

Table of Contents

Preface	1
Chapter 1: Introduction to Malware Analysis	7
1. What Is Malware?	7
2. What Is Malware Analysis?	9
3. Why Malware Analysis?	9
4. Types Of Malware Analysis	10
5. Setting Up The Lab Environment	11
5.1 Lab Requirements	12
5.2 Overview Of Lab Architecture	12
5.3 Setting Up And Configuring Linux VM	14
5.4 Setting Up And Configuring Windows VM	21
6. Malware Sources	24
Summary	25
Chapter 2: Static Analysis	27
1. Determining the File Type	27
1.1 Identifying File Type Using Manual Method	28
1.2 Identifying File Type Using Tools	29
1.3 Determining File Type Using Python	29
2. Fingerprinting the Malware	31
2.1 Generating Cryptographic Hash Using Tools	31
2.2 Determining Cryptographic Hash in Python	32
3. Multiple Anti-Virus Scanning	32
3.1 Scanning the Suspect Binary with VirusTotal	33
3.2 Querying Hash Values Using VirusTotal Public API	34
4. Extracting Strings	36
4.1 String Extraction Using Tools	37
4.2 Decoding Obfuscated Strings Using FLOSS	39
5. Determining File Obfuscation	40
5.1 Packers and Cryptors	41
5.2 Detecting File Obfuscation Using Exeinfo PE	43
6. Inspecting PE Header Information	44
6.1 Inspecting File Dependencies and Imports	45
6.2 Inspecting Exports	49
6.3 Examining PE Section Table And Sections	50
6.4 Examining the Compilation Timestamp	53
6.5 Examining PE Resources	54
7. Comparing And Classifying The Malware	56
7.1 Classifying Malware Using Fuzzy Hashing	57

7.2 Classifying Malware Using Import Hash	59
7.3 Classifying Malware Using Section Hash	60
7.4 Classifying Malware Using YARA	61
7.4.1 Installing YARA	61
7.4.2 YARA Rule Basics	62
7.4.3 Running YARA	63
7.4.4 Applications of YARA	64
Summary	69
Chapter 3: Dynamic Analysis	71
1. Lab Environment Overview	72
2. System And Network Monitoring	73
3. Dynamic Analysis (Monitoring) Tools	73
3.1 Process Inspection with Process Hacker	74
3.2 Determining System Interaction with Process Monitor	75
3.3 Logging System Activities Using Noriben	76
3.4 Capturing Network Traffic With Wireshark	78
3.5 Simulating Services with INetSim	79
4. Dynamic Analysis Steps	82
5. Putting it All Together: Analyzing a Malware Executable	82
5.1 Static Analysis of the Sample	83
5.2 Dynamic Analysis of the Sample	85
6. Dynamic-Link Library (DLL) Analysis	88
6.1 Why Attackers Use DLLs	90
6.2 Analyzing the DLL Using rundll32.exe	91
6.2.1 Working of rundll32.exe	91
6.2.2 Launching the DLL Using rundll32.exe	92
Example 1 – Analyzing a DLL With No Exports	92
Example 2 – Analyzing a DLL Containing Exports	94
Example 3 – Analyzing a DLL Accepting Export Arguments	95
6.3 Analyzing a DLL with Process Checks	96
Summary	98
Chapter 4: Assembly Language and Disassembly Primer	99
1. Computer Basics	100
1.1 Memory	101
1.1.1 How Data Resides In Memory	102
1.2 CPU	102
1.2.1 Machine Language	102
1.3 Program Basics	103
1.3.1 Program Compilation	103
1.3.2 Program On Disk	103
1.3.3 Program In Memory	105
1.3.4 Program Disassembly (From Machine code To Assembly code)	108
2. CPU Registers	109
2.1 General-Purpose Registers	109
2.2 Instruction Pointer (EIP)	110

2.3 EFLAGS Register	110
3. Data Transfer Instructions	110
3.1 Moving a Constant Into Register	110
3.2 Moving Values From Register To Register	111
3.3 Moving Values From Memory To Registers	111
3.4 Moving Values From Registers To Memory	113
3.5 Disassembly Challenge	114
3.6 Disassembly Solution	114
4. Arithmetic Operations	116
4.1 Disassembly Challenge	117
4.2 Disassembly Solution	118
5. Bitwise Operations	120
6. Branching And Conditionals	121
6.1 Unconditional Jumps	122
6.2 Conditional Jumps	122
6.3 If Statement	123
6.4 If-Else Statement	124
6.5 If-Elseif-Else Statement	125
6.6 Disassembly Challenge	126
6.7 Disassembly Solution	126
7. Loops	129
7.1 Disassembly Challenge	131
7.2 Disassembly Solution	132
8. Functions	134
8.1 Stack	134
8.2 Calling Function	136
8.3 Returning From Function	136
8.4 Function Parameters And Return Values	136
9. Arrays And Strings	142
9.1 Disassembly Challenge	143
9.2 Disassembly Solution	144
9.3 Strings	148
9.3.1 String Instructions	149
9.3.2 Moving From Memory To Memory (movsx)	149
9.3.3 Repeat Instructions (rep)	150
9.3.4 Storing Value From Register to Memory (stosx)	151
9.3.5 Loading From Memory to Register (lodsx)	151
9.3.6 Scanning Memory (scasx)	151
9.3.7 Comparing Values in Memory (cmpsx)	151
10. Structures	152
11. x64 Architecture	153
11.1 Analyzing 32-bit Executable On 64-bit Windows	155
12. Additional Resources	156
Summary	156
Chapter 5: Disassembly Using IDA	157

