

Exploiting Software

Greg Hoglund
Gary McGraw

• Addison-Wesley

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid • Capetown
Sydney • Tokyo • Singapore • Mexico City



Contents

Attack Patterns xxiii

Foreword xxv

Preface xxix

What This Book Is About xxx

How to Use This Book xxxi

But Isn't This Too Dangerous? xxxii

Acknowledgments xxxiii

1 Software—The Root of the Problem 1

A Brief History of Software 2

Software and the Information Warrior 5

Digital Tradecraft 6

Bad Software Is Ubiquitous 10

NASA Mars Lander 11

Denver Airport Baggage 11

MV-22 Osprey 11

The US Vicennes 12

Microsoft and the Love Bug 12

The Trinity of Trouble 14

Complexity 14

More Lines, More Bugs 16

Extensibility 18

Connectivity 21

The Upshot 23

The Future of Software 23

Short-Term Future: 2003-2004 24

Medium-Term Future: 2005-2007 28

<i>Long-Term Future: 2008-2010</i>	30
<i>Ten Threads Emerge</i>	32
What Is Software Security?	33
Conclusion	34
2 Attack Patterns	37
A Taxonomy	38
<i>Bugs</i>	39
<i>Flaws</i>	39
<i>Vulnerabilities</i>	39
<i>Design Vulnerabilities</i>	41
An Open-Systems View	42
<i>Risk</i>	44
<i>Damage Potential</i>	45
<i>Exposure and Potency</i>	46
<i>Actual Risk</i>	47
Tour of an Exploit	48
<i>The Attacker's Viewpoint</i>	48
<i>Why Trusting Users Is Bad</i>	49
<i>Like a Lock Pick</i>	50
<i>A Simple Example</i>	51
Attack Patterns: Blueprints for Disaster	55
<i>Exploit, Attack, and Attacker</i>	55
<i>Attack Pattern</i>	56
<i>Injection Vector</i>	56
<i>Activation Zone</i>	57
<i>Output Event</i>	57
<i>Feedback Event</i>	57
An Example Exploit: Microsoft's Broken C++ Compiler	57
<i>Technical Details of the Attack</i>	59
<i>An Overview of Microsoft's Port of StackGuard</i>	61
<i>Bypassing the Microsoft Feature</i>	61
<i>Solutions</i>	63
<i>An Exploit in Retrospect</i>	64
Applying Attack Patterns	65
<i>Network Scanning</i>	65
<i>OS Stack Identification</i>	66
<i>Port Scans</i>	67
<i>Traceroute and Zone Transfers</i>	67

<i>Target Components</i>	67
<i>Choosing Attack Patterns</i>	68
<i>Leveraging Faults in the Environment</i>	68
<i>Using Indirection</i>	68
<i>Planting Backdoors</i>	69
Attack Pattern Boxes	70
<i>Attack Pattern: Target Programs That Write to Privileged OS Resources</i>	70
Conclusion	70
3 Reverse Engineering and Program Understanding	71
Into the House of Logic	72
<i>Reverse Engineering</i>	72
<i>Why Reverse Engineer?</i>	73
Should Reverse Engineering Be Illegal?	75
Reverse Engineering Tools and Concepts	77
<i>The Debugger</i>	77
<i>Fault Injection Tools</i>	78
<i>The Disassembler</i>	78
<i>The Reverse Compiler or Decompiler</i>	79
Approaches to Reverse Engineering	79
<i>White Box Analysis</i>	79
<i>Black Box Analysis</i>	80
<i>Gray Box Analysis</i>	81
<i>Using Gray Box Techniques to Find Vulnerabilities in Microsoft SQL Server 7</i>	82
Methods of the Reverser	84
<i>Tracing Input</i>	84
<i>Exploiting Version Differences</i>	86
<i>Making Use of Code Coverage</i>	87
<i>Accessing the Kernel</i>	88
<i>Leaking Data in Shared Buffers</i>	88
<i>Auditing for Access Requirement Screwups</i>	90
<i>Using Your API Resources</i>	90
Writing Interactive Disassembler (IDA) Plugins	92
Decompiling and Disassembling Software	104
Decompilation in Practice: Reversing <code>hel pctr. exe</code>	105
<i>Bug Report</i>	106
<i>The Debug Log</i>	107

