



Malware Analysis and Detection Engineering

A Comprehensive Approach to
Detect and Analyze Modern Malware

Abhijit Mohanta, Anoop Saldanha

Foreword by Pedram Amini

Apress®

Malware Analysis and Detection Engineering

A Comprehensive Approach to Detect
and Analyze Modern Malware

Abhijit Mohanta, Anoop Saldanha

Foreword by Pedram Amini

Apress®

Malware Analysis and Detection Engineering: A Comprehensive Approach to Detect and Analyze Modern Malware

Abhijit Mohanta
Independent Cybersecurity Consultant,
Bhubaneswar, Odisha, India

Anoop Saldanha
Independent Cybersecurity Consultant,
Mangalore, Karnataka, India

ISBN-13 (pbk): 978-1-4842-6192-7
<https://doi.org/10.1007/978-1-4842-6193-4>

ISBN-13 (electronic): 978-1-4842-6193-4

Copyright © 2020 by Abhijit Mohanta, Anoop Saldanha

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Celestin Suresh John
Development Editor: Laura Berendson
Coordinating Editor: Divya Modi

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at <http://www.apress.com/bulk-sales>.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484261927. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

Table of Contents

About the Authors.....xxi

About the Technical Reviewerxxiii

About the Foreword Authorxxv

Acknowledgmentsxxvii

Forewordxxix

Introductionxxxi

Part I: Introduction 1

Chapter 1: Introduction..... 3

 Types of Malware..... 5

 Platform Diversity 7

 Target Diversity..... 8

 The Cyber Kill Chain 8

 Malware Attack Life Cycle..... 10

 Development Phase..... 11

 Distribution Phase: The Diverse Transport System 13

 Infection Phase 14

 Post-Infection Phase 14

 The Malware Business Model 14

 The War Against Malware 15

 The Combat Teams 16

 Anti-malware Products..... 19

 Terminologies..... 20

 Summary..... 24

TABLE OF CONTENTS

Chapter 2: Malware Analysis Lab Setup 25

Host System Requirements..... 26

Network Requirements 27

Creating the Malware Analysis VM..... 28

Tweaking an Analysis VM..... 29

 Disable Hidden Extensions 29

 Show Hidden Files and Folders 30

 Disable ASLR 32

 Disable Windows Firewall..... 33

 Disable Windows Defender (or Any Antivirus)..... 33

 Mimic an End-User System 34

Snapshots 35

Tools..... 36

 Hashing Tools: HashMyFiles and Others..... 38

 APIMiner 39

 PE File Exploration: CFF Explorer and PEView 39

 File Type Identification Tools..... 39

 Process Hacker, Process Explorer, CurrProcess 40

 ProcMon: Process Monitor..... 40

 Autoruns 40

 Regshot 41

 NTTrace..... 41

 FakeNet 41

 BinText..... 42

 YARA 42

 Wireshark 43

 Microsoft Network Monitor..... 43

 OllyDbg 2.0 44

 Notepad++ 44

 Malzilla 44

 PEiD 45

FTK Imager Lite	45
Volatility Standalone	45
Ring3 API Hook Scanner	45
GMER	45
SSDTreeView	46
DriverView	46
Strings	46
SimpleWMIView	46
Registry Viewer	46
Bulk Extractor	47
Suricata	47
Cuckoo Sandbox	47
rundll32	48
oledump.py	48
OllyDumpEx	48
DocFileViewerEx	49
Fiddler	49
IDA Pro	49
x64dbg and Immunity Debugger	49
Summary	50
Part II: OS and System Fundamentals	51
Chapter 3: Files and File Formats	53
Visualizing a File in Its Native Hex Form	53
Hash: Unique File Fingerprint	55
Identifying Files	57
File Extension	58
File Format: The Real File Extension	64
Manual Identification of File Formats	68
Summary	69