1. Code Analysis Tools	157
2. Static Code Analysis (Disassembly) Using IDA	158
2.1 Loading Binary in IDA	159
2.2 Exploring IDA Displays	161
2.2.1 Disassembly Window	161
2.2.2 Functions Window	163
2.2.3 Output Window	164
2.2.4 Hex View Window	164
2.2.5 Structures Window	164
2.2.6 Imports Window	164
2.2.7 Exports Window	165
2.2.8 Strings Window	165
2.2.9 Segments Window	165
2.3 Improving Disassembly Using IDA	166
2.3.1 Renaming Locations	168
2.3.2 Commenting in IDA	169
2.3.3 IDA Database	170
2.3.4 Formatting Operands	172
2.3.5 Navigating Locations	172
2.3.6 Cross-References	173
2.3.7 Listing All Cross-References	176
2.3.8 Proximity View And Graphs	177
3. Disassembling Windows API	179
3.1 Understanding Windows API	180
3.1.1 ANSI and Unicode API Functions	185
3.1.2 Extended API Functions	185
3.2 Windows API 32-Bit and 64-Bit Comparison	185
4. Patching Binary Using IDA	188
4.1 Patching Program Bytes	189
4.2 Patching Instructions	191
5. IDA Scripting and Plugins	192
5.1 Executing IDA Scripts	192
5.2 IDAPython	193
5.2.1 Checking The Presence Of CreateFile API	194
5.2.2 Code Cross-References to CreateFile Using IDAPython	195
5.3 IDA Plugins	196
Summary	196
Chapter 6: Debugging Malicious Binaries	197
1. General Debugging Concepts	198
1.1 Launching And Attaching To Process	198
1.2 Controlling Process Execution	199
1.3 Interrupting a Program with Breakpoints	200
1.4 Tracing Program Execution	201
2. Debugging a Binary Using x64dbg	201
2.1 Launching a New Process in x64dbg	202
2.2 Attaching to an Existing Process Using x64dbg	203

2.3 x64dbg Debugger Interface	204
2.4 Controlling Process Execution Using x64dbg	208
2.5 Setting a Breakpoint in x64dbg	208
2.6 Debugging 32-bit Malware	209
2.7 Debugging 64-bit Malware	210
2.8 Debugging a Malicious DLL Using x64dbg	213
2.8.1 Using rundll32.exe to Debug the DLL in x64dbg	214
2.8.2 Debugging a DLL in a Specific Process	215
2.9 Tracing Execution in x64dbg	216
2.9.1 Instruction Tracing	218
2.9.2 Function Tracing	219
2.10 Patching in x64dbg	220
3. Debugging a Binary Using IDA	221
3.1 Launching a New Process in IDA	222
3.2 Attaching to an Existing Process Using IDA	222
3.3 IDA's Debugger Interface	223
3.4 Controlling Process Execution Using IDA	226
3.5 Setting a Breakpoint in IDA	226
3.6 Debugging Malware Executables	228
3.7 Debugging a Malicious DLL Using IDA	229
3.7.1 Debugging a DLL in a Specific Process	231
3.8 Tracing Execution Using IDA	232
3.9 Debugger Scripting Using IDAPython	235
3.9.1 Example – Determining Files Accessed by Malware	237
4. Debugging a .NET Application	239
Summary	241
Chapter 7: Malware Functionalities and Persistence	243
1. Malware Functionalities	243
1.1 Downloader	243
1.2 Dropper	245
1.2.1 Reversing a 64-bit Dropper	247
1.3 Keylogger	247
1.3.1 Keylogger Using GetAsyncKeyState()	248
1.3.2 Keylogger Using SetWindowsHookEx()	249
1.4 Malware Replication Via Removable Media	250
1.5 Malware Command and Control (C2)	255
1.5.1 HTTP Command and Control	255
1.5.2 Custom Command and Control	259
1.6 PowerShell-Based Execution	262
1.6.1 PowerShell Command Basics	263
1.6.2 PowerShell Scripts And Execution Policy	264
1.6.2 Analyzing PowerShell Commands/Scripts	265
1.6.3 How Attackers Use PowerShell	266
2. Malware Persistence Methods	268
2.1 Run Registry Key	268
2.2 Scheduled Tasks	269