Automatic, Bulk Auditing for Vulnerabilities	111
<i>Batch Analysis with IDA-Pro</i>	114
Writing Your Own Cracking Tools	121
<i>x86 Tools</i>	121
<i>The Basic x86 Debugger</i>	121
<i>On Breakpoints</i>	123
<i>Reading and Writing Memory</i>	126
<i>Debugging Multithreaded Programs</i>	127
<i>Enumerate Threads or Processes</i>	129
<i>Single Stepping</i>	130
<i>Patching</i>	132
<i>Fault Injection</i>	132
<i>Process Snapshots</i>	133
<i>Disassembling Machine Code</i>	138
Building a Basic Code Coverage Tool	139
<i>Checking for Boron Tags</i>	144
Conclusion	145

4 Exploiting Server Software 147

The Trusted Input Problem	149
<i>Attack Pattern: Make the Client Invisible</i>	150
The Privilege Escalation Problem	151
<i>Process-Permissions Equal Trust</i>	151
<i>If We Don't Run as Administrator, Everything Breaks!</i>	152
<i>Attack Pattern: Target Programs That Write to Privileged</i>	
<i>OS Resources</i>	152
<i>Elevated Processes That Read Data from Untrusted Sources</i>	153
<i>Attack Pattern: Use a User-Supplied Configuration File to Run</i>	
<i>Commands That Elevate Privilege</i>	153
<i>Processes That Use Elevated Components</i>	153
Finding Injection Points	154
<i>Watching Input Files</i>	155
<i>Attack Pattern: Make Use of Configuration File Search Paths</i>	156
Input Path Tracing	156
<i>Using GDB and IDA-Pro Together on a Solaris SPARC</i>	
<i>Binary</i>	156
<i>Setting Breakpoints and Expressions</i>	156
<i>Mapping Runtime Memory Addresses from IDA</i>	157
<i>Attaching to a Running Process</i>	158

<i>Using Truss to Model the Target on Solaris</i>	159
Exploiting Trust through Configuration	161
<i>Attack Pattern: Direct Access to Executable Files</i>	162
<i>Auditing for Directly Executable Files</i>	162
<i>Know the Current Working Directory (CWD)</i>	163
<i>What If the Web Server Won't Execute cgi Programs?</i>	163
<i>Attack Pattern: Embedding Scripts within Scripts</i>	164
<i>What About Nonexecutable Files?</i>	165
<i>Attack Pattern: Leverage Executable Code in Nonexecutable Files</i>	165
<i>Playing with Policy</i>	166
Specific Techniques and Attacks for Server Software	167
<i>Technique: Shell Command Injection</i>	167
<i>Attack Pattern: Argument Injection</i>	169
<i>Attack Pattern: Command Delimiters</i>	172
<i>Attack Pattern: Multiple Parsers and Double Escapes</i>	173
<i>Technique: Plumbing Pipes, Ports, and Permissions</i>	181
<i>Technique: Exploring the File System</i>	184
<i>Attack Pattern: User-Supplied Variable Passed to File System Calls</i>	185
<i>Attack Pattern: Postfix NULL Terminator</i>	186
<i>Attack Pattern: Postfix, Null Terminate, and Backslash</i>	186
<i>Attack Pattern: Relative Path Traversal</i>	187
<i>Technique: Manipulating Environment Variables</i>	189
<i>Attack Pattern: Client-Controlled Environment Variables</i>	189
<i>Technique: Leveraging Extraneous Variables</i>	190
<i>Attack Pattern: User-Supplied Global Variables (DEBUG=1, PHP Globals, and So Forth)</i>	190
<i>Technique: Leveraging Poor Session Authentication</i>	192
<i>Attack Pattern: Session ID, Resource ID, and Blind Trust</i>	192
<i>Technique: Brute Forcing Session IDs</i>	193
<i>Technique: Multiple Paths of Authentication</i>	198
<i>Technique: Failure to Check Error Codes</i>	199
Conclusion	199
5 Exploiting Client Software	201
Client-side Programs as Attack Targets	201
<i>The Server Controls the Client</i>	202
<i>Software Honey pots</i>	203