Chapter 4: Virtual Memory and the Portable Executable (PE) File 71

 Process Creation 71

 Executing the Program 72

 Exploring the Process with Process Hacker 74

 Virtual Memory 76

 Addressability 79

 Memory Pages 79

 Demand Paging 81

 Page Table 82

 Division of Virtual Memory Address Space 83

 Inspecting Pages Using Process Hacker 84

 Strings in Virtual Memory 89

 Using Virtual Memory Against Malware 92

 Portable Executable File 93

 Exploring Windows Executable 94

 Important PE Headers and Fields 102

 Dynamic-Link Library (DLL) 111

 Import Address Table 118

 Summary 122

Chapter 5: Windows Internals 123

 Win32 API 123

 Obtaining API Logs 124

 Win32 DLLs 124

 Studying Win32 API and MSDN Docs 126

 Behavior Identification with APIs 134

 Windows Registry 136

 Logical View of Registry 136

 Data Storage in Registry 139

 Adding Storage to the Registry 139

 Malware and Registry Love Affair 140

Important Directories on Windows.....	141
system32.....	141
Program Files	142
User Document and Settings.....	142
What to Look for as a Malware Analyst	143
Windows Processes	144
Attributes of a Process and Malware Anomalies	144
Windows Services.....	150
Executable Service Under SVCHOST.EXE.....	152
DLL Services under Svchost.....	153
Malware as Windows Services.....	156
Syscall.....	159
Mutants/Mutex.....	160
Summary.....	162
Part III: Malware Components and Analysis.....	163
Chapter 6: Malware Components and Distribution.....	165
Malware Components	165
Payload.....	166
Packer	167
Persistence.....	168
Communication	168
Propagation	169
Armoring.....	169
Stealth	170
Distribution Mechanisms	170
Exploits and Exploit Kits	172
Spam	181
Infected Storage Devices.....	183
Malvertising.....	184
Drive-by Download	184

TABLE OF CONTENTS

Downloaders.....	185
Direct Login via Weak Authentication	185
Shared Folders	186
Summary.....	188
Chapter 7: Malware Packers	189
Encryption and Compression	189
Packers	191
How Packers Work.....	191
Cryptors and Protectors.....	193
Installers.....	193
Let's Pack.....	194
Comparing Packed and Unpacked Samples.....	195
Identifying Packed Samples.....	197
Entropy	198
Strings	199
Identifying Packers	206
PEiD Tool.....	207
Code at the Entry Point.....	208
Section Names	209
Custom Packers.....	210
Misconceptions About Packers	211
Summary.....	211
Chapter 8: Persistence Mechanisms	213
Resources Used for Persistence	213
Analysis Tools.....	214
Autoruns	214
ProcMon	215
Startup Shell Directories.....	217
Registry RUN	221
Services	225

File Infection	228
DLL Hijacking	229
Winlogon	230
Task Scheduler.....	231
Debuggers.....	233
Image File Execution Option (IFE0).....	233
SilentProcessExit.....	234
Summary.....	236
Chapter 9: Network Communication.....	237
Why Communicate?	237
CnC Servers, Relays, Proxies, and Malware Networks	240
Resolving CnC Server IPs.....	242
Fixed IP Addresses	242
Fixed Domain Names.....	242
Domain Flux and DGA	243
CnC/Data Exfiltration Methods	247
HTTP	247
IRC	251
Other Methods.....	253
Lateral Movement	254
Reconnaissance	254
Credential Stealing/Exploit Preparation.....	256
Gaining Access	258
SMB, PsExec, and Others.....	258
Detecting Network Communication	259
Networking APIs and API Logs with APIMiner.....	259
String Analysis.....	262
IP and Domain Reputation	262
Static Signatures: IDS and Firewalls	263
Anomaly Baselines	264
Summary.....	265

