The Shellcoder's Handbook: Discovering and Exploiting Security Holes

Jack Koziol, Dave Aitel, David Litchfield, Chris Anley, Sinan "noir" Eren, Neel Mehta, Riley Hassell

WILEY
Wiley Publishing, Inc.
# Contents

## About the Authors

About the Authors

## Credits

Credits

## Acknowledgments

Acknowledgments

## Part 1 Introduction to Exploitation: Linux on x86

### Chapter 1 Before You Begin

- Basic Concepts  
  - Memory Management  
  - Assembly  
  - Registers  
- Recognizing C++ Code Constructs in Assembly  
- Conclusion

### Chapter 2 Stack Overflows

- Buffers  
- The Stack  
  - Functions and the Stack  
- Overflowing Buffers on the Stack  
  - Controlling EIP  
- Using an Exploit to Get Root Privileges  
  - The Address Problem  
  - The NOP Method  
- Defeating a Non-Executable Stack  
  - Return to libc  
- Conclusion
Chapter 3  Shellcode  
Understanding System Calls  36 
Writing Shellcode for the exit() Syscall  38 
Injectable Shellcode  42 
Spawning a Shell  44 
Conclusion  53 

Chapter 4  Introduction to Format String Bugs  
Prerequisites  55 
What Is a Format String?  56 
What Is a Format String Bug?  57 
Format String Exploits  62 
  Crashing Services  63 
  Information Leakage  64 
Controlling Execution for Exploitation  69 
Why Did This Happen?  79 
Format String Technique Roundup  79 
Conclusion  82 

Chapter 5  Introduction to Heap Overflows  
What Is a Heap?  84 
  How a Heap Works  85 
Finding Heap Overflows  86 
  Basic Heap Overflows  87 
  Intermediate Heap Overflows  93 
  Advanced Heap Overflow Exploitation  99 
    What to Overwrite  100 
Conclusion  101 

Part 2  Exploiting More Platforms: Windows, Solaris, and Tru64  

Chapter 6  The Wild World of Windows  
How Does Windows Differ from Linux?  105 
  Win32 API and PE-COFF  106 
Heaps  108 
  Threading  109 
The Genius and Idiocy of the Distributed Common Object Model and DCE-RPC  110 
Recon  112 
Exploitation  114 
Tokens and Impersonation  114 
Exception Handling under Win32  116 
Debugging Windows  118 
  Bugs in Win32  118 
  Writing Windows Shellcode  119 
A Hacker's Guide to the Win32 API  119 
  A Windows Family Tree from the Hacker's Perspective  120 
Conclusion  121
# 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**

# Chapter 10 Introduction to Solaris Exploitation

- **Introduction to the SPARC Architecture**
- **Registers and Register Windows**
- **The Delay Slot**
- **Synthetic Instructions**
- **Solaris/SPARC Shellcode Basics**
  - Self-Location Determination and SPARC Shellcode
  - Simple SPARC exec Shellcode
  - Useful System Calls on Solaris
  - NOP and Padding Instructions
- **Solaris/SPARC Stack Frame Introduction**
- **Stack-Based Overflow Methodologies**
  - Arbitrary Size Overflow
  - Register Windows and Stack Overflow Complications
  - Other Complicating Factors
  - Possible Solutions
  - Off-By-One Stack Overflow Vulnerabilities
  - Shellcode Locations
- **Stack Overflow Exploitation In Action**
  - The Vulnerable Program
  - The Exploit
- **Heap-Based Overflows on Solaris/SPARC**
  - Solaris System V Heap Introduction
  - Heap Tree Structure
  - Basic Exploit Methodology (t_delete)
  - Standard Heap Overflow Limitations
  - Targets for Overwrite
    - The Bottom Chunk
    - Small Chunk Corruption
  - Other Heap-Related Vulnerabilities
    - Off-by-One Overflows
    - Double Free Vulnerabilities
    - Arbitrary Free Vulnerabilities
Heap Overflow Example 262
  The Vulnerable Program 262
Other Solaris Exploitation Techniques 266
  Static Data Overflows 267
  Bypassing the Non-Executable Stack Protection 267
Conclusion 268

Chapter 11 Advanced Solaris Exploitation 269
  Single Stepping the Dynamic Linker 271
  Various Style Tricks for Solaris SPARC Heap Overflows 286
  Advanced Solaris/SPARC Shellcode 288
Conclusion 300

