

BRIEF CONTENTS

Foreword	xv
Preface	xvii
Acknowledgements	xix
Introduction	1

PART I: BINARY FORMATS

Chapter 1: Anatomy of a Binary	11
Chapter 2: The ELF Format	31
Chapter 3: The PE Format: A Brief Introduction	57
Chapter 4: Building a Binary Loader Using libbfd	67

PART II: BINARY ANALYSIS FUNDAMENTALS

Chapter 5: Basic Binary Analysis in Linux	89
Chapter 6: Disassembly and Binary Analysis Fundamentals	115
Chapter 7: Simple Code Injection Techniques for ELF	155

PART III: ADVANCED BINARY ANALYSIS

Chapter 8: Customizing Disassembly	191
Chapter 9: Binary Instrumentation	223
Chapter 10: Principles of Dynamic Taint Analysis	265
Chapter 11: Practical Dynamic Taint Analysis with libdft	279
Chapter 12: Principles of Symbolic Execution	309
Chapter 13: Practical Symbolic Execution with Triton	333

PART IV: APPENDICES

Appendix A: A Crash Course on x86 Assembly	373
Appendix B: Implementing PT_NOTE Overwriting Using libelf	391
Appendix C: List of Binary Analysis Tools	413
Appendix D: Further Reading	417

Index419

CONTENTS IN DETAIL

FOREWORD	xv
-----------------	-----------

PREFACE	xvii
----------------	-------------

ACKNOWLEDGEMENTS	xix
-------------------------	------------

INTRODUCTION	1
What is Binary Analysis, and Why Do You Need It?	2
What Makes Binary Analysis Challenging?	3
What You Will and Won't Find in This Book	4
Who This Book is For	4
Conventions	5
Instruction Set Architecture	5
Assembly Syntax	5
Binary Format and Development Platform	6
Code Samples and Virtual Machine	6
Exercises	7
Topics Covered in This Book	7

PART I BINARY FORMATS

1	
ANATOMY OF A BINARY	11
The C Compilation Process	12
The Preprocessing Phase	12
The Compilation Phase	14
The Assembly Phase	16
The Linking Phase	17
Symbols and Stripped Binaries	18
Another Binary Turns to the Dark Side: Stripping a Binary	20
Disassembling a Binary	20
Looking Inside an Object File	21
Examining a Complete Binary Executable	23
Loading and Executing a Binary	26

Summary	28
Exercises	28

2 THE ELF FORMAT 31

The Executable Header	33
The e_ident Array	33
The e_type, e_machine, and e_version Fields	35
The e_entry Field	36
The e_phoff and e_shoff Fields	36
The e_flags Field	36
The e_ehsize Field	37
The e_*entsize and e_*num Fields	37
The e_shstrndx Field	37
Section Headers	38
The sh_name Field	39
The sh_type Field	39
The sh_flags Field	40
The sh_addr, sh_offset, and sh_size Fields	40
The sh_link Field	40
The sh_info Field	41
The sh_addralign Field	41
The sh_entsize Field	41
Sections	41
The .init and .fini Sections	43
The .text Section	43
The .bss, .data, and .rodata Sections	44
Lazy Binding and the .plt, .got, and .got.plt Sections	45
The .rel.* and .rela.* Sections	48
The .dynamic Section	49
The .init_array and .fini_array Sections	51
The .shstrtab, .symtab, .strtab, .dynsym, and .dynstr Sections	51
Program Headers	52
The p_type Field	53
The p_flags Field	54
The p_offset, p_vaddr, p_paddr, p_filesz, and p_memsz Fields	54
The p_align Field	55
Summary	55
Exercises	55

3 THE PE FORMAT: A BRIEF INTRODUCTION 57

The MS-DOS Header and MS-DOS Stub	58
The PE Signature, PE File Header, and PE Optional Header	58
The PE Signature	61

The PE File Header	61
The PE Optional Header	62
The Section Header Table	62
Sections	63
The .edata and .idata Sections	63
Padding in PE Code Sections	64
Summary	65
Exercises	65