In-band Signals	204
<i>Ancient (But Relevant) History</i>	204
<i>Attack Pattern: Analog In-Band Switching Signals (aka "Blue Boxing")</i>	205
<i>Basic In-band Data Use</i>	207
<i>In-band Fun with Printers</i>	208
<i>In-band Terminal Character Injection in Linux</i>	209
<i>Attack Pattern Fragment: Manipulating Terminal Devices</i>	210
<i>The Reflection Problem</i>	211
Cross-site Scripting (XSS)	212
<i>Using Reflection against Trusted Sites</i>	213
<i>Attack Pattern: Simple Script Injection</i>	214
<i>Attack Pattern: Embedding Script in Nonscript Elements</i>	215
<i>Attack Pattern: XSS in HTTP Headers</i>	216
<i>Attack Pattern: HTTP Query Strings</i>	216
<i>Attack Pattern: User-Controlled Filename</i>	217
Clients Scripts and Malicious Code	217
<i>Auditing for Weak Local Calls</i>	219
<i>Web Browsers and ActiveX</i>	224
<i>Attack Pattern: Passing Local Filenames to Functions That Expect a URL</i>	225
<i>E-mail Injection</i>	226
<i>Attack Pattern: Meta-characters in E-mail Header</i>	226
Content-Based Attacks	229
<i>Attack Pattern: File System Function Injection, Content Based</i>	229
Backwash Attacks: Leveraging Client-side Buffer Overflows	230
<i>Attack Pattern: Client-side Injection, Buffer Overflow</i>	231
Conclusion	232
6	
Crafting (Malicious) Input	233
The Defender's Dilemma	235
<i>Filters</i>	236
<i>Communicating Systems</i>	237
Intrusion Detection (Not)	237
<i>Signature-Based versus Anomaly-Based IDSs</i>	238
<i>IDSs as a Reactive Subscription Service</i>	239
<i>The Effect of Alternate Encoding on IDSs</i>	240

Partition Analysis	242
<i>APISPY for Windows Revisited</i>	243
<i>Red Pointing</i>	243
Tracing Code	244
<i>Backtracing from Vulnerable Locations</i>	245
<i>Dead Ends and Runouts</i>	247
<i>Runtime Tracing</i>	247
<i>Speedbreaks</i>	250
<i>Tracing a Buffer</i>	251
<i>Leapfrogging</i>	251
<i>Memory Page Break Points</i>	253
<i>Boron Tagging</i>	253
Reversing Parser Code	254
<i>Character Conversion</i>	255
<i>Byte Operations</i>	255
<i>Pointer Operations</i>	256
<i>NULL Terminators</i>	257
Example: Reversing I-Planet Server 6.0 through the Front Door	258
Misclassification	263
<i>Attack Pattern: Cause Web Server Misclassification</i>	263
Building "Equivalent" Requests	264
<i>Mapping the API Layer</i>	264
<i>Ghost Characters</i>	266
<i>Attack Pattern: Alternate Encoding the Leading Ghost Characters</i>	267
<i>Equivalent Meta-characters</i>	268
<i>Attack Pattern: Using Slashes in Alternate Encoding</i>	268
<i>Escaped Meta-characters</i>	269
<i>Attack Pattern: Using Escaped Slashes in Alternate Encoding</i>	270
<i>Character Conversion</i>	271
<i>Attack Pattern: Unicode Encoding</i>	271
<i>Attack Pattern: UTF-8 Encoding</i>	273
<i>Attack Pattern: URL Encoding</i>	273
<i>Attack Pattern: Alternative IP Addresses</i>	274
<i>Combined Attacks</i>	274
<i>Attack Pattern: Slashes and URL Encoding Combined</i>	274
Audit Poisoning	275
<i>Attack Pattern: Web Logs</i>	275
Conclusion	276