Chapter 12 HP Tru64 Unix Exploitation 301
  The Alpha Architecture 302
    Alpha Registers 302
    Instruction Set 303
    Calling Conventions 305
  Retrieving the Program Counter (GetPC) 306
  System Call Invocation 308
  XOR Decoder for Shellcode 308
    main setuid + execve Shellcode 310
      Code the setuid(0) + execve("/bin/sh",...) systemcalls 310
    Compile the Assembly Code and Extract the Main Function 311
    Encode the Extracted opcodes with the XOR Key 312
    Plug the Encoded Code into the XOR Decoder 313
    Compile and Extract the Final Shellcode 314
  Connect-Back Shellcode 316
  Find-Socket Shellcode 317
  Bind-Socket Shellcode 319
  Stack Overflow Exploitation 320
    Defeating the Non-Executable Stack 321
  Exploiting rpc.ttdbserver 322
Conclusion 330

Part 3 Vulnerability Discovery 331

Chapter 13 Establishing a Working Environment 333
  What You Need for Reference 334
  What You Need for Code 334
    gcc 334
    gdb 335
    NASM 335
    WinDbg 335
    OllyDbg 335
    SoftICE 335
    Visual C++ 336
    Python 336
What You Need for Investigation 336
Useful Custom Scripts/Tools 336
   An Offset Finder 336
   Generic Fuzzers 337
   The Debug Trick 337
All Platforms 338
Unix
   ltrace 339
   strace 339
   fstat (BSD) 339
   tcpdump 339
   Ethereal 339
Windows
   IDA Pro Disassembler 340
What You Need to Know 340
Paper Archives 343
Optimizing Shellcode Development 343
   Plan the Exploit 343
   Write the Shellcode in Inline Assembler 344
   Maintain a Shellcode Library 345
   Make It Continue Nicely 346
   Make the Exploit Stable 347
   Make It Steal the Connection 347
Conclusion 348

Chapter 14 Fault Injection 349
Design Overview 350
   Input Generation 351
      Manual Generation 352
      Automated Generation 352
      Live Capture 353
      “Fuzz” Generation 353
   Fault Injection 354
   Modification Engines 354
      Delimiting Logic 355
      Getting around Input Sanitization 357
   Fault Delivery 358
   Nagel Algorithm 359
   Timing 359
   Heuristics 359
   Stateless versus State-Based Protocols 360
Fault Monitoring 360
   Using a Debugger 360
   FaultMon 361
Putting It Together 361
Conclusion 362
Chapter 15  The Art of Fuzzing

General Theory of Fuzzing 363
Static Analysis versus Fuzzing 368
Fuzzing Is Scalable 368
Weaknesses in Fuzzers 370
Modeling Arbitrary Network Protocols 371
Other Fuzzer Possibilities 371
Bit Flipping 371
Modifying Open Source Programs 372
Fuzzing with Dynamic Analysis 372
SPIKE 372
What Is a Spike? 373
Why Use the SPIKE Data Structure to Model Network Protocols? 374
Various Programs Included with SPIKE 374
SPIKE Example: dtlogin 374
Other Fuzzers 381
Conclusion 382

Chapter 16  Source Code Auditing: Finding Vulnerabilities in C-Based Languages

Tools 384
Cscope 384
Ctags 385
Editors 385
Cbrowser 385
Automated Source Code Analysis Tools 386
Methodology 386
Top-Down (Specific) Approach 387
Bottom-Up Approach 387
Selective Approach 387
Vulnerability Classes 388
Generic Logic Errors 388
(Almost) Extinct Bug Classes 388
Format Strings 389
Generic Incorrect Bounds-Checking 390
Loop Constructs 392
Off-by-One Vulnerabilities 392
Non-Null Termination Issues 393
Skipping Null-Termination Issues 394
Signed Comparison Vulnerabilities 395
Integer-Related Vulnerabilities 396
Different-Sized Integer Conversions 398
Double Free Vulnerabilities 400
Out-of-Scope Memory Usage Vulnerabilities 400
Uninitialized Variable Usage 400
| Contents |
|-----------------|--------|
| Use After Free Vulnerabilities | 401 |
| Multithreaded Issues and Re-Entrant Safe Code | 402 |
| Beyond Recognition: A Real Vulnerability versus a Bug | 402 |
| Conclusion | 403 |

**Chapter 17 Instrumented Investigation: A Manual Approach**

- Philosophy | 405 |
- Oracle extproc Overflow | 406 |
- Common Architectural Failures | 410 |
  - Problems Happen at Boundaries | 410 |
    - A Process Calling into an External Process on the Same Host | 410 |
    - A Process Calling into an External, Dynamically Loaded Library | 410 |
    - A Process Calling into a Function on a Remote Host | 411 |
  - Problems Happen When Data Is Translated | 411 |
  - Problems Cluster in Areas of Asymmetry | 413 |
  - Problems Occur When Authentication and Authorization Are Confused | 414 |
  - Problems Occur in the Dumbest Places | 414 |
- Bypassing Input Validation and Attack Detection | 415 |
  - Stripping Bad Data | 415 |
  - Using Alternate Encodings | 415 |
  - Using File-Handling Features | 416 |
    - Required String Is Present in Path | 416 |
    - Prohibited String Not Present in Path | 417 |
    - Incorrect Behavior Based on File Extension | 417 |
- Evading Attack Signatures | 419 |
- Defeating Length Limitations | 419 |
  - Sea Monkey Data | 419 |
  - Harmful Truncation—Severing Escape Characters | 420 |
  - Multiple Attempts | 421 |
  - Context-Free Length Limits | 421 |
- Windows 2000 SNMP DOS | 422 |
- Finding DOS Attacks | 423 |
- SQL-UDP | 424 |
- Conclusion | 424 |