4 BUILDING A BINARY LOADER USING LIBBFD 67

What Is libbfd?	68
A Simple Binary-Loading Interface	68
The Binary Class	71
The Section Class	71
The Symbol Class	71
Implementing the Binary Loader	72
Initializing libbfd and Opening a Binary	73
Parsing Basic Binary Properties	75
Loading Symbols	77
Loading Sections	81
Testing the Binary Loader	83
Summary	85
Exercises	85

PART II BINARY ANALYSIS FUNDAMENTALS

5 BASIC BINARY ANALYSIS IN LINUX 89

Resolving Identity Crises Using file	90
Using ldd to Explore Dependencies	92
Viewing File Contents with xxd	94
Parsing the Extracted ELF with readelf	96
Parsing Symbols with nm	99
Looking for Hints with strings	102
Tracing System Calls and Library Calls with strace and ltrace	105
Examining Instruction-Level Behavior Using objdump	109
Dumping a Dynamic String Buffer Using gdb	111
Summary	114
Exercises	114

6 DISASSEMBLY AND BINARY ANALYSIS FUNDAMENTALS 115

Static Disassembly	116
Dynamic Disassembly	121
Example: Tracing a Binary Execution with gdb	122
Code Coverage Strategies	125
Structuring Disassembled Code and Data	129
Structuring Code	129
Structuring Data	136
Decompilation	138
Intermediate Representations	139
Fundamental Analysis Methods	141
Binary Analysis Properties	142
Control-Flow Analysis	146
Data-Flow Analysis	148
Effects of Compiler Settings on Disassembly	152
Summary	153
Exercises	153

7 SIMPLE CODE INJECTION TECHNIQUES FOR ELF 155

Bare-metal Binary Modification Using Hex Editing	155
Observing an Off-by-one Bug in Action	156
Fixing the Off-by-one Bug	159
Modifying Shared Library Behavior Using LD_PRELOAD	162
A Heap Overflow Vulnerability	163
Detecting the Heap Overflow	165
Injecting a Code Section	169
Injecting an ELF Section: A High-level Overview	169
Using elfinject to Inject an ELF Section	171
Calling Injected Code	175
Entry Point Modification	175
Hijacking Constructors and Destructors	179
Hijacking GOT Entries	182
Hijacking PLT Entries	185
Redirecting Direct and Indirect Calls	186
Summary	187
Exercises	187

PART III ADVANCED BINARY ANALYSIS

8 CUSTOMIZING DISASSEMBLY 191

Why Write a Custom Disassembly Pass?	191
A Case for Custom Disassembly: Obfuscated Code	192
Other Reasons to Write a Custom Disassembler	194
Introduction to Capstone	196
Installing Capstone	196
Linear Disassembly with Capstone	198
Exploring the Capstone C API	203
Recursive Disassembly with Capstone	204
Implementing a ROP Gadget Scanner	213
Introduction to Return-Oriented Programming (ROP)	213
Finding ROP Gadgets	215
Summary	221
Exercises	221

9

BINARY INSTRUMENTATION **223**

What is Binary Instrumentation?	223
Binary Instrumentation APIs	224
Static versus Dynamic Binary Instrumentation	225
Static Binary Instrumentation	226
The INT 3 Approach	226
The Trampoline Approach	228
Dynamic Binary Instrumentation	233
Architecture of a DBI System	233
Introduction to Pin	235
Profiling with Pin	236
The Profiler's Data Structures and Setup Code	237
Parsing Function Symbols	240
Instrumenting Basic Blocks	241
Instrumenting Control Flow Instructions	243
Counting Instructions, Control Transfers, and Syscalls	246
Testing the Profiler	247
Automatic Binary Unpacking with Pin	250
Introduction to Executable Packers	250
The Unpacker's Data Structures and Setup Code	252
Instrumenting Memory Writes and Control Flow Instructions	254
Tracking Memory Writes	256
Detecting the Original Entry Point and Dumping the Unpacked Binary	257
Testing the Unpacker	259
Summary	263
Exercises	263

