Loadable Kernel Module Path Traversal Exploit
  - Crafting the Exploit
  - The Kernel Module to Load
  - Getting root (uid=0)
- Conclusion

### Chapter 27 Hacking the Windows Kernel
- Windows Kernel Mode Flaws—An Increasingly Hunted Species
- Introduction to the Windows Kernel
- Common Kernel-Mode Programming Flaws
  - Stack Overflows
  - Heap Overflows
  - Insufficient Validation of User-Mode Addresses
  - Repurposing Attacks
  - Shared Object Attacks
- Windows System Calls
  - Understanding System Calls
  - Attacking System Calls
- Communicating with Device Drivers
  - I/O Control Code Components
  - Finding Flaws in IOCTL Handlers
- Kernel-Mode Payloads
  - Elevating a User-Mode Process
  - Running an Arbitrary User-Mode Payload
  - Subverting Kernel Security
  - Installing a Rootkit
- Essential Reading for Kernel Shellcoders
- Conclusion

### Index