7 Buffer Overflow 277

Buffer Overflow 101	277
<i>Smashing the Stack (for Fun and Profit)</i>	279
<i>Corrupting State</i>	279
Injection Vectors: Input Rides Again	280
<i>Where Injection Stops and Payload Begins</i>	282
<i>Choosing the Correct Code Address to Target</i>	282
<i>Highland and Lowland Addresses</i>	283
<i>Big Endian and Little Endian Representation</i>	284
<i>Using Registers</i>	285
<i>Using Existing Code or Data Blocks in Memory</i>	286
Buffer Overflows and Embedded Systems	286
<i>Embedded Systems in Military and Commercial Use</i>	287
Database Buffer Overflows	289
<i>Stored Procedures</i>	290
<i>Command-Line Applications</i>	290
<i>Clients of the Database</i>	290
Buffer Overflows and Java?!	291
<i>Using Java and C/C++ Together</i>	292
<i>Stored Procedures and DLLs</i>	293
Content-Based Buffer Overflow	293
<i>Attack Pattern: Overflow Binary Resource File</i>	293
<i>Attack Pattern: Overflow Variables and Tags</i>	294
<i>Attack Pattern: Overflow Symbolic Links</i>	294
<i>Attack Pattern: MIME Conversion</i>	295
<i>Attack Pattern: HTTP Cookies</i>	295
Audit Truncation and Filters with Buffer Overflow	296
<i>Attack Pattern: Filter Failure through Buffer Overflow</i>	296
Causing Overflow and Environment Variables	296
<i>Attack Pattern: Buffer Overflow with Environment Variables</i>	297
<i>Attack Pattern: Buffer Overflow in an API Call</i>	297
<i>Attack Pattern: Buffer Overflow in Local Command-Line Utilities</i>	297
The Multiple Operation Problem	298
<i>Attack Pattern: Parameter Expansion</i>	298
Finding Potential Buffer Overflows	298
<i>Exception Handling Hides Errors</i>	299
<i>Using a Disassembler</i>	299

Stack Overflow	300
<i>Fixed-Size Buffers</i>	301
<i>Functions That Do Not Automatically NULL Terminate</i>	302
<i>Functions with Off-By-One NULL Termination</i>	304
<i>Overwriting Exception Handler Frames</i>	308
Arithmetic Errors in Memory Management	309
<i>Negative Values Equal Large Values</i>	309
<i>Signed/Unsigned Mismatch</i>	310
<i>Signed Values and Memory Management</i>	315
Format String Vulnerabilities	317
<i>Printing Data from Anywhere in Memory</i>	319
<i>Detecting the Problem in Code</i>	324
<i>Attack Pattern: String Format Overflow in sys log()</i>	324
Heap Overflows	324
<i>Malloc and the Heap</i>	327
Buffer Overflows and C++	329
<i>Vtables</i>	329
Payloads	329
<i>Getting Your Bearings</i>	331
<i>Payload Size</i>	332
<i>Using Hard-Coded Function Calls</i>	332
<i>Using a Dynamic Jump Table</i>	333
<i>Locating the Data Section</i>	334
<i>XOR Protection</i>	335
<i>Checksum/Hash Loading</i>	335
Payloads on RISC Architectures	336
<i>"Branch Delay" or "Delay Slot"</i>	337
<i>MIPS-Based Payload Construction</i>	337
<i>MIPS Instructions</i>	337
<i>Getting Bearings</i>	338
<i>Avoiding NULL Bytes in MIPS Opcodes</i>	339
<i>Syscalls on MIPS</i>	340
<i>SPARC Payload Construction</i>	340
<i>SPARC Register Window</i>	341
<i>Walking the Stack on SPARC</i>	342
<i>Function Call Nesting in SPARC</i>	344
<i>PA-RISC Payload Construction</i>	345
<i>Walking the Stack on PA-RISC</i>	347
<i>Stack Overflow on HPUX PA-RISC</i>	349