10

PRINCIPLES OF DYNAMIC TAINT ANALYSIS **265**

What is DTA?	266
--------------------	-----

DTA in Three Steps: Taint Sources, Taint Sinks, and Taint Propagation	266
Defining taint sources	266
Defining taint sinks	267
Tracking taint propagation	267
Using DTA to Detect the Heartbleed Bug	268
A Brief Overview of the Heartbleed Vulnerability	268
Detecting Heartbleed Through Tainting	269
DTA Design Factors: Taint Granularity, Taint Colors, and Taint Policies	271
Taint Granularity	271
Taint Colors	272
Taint Propagation Policies	273
Overtainting and Undertainting	275
Control Dependencies	275
Shadow Memory	276
Summary	278
Exercises	278

11

PRACTICAL DYNAMIC TAINT ANALYSIS WITH LIBDFT 279

Introducing libdft	279
Internals of libdft	280
Taint Policy	282
Using DTA to Detect Remote Control Hijacking	283
Checking Taint Information	286
Taint Sources: Tainting Received Bytes	287
Taint Sinks: Checking execve Arguments	289
Detecting a Control Flow Hijacking Attempt	291
Circumventing DTA with Implicit Flows	296
A DTA-based Data Exfiltration Detector	297
Taint Sources: Tracking Taint for Open Files	299
Taint Sinks: Monitoring Network Sends for Data Exfiltration	302
Detecting a Data Exfiltration Attempt	304
Summary	306
Exercises	307

12

PRINCIPLES OF SYMBOLIC EXECUTION 309

An Overview of Symbolic Execution	309
Symbolic versus Concrete Execution	310
Variants and Limitations of Symbolic Execution	314
Increasing the Scalability of Symbolic Execution	319
Constraint Solving with Z3	320
Proving Reachability of an Instruction	321
Proving Unreachability of an Instruction	324
Proving Validity of a Formula	325

Simplifying Expressions	326
Modeling Constraints for Machine Code with Bitvectors	327
Solving an Opaque Predicate over Bitvectors	329
Summary	330
Exercises	330

13 PRACTICAL SYMBOLIC EXECUTION WITH TRITON 333

Introduction to Triton	333
Maintaining Symbolic State with Abstract Syntax Trees	335
Backward Slicing with Triton	337
Triton Header Files and Configuring Triton	340
The Symbolic Configuration File	340
Emulating Instructions	342
Setting Triton’s Architecture	343
Computing the Backward Slice	344
Using Triton to Increase Code Coverage	345
Creating Symbolic Variables	348
Finding a Model for a New Path	348
Testing the Code Coverage Tool	351
Automatically Exploiting a Vulnerability	355
The Vulnerable Program	355
Finding the Address of the Vulnerable Call Site	359
Building the Exploit Generator	361
Getting a Root Shell	367
Summary	369
Exercises	370

PART IV APPENDICES

A A CRASH COURSE ON X86 ASSEMBLY 373

Layout of an Assembly Program	373
Assembly Instructions, Directives, Labels, and Comments	374
Separation Between Code and Data	375
AT&T versus Intel Syntax	375
Structure of an x86 Instruction	376
Assembly-level Representation of x86 Instructions	376
Machine-level Structure of x86 Instructions	376
Register Operands	377
Memory Operands	379
Immediates	380
Common x86 Instructions	380

Comparing Operands and Setting Status Flags	380
Implementing System Calls	382
Implementing Conditional Jumps	382
Loading Memory Addresses	382
Common Code Constructs in Assembly	383
The Stack	383
Function Calls and Function Frames	384
Conditional Branches	388
Loops	389

B

IMPLEMENTING PT_NOTE OVERWRITING USING LIBELF 391

Required Headers	392
Data Structures Used In elfinject	392
Initializing libelf	393
Getting the Executable Header	398
Finding the PT_NOTE Segment	398
Injecting the Code Bytes	400
Aligning the Load Address for the Injected Section	401
Overwriting the .note.ABI-tag Section Header	401
Setting the Name of the Injected Section	406
Overwriting the PT_NOTE Program Header	408
Modifying the Entry Point	411

C

LIST OF BINARY ANALYSIS TOOLS 413

Disassemblers	413
Debuggers	414
Disassembly Frameworks	415
Binary Analysis Frameworks	415

D

FURTHER READING 417

Standards and References	417
Papers and Articles	418
Books	418

INDEX 419