Chapter 10: Code Injection, Process Hollowing, and API Hooking..... 267

What Is Code Injection? 267

Code Injection Motives..... 268

 Hiding 268

 Process Piggybacking 269

 Altering Functionality..... 270

Code Injection Target 271

Popular Code Injection Techniques 272

Steps for Code Injection..... 272

 Steps for Process User-Mode Code Injection 273

Classical DLL Injection 291

Process Hollowing 297

Classical Shellcode Injection 303

Reflective DLL Injection 305

Important APIs to Remember 305

Why Do These Malicious APIs Exist? 307

Code/API Hooking 307

 Identify Hooking Point/Target 308

 Placing Hooks in User Space..... 310

 Why a Malware Hooks? 313

 Application of Hooks in Security Software 318

 Hook Scanning Tools 318

 Case Study: DeleteFile() Hooked..... 321

 Case Study: Internet Explorer Hooked 324

 APIMiner 327

 Summary 329

Chapter 11: Stealth and Rootkits..... 331

Why Stealth?..... 332

Simple Stealth Techniques..... 332

 File Properties and Permissions..... 332

 Thumbnail Faking..... 339

Filename Faking and Extension Faking	341
Stealing System File Names.....	342
The Psycholinguistic Technique.....	342
Hiding Process Window	344
Code Injection	345
Rootkits.....	346
User Mode Rootkits	346
Kernel Mode Rootkits	350
Other Ways of Creating Rootkits	373
Summary.....	373
Part IV: Malware Analysis and Classification	375
Chapter 12: Static Analysis	377
Why Static Analysis?	377
Sample Hash for Information Xchange	378
Hash Generation	379
Internet, Blogs, and Analysis Reports.....	380
VirusTotal and Other Analysis Platforms	381
They Say It's Clean! Is It?	383
Figuring Out the File Format	384
Obtain Full Infection Context.....	385
Filename Faking and Extension Faking	386
File Thumbnail Faking.....	388
File Type and File Extension Mismatch	390
Version Information/Details.....	391
Code Signer Information	392
String Analysis Statically.....	394
Strings That Indicate Maliciousness.....	397
YARA.....	399
Where Does YARA Fail?	401

TABLE OF CONTENTS

Static Fail: Feeder for Dynamic Analysis.....	401
Summary.....	402
Chapter 13: Dynamic Analysis.....	403
Keep Your Base Snapshot Handy	403
First Run: A Bird's-Eye View	404
Whoa! The Sample Refuses to Run!	405
Run as Administrator or Not	406
Case Study 1.....	406
Case Study 2.....	408
Case Study 3.....	410
APIMiner: API Log Behavior Identification	412
Classify the Malware Family.....	415
String Analysis Dynamically.....	416
File Type.....	417
Version Information/Details	417
Packed or Unpacked?	418
Dynamic Observation of Strings in Memory	420
ProcMon: Behavior Events Analysis	424
AutoRuns	426
Detecting Code Injection	428
GMER and Ring3 API Hook Scanner.....	429
Yara on Live Processes	429
Other Malicious Behavior	429
Stealing System File Names for Stealth	430
Weird File and Process Names.....	430
Disappearing Executable.....	430
Number of Process Instances.....	431
Process Session IDs	431
Summary.....	431

Chapter 14: Memory Forensics with Volatility	433
What Are Memory Forensics?	433
Why Another Technique?	434
Memory Forensics Steps	436
Memory Acquisition	436
Sample-14-1.mem	437
Sample-14-2.mem	438
Sample-14-3.mem	439
Memory Analysis/Forensics	439
Volatility Command Format	443
Image Information	444
Listing Processes and Services	445
Virtual Memory Inspection	449
Listing Process Modules	453
Listing Handles	456
Scanning Registry	460
Identifying Code Injection and API Hooking	467
Inspecting the Kernel	471
Network Communication	474
Summary	476
Chapter 15: Malware Payload Dissection and Classification	477
Malware Type, Family, Variant, and Clustering	478
Nomenclature	479
Importance of Classification	481
Proactive Detection	481
Correct Remediation	481
Intelligence	482
Intention and Scope of Attack	482
Classification Basis	483