<i>Inter-space Branching on the PA-RISC</i>	349
<i>Inter-space Trampolines</i>	351
<i>Getting Bearings</i>	351
<i>Self-Decrypting Payload on HPUX</i>	353
<i>ALX/PowerPC Payload Construction</i>	356
<i>Getting Bearings</i>	356
<i>Active Armor for the PowerPC Shell Code</i>	356
<i>Removing the NULL Characters</i>	358
Multiplatform Payloads	358
<i>Multiplatform nopSled</i>	360
Prolog/Epilog Code to Protect Functions	360
<i>Defeating Canary Values (aka StackGuard)</i>	361
<i>Defeating Nonexecutable Stacks</i>	364
Conclusion	366

8 Rootkits 367

Subversive Programs	367
<i>What Is a Rootkit?</i>	368
<i>What Is a Kernel Rootkit?</i>	368
<i>Kernel Rootkits and the Trusted Computing Base</i>	369
A Simple Windows XP Kernel Rootkit	369
<i>Writing a Rootkit</i>	369
<i>The Checked Build Environment</i>	369
<i>Files in the Rootkit Source</i>	370
<i>Building Things</i>	370
<i>Kernel Drivers</i>	370
<i>The Basic Structure of a Driver</i>	371
<i>When Programs Use a Driver</i>	372
<i>Allowing the Driver to Be Unloaded</i>	373
<i>Registering the Driver</i>	375
<i>Using SystemLoadAndCa7 7Image</i>	377
Call Hooking	380
<i>Hiding a Process</i>	381
<i>Hooking a System Call</i>	381
<i>Structure of Our Basic Call Hook</i>	381
<i>Removing a Process Record</i>	382
<i>Process Injection Alternative</i>	386
Trojan Executable Redirection	386
<i>Redirection and the Problem with Tripwire</i>	386
<i>The Redirection Driver</i>	387

- Hiding Files and Directories 392
- Patching Binary Code 394
 - Peephole Patches* 395
 - Patching the NT Kernel to Remove All Security* 397
- The Hardware Virus 408
 - Reading and Writing Hardware Memory* 410
 - Example: Read/Write to the Keyboard Hardware* 411
 - Enable Read/Write from EEPROM* 417
 - CIH* 417
 - EEPROM and Timing* 421
 - The Ethernet EEPROM* 421
 - Serial EEPROM versus Parallel EEPROM* 424
 - Burning Out Hardware* 425
 - Manufacturers* 425
 - Detecting Chips via Common Flash Interface (CFI)* 426
 - Example: Detect a Flash RAM Chip* 427
 - Detecting Chips via ID Mode or JEDEC ID* 427
- Low-Level Disk Access 429
 - Reading/Writing the Master Boot Record (MBR)* 429
 - Infecting CD-ROM Images* 429
- Adding Network Support to a Driver 430
 - Using the NDIS Library* 430
 - Putting the Interface in Promiscuous Mode* 432
 - Finding the Correct Network Card* 433
 - Using boron Tags for Security* 438
 - Adding an Interactive Shell* 438
- Interrupts 439
 - Intel Interrupt Request (IRQ) Architecture* 439
 - Hooking the Interrupt Descriptor Table (IDT)* 441
 - The Mystery of the Programmable Interrupt Controller (PIC)* 441
- Key Logging 443
 - Linux Key Logger* 443
 - Windows NT/2000/XP Key Logger* 443
 - The Keyboard Controller Chip* 444
- Advanced Rootkit Topics 444
 - Using a Rootkit as a Debugger* 445
 - Disabling Windows System File Protection* 445
 - Writing Directly to Physical Memory* 445
 - Kernal Buffer Overflows* 445
 - Infecting the Kernel Image* 446

Execute Redirection 446
Detecting Rootkits 446
Conclusion 446

References 449

Index 453