2.3 Startup Folder	270
2.4 Winlogon Registry Entries	271
2.5 Image File Execution Options	272
2.6 Accessibility Programs	273
2.7 Applnit_DLLs	275
2.8 DLL Search Order Hijacking	276
2.9 COM hijacking	278
2.10 Service	281
Summary	286
Chapter 8: Code Injection and Hooking	287
1. Virtual Memory	288
1.1 Process Memory Components (User Space)	291
1.2 Kernel Memory Contents (Kernel Space)	292
2. User Mode And Kernel Mode	293
2.1 Windows API Call Flow	295
3. Code Injection Techniques	297
3.1 Remote DLL Injection	299
3.2 DLL Injection Using APC (APC Injection)	302
3.3 DLL Injection Using SetWindowsHookEx()	304
3.4 DLL Injection Using The Application Compatibility Shim	306
3.4.1 Creating A Shim	307
3.4.2 Shim Artifacts	312
3.4.3 How Attackers Use Shims	313
3.4.4 Analyzing The Shim Database	314
3.5 Remote Executable/Shellcode Injection	315
3.6 Hollow Process Injection (Process Hollowing)	317
4. Hooking Techniques	322
4.1 IAT Hooking	322
4.2 Inline Hooking (Inline Patching)	324
4.3 In-memory Patching Using Shim	326
5. Additional Resources	330
Summary	331
Chapter 9: Malware Obfuscation Techniques	333
1. Simple Encoding	335
1.1 Caesar Cipher	335
1.1.1 Working Of Caesar Cipher	335
1.1.2 Decrypting Caesar Cipher In Python	337
1.2 Base64 Encoding	338
1.2.1 Translating Data To Base64	338
1.2.2 Encoding And Decoding Base64	339
1.2.3 Decoding Custom Base64	341
1.2.4 Identifying Base64	344
1.3 XOR Encoding	345
1.3.1 Single Byte XOR	346
1.3.2 Finding XOR Key Through Brute-Force	349

1.3.3 NULL Ignoring XOR Encoding	350
1.3.4 Multi-byte XOR Encoding	352
1.3.5 Identifying XOR Encoding	354
2. Malware Encryption	355
2.1 Identifying Crypto Signatures Using Signsrch	355
2.2 Detecting Crypto Constants Using FindCrypt2	359
2.3 Detecting Crypto Signatures Using YARA	359
2.4 Decrypting In Python	361
3. Custom Encoding/Encryption	362
4. Malware Unpacking	367
4.1 Manual Unpacking	368
4.1.1 Identifying The OEP	368
4.1.2 Dumping Process Memory With Scylla	372
4.1.3 Fixing The Import Table	373
4.2 Automated Unpacking	374
Summary	377
Chapter 10: Hunting Malware Using Memory Forensics	379
1. Memory Forensics Steps	380
2. Memory Acquisition	380
2.1 Memory Acquisition Using DumpIt	381
3. Volatility Overview	384
3.1 Installing Volatility	384
3.1.1 Volatility Standalone Executable	384
3.1.2 Volatility Source Package	385
3.2 Using Volatility	386
4. Enumerating Processes	388
4.1 Process Overview	389
4.1.1 Examining the _EPROCESS Structure	390
4.1.2 Understanding _ActiveProcessLinks	394
4.2 Listing Processes Using psscan	396
4.2.1 Direct Kernel Object Manipulation (DKOM)	397
4.2.2 Understanding Pool Tag Scanning	398
4.3 Determining Process Relationships	401
4.4 Process Listing Using psxview	402
5. Listing Process Handles	404
6. Listing DLLs	406
6.1 Detecting a Hidden DLL Using ldrmodules	410
7. Dumping an Executable and DLL	411
8. Listing Network Connections and Sockets	413
9. Inspecting Registry	415
10. Investigating Service	417
11. Extracting Command History	419
Summary	421
Chapter 11: Detecting Advanced Malware Using Memory Forensics	423

1. Detecting Code Injection	424
1.1 Getting VAD Information	425
1.2 Detecting Injected Code Using VAD	427
1.3 Dumping The Process Memory Region	429
1.4 Detecting Injected Code Using malfind	430
2. Investigating Hollow Process Injection	431
2.1 Hollow Process Injection Steps	431
2.2 Detecting Hollow Process Injection	433
2.3 Hollow Process Injection Variations	435
3. Detecting API Hooks	438
4. Kernel Mode Rootkits	439
5. Listing Kernel Modules	440
5.1 Listing Kernel Modules Using driverscan	443
6. I/O Processing	444
6.1 The Role Of The Device Driver	447
6.2 The Role Of The I/O Manager	454
6.3 Communicating With The Device Driver	455
6.4 I/O Requests To Layered Drivers	457
7. Displaying Device Trees	461
8. Detecting Kernel Space Hooking	464
8.1 Detecting SSDT Hooking	464
8.2 Detecting IDT Hooking	467
8.3 Identifying Inline Kernel Hooks	468
8.4 Detecting IRP Function Hooks	470
9. Kernel Callbacks And Timers	473
Summary	479
Other Books You May Enjoy	481
Index	485