TABLE OF CONTENTS

KeyLogger 488

 Hooking Keyboard Messages 488

 Getting Keyboard Status 490

Information Stealers (PWS) 492

 Dynamic Events and API Logs 493

 String Analysis of Info Stealers..... 494

Banking Malware 496

 API Logs and Hook Scanners..... 496

 String Analysis on Banking Trojans 498

Point-of-Sale (POS) Malware 501

 How POS Devices Work 501

 How POS Malware Work 503

 Identifying and Classifying POS 504

 Strings In POS Malware..... 505

ATM Malware 508

RATs 509

 Identifying RATs..... 510

 Strings in RAT Malware 511

Ransomware 512

 Identifying Ransomware..... 513

 Strings in Ransomware 516

Cryptominer 517

Virus (File Infectors) 519

Summary..... 521

Part V: Malware Reverse Engineering 523

Chapter 16: Debuggers and Assembly Language 525

 Reversing and Disassemblers: Source ➤ Assembly ➤ Back..... 526

 PE and Machine Code 527

 x86 Assembly Language 529

 Instruction: The Format 530

 Registers 536

Important x86 Instructions	542
Other Instructions and Reference Manual	558
Debuggers and Disassembly.....	558
Debugger Basics	559
OllyDbg vs. IDA Pro.....	559
Exploring OllyDbg	560
Exploring IDA Debugger.....	575
Notations in OllyDbg and IDA.....	581
Identifying Code Constructs in Assembly.....	585
Identifying The Stack Frame.....	585
Identifying a Function Epilogue and Prologue	588
Identifying Local Variables.....	589
Identifying Pointers.....	592
Identifying Global Variables	594
Identifying Array on Stack	596
Identifying Structures on Stack	599
Function Call Parameter Identification	602
Identifying Branch Conditions.....	605
Identifying Loops	607
Making Disassembly Readable	610
Color Coding	610
Labels and Comments	610
Tracking Variables	612
Accelerating Disassembly Analysis.....	613
Skipping Compiler Stub and Library Code.....	613
Condensing Instructions With Algebra.....	614
Using Decompilers.....	615
Blocks and Flowcharts	617
References (XREF)	619
Advance Usage of Debuggers	626
Observing API Calls and Parameters	627
Breaking on Win32 APIs.....	628

TABLE OF CONTENTS

Debugger Events 632

Patching..... 634

Call Stack..... 636

Summary 637

Chapter 17: Debugging Tricks for Unpacking Malware 639

Unpacking Internals 640

 OEP and Payload..... 640

 Execution of a Packed Binary 641

Manual Unpacking Using Debuggers 649

 Fast Unpacking Using API Logs and APIMiner 649

 Debugging Tricks for Known Packers 651

 Other Tricks 658

 Compiler Stubs to Identify OEP 658

 Back Tracing 659

 Are We Inside the Payload? 663

Variations in Unpacking Techniques 663

Summary..... 664

Chapter 18: Debugging Code Injection 665

API Logs and Breakpoints 665

IEP: Injection Entry Point 666

Locating IEP with CreateRemoteThread..... 667

Locating IEP with Thread Context 675

The EBFEB Trick 686

Summary..... 689

Chapter 19: Armoring and Evasion: The Anti-Techniques 691

Armoring Techniques 691

 Anti-Static Analysis 691

 Anti-Dynamic Analysis..... 692

Anti-Debugging.....	704
Anti-Disassembly Using Garbage Code	712
Evasion Techniques.....	713
Antivirus Evasion	713
Network Security Evasion.....	714
Sandbox Evasion	715
Fooling Malware Armoring.....	718
Open Source Anti-Projects	719
Summary.....	720
Chapter 20: Fileless, Macros, and Other Malware Trends	721
Windows Scripting Environment.....	721
Obfuscation.....	723
Hex Equivalents	724
Splits and Joins	725
Inserting Junk.....	726
Expression Evaluation with eval	727
Encryption Algorithms	728
Deobfuscation	729
Static Deobfuscation	730
Dynamic Deobfuscation.....	732
Embedded Script Debuggers.....	734
The Payload.....	738
Downloaders and Droppers	738
Exploits.....	740
VBScript Malware	741
Microsoft Office Malware.....	743
OLE File Format	743
Dissecting the OLE Format	745
Extracting Streams	747
Macros.....	749