**Chapter 18 Tracing for Vulnerabilities**

- Overview | 428 |
  - A Vulnerable Program | 429 |
  - Component Design | 431 |
    - Process Injection | 432 |
    - Machine-Code Analysis | 432 |
    - Function Hooking | 436 |
    - Data Collection | 439 |
### Part 4  Advanced Materials  

#### Chapter 20  Alternative Payload Strategies  
- [Modifying the Program](#)  
- [The SQL Server 3-Byte Patch](#)  
- [The MySQL 1-Bit Patch](#)  
- [OpenSSH RSA Authentication Patch](#)  
- [Other Runtime Patching Ideas](#)  
  - GPG 1.2.2 Randomness Patch  
- [Upload and Run (or Proglet Server)](#)  
- [Syscall Proxies](#)  
- [Problems with Syscall Proxies](#)  
- [Conclusion](#)  

#### Chapter 21  Writing Exploits that Work in the Wild  
- [Factors in Unreliability](#)  
  - Magic Numbers  
  - Versioning  
- [Shellcode Problems](#)  
  - Network Related  
  - Privilege Related  
  - Configuration Related  
  - Host IDS Related  
  - Thread Related  
- [Countermeasures](#)  
  - Preparation  
  - Brute Forcing  
  - Local Exploits  
  - OS/Application Fingerprinting  
  - Information Leaks  
- [Conclusion](#)  

#### Chapter 22  Attacking Database Software  
- [Network Layer Attacks](#)  
- [Application Layer Attacks](#)  
- [Running Operating System Commands](#)  
  - Microsoft SQL Server  
  - Oracle  
  - IBM DB2  
- [Exploiting Overruns at the SQL Level](#)  
- [SQL Functions](#)  
  - Using the CHR/CHAR Function  
- [Conclusion](#)  

#### Chapter 23  Kernel Overflows  
- [Kernel Vulnerability Types](#)  
- [0day Kernel Vulnerabilities](#)  
  - OpenBSD exec_ibcs2_coff_prep_zmagic() Stack Overflow  
  - The Vulnerability  

---

**Contents**

**Part 4  Advanced Materials**

**Chapter 20  Alternative Payload Strategies**

- Modifying the Program
- The SQL Server 3-Byte Patch
- The MySQL 1-Bit Patch
- OpenSSH RSA Authentication Patch
- Other Runtime Patching Ideas
  - GPG 1.2.2 Randomness Patch
- Upload and Run (or Proglet Server)
- Syscall Proxies
- Problems with Syscall Proxies
- Conclusion

**Chapter 21  Writing Exploits that Work in the Wild**

- Factors in Unreliability
  - Magic Numbers
  - Versioning
- Shellcode Problems
  - Network Related
  - Privilege Related
  - Configuration Related
  - Host IDS Related
  - Thread Related
- Countermeasures
  - Preparation
  - Brute Forcing
  - Local Exploits
  - OS/Application Fingerprinting
  - Information Leaks
- Conclusion

**Chapter 22  Attacking Database Software**

- Network Layer Attacks
- Application Layer Attacks
- Running Operating System Commands
  - Microsoft SQL Server
  - Oracle
  - IBM DB2
- Exploiting Overruns at the SQL Level
- SQL Functions
  - Using the CHR/CHAR Function
- Conclusion

**Chapter 23  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 544
   The sysfs() System Call 546
   The mount() System Call 547
   Conclusion 548

Chapter 24 Exploiting Kernel Vulnerabilities 549
   The exec_ibcs2_coff_prep_zmagic() Vulnerability 549
      Calculating Offsets and Breakpoints 554
      Overwriting the Return Address and Redirecting Execution 557
      Locating the Process Descriptor (or the Proc Structure) 558
         Stack Lookup 558
         sysctl() Syscall 558
      Kernel Mode Payload Creation 560
         p_cred and u_cred 560
         Breaking chroot 561
      Returning Back from Kernel Payload 561
         Return to User Mode: iret Technique 562
         Return to Kernel Code: sidt Technique 564
            and _kernel_text Search 564
      Getting root (uid=0) 567
Solaris vfs_getvfssw() Loadable Kernel Module Path Traversal Exploit 574
   Crafting the Exploit 574
   The Kernel Module to Load 576
   Getting root (uid=0) 579
   Conclusion 580

Index 581