TABLE OF CONTENTS

Fileless Malware 757

 Windows Management Instrumentation (WMI) 757

 PowerShell 761

Summary..... 766

Part VI: Detection Engineering..... 769

Chapter 21: Dev Analysis Lab Setup..... 771

 Linux VM..... 771

 Suricata Setup..... 773

 APIMiner and Cuckoo Monitor Setup..... 776

 Windows VM..... 776

 Visual Studio Installation 777

 Cygwin Installation 780

 Cygwin + Visual Studio..... 782

 Other Tools..... 783

Summary..... 784

Chapter 22: Antivirus Engines 785

 Main Components of Antiviruses 785

 Signatures and Signature Module..... 787

 Signature Categories 789

 Caveats..... 802

 Signature Optimization 803

 Risk Minimization 806

 File Scanner 807

 Unpacker Module 809

 Memory Scanner..... 811

 Hook and Rootkit Detection Modules..... 812

 Viral Polymorphism and Emulators 814

 Remediation Module 815

 Next-Gen Antiviruses 815

Summary..... 817

Chapter 23: IDS/IPS and Snort/Suricata Rule Writing	819
Network Traffic Flow	820
North-South Traffic	820
East-West Traffic	821
Network Traffic Analysis.....	821
Network Security with IDPS.....	822
IDS vs. IPS	822
IDS: traffic Feed Mechanisms.....	823
IPS Traffic Feed.....	825
Pseudo IPS = IDS + Pseudo Inline.....	827
Deployment Quirks for IDPS Sensors.....	827
IDPS Components	828
Packet Capture Module	829
Packet Layer Decoding.....	831
TCP Stream Reassembly Module.....	832
App Layer Parsing	832
Detection Engine	833
Logging Module	834
Rule Language.....	834
Suricata IDPS	835
Yaml Config.....	835
Running Suricata in PCAP File Mode	837
Rule Writing with Suricata	838
Basic Rule Structure.....	839
IP-Only Rules	843
Keywords.....	843
Other Keywords	849
Summary.....	850
Chapter 24: Malware Sandbox Internals	851
What Is a Malware Sandbox?.....	851
Why Malware Sandbox?.....	853

TABLE OF CONTENTS

Sandbox In Your Security Architecture 854

Sandbox Design 856

 Sample Analysis Workflow 857

 Guest 858

 Host and Guest Agents 861

 Monitoring Agent 865

 Deception and Other Agents..... 867

 Communication Channel Host <-> Guest..... 867

 Logging Technique: Files vs. Streaming 868

Writing Detection on Sandbox Results..... 868

Machine Learning Using Sandbox..... 869

Summary..... 870

Chapter 25: Binary Instrumentation for Reversing Automation 871

 What Is Binary Instrumentation?..... 871

 DBI: Terminologies and Internals 874

 Inserting Instrumentation Code 876

 DBI for Malware Analysis 878

 Cons..... 879

 Tool Writing Using DBI..... 879

 Setting up PIN..... 880

 Tool 1: Logging All Instructions..... 884

 Tool 2: Win32 API Logging..... 888

 Tool 3: Code Modification and Branch Bypass..... 891

 Summary..... 895

Index..... 897

About the Authors



Abhijit Mohanta is an independent cybersecurity consultant and corporate trainer who has worked extensively in malware reverse engineering, vulnerability research, antivirus engine development, anti-malware signature writing, and sandbox development. He has worked with Symantec, McAfee, and Juniper Networks anti-malware labs. He holds several patents. He blogs regularly and has been a speaker at security conferences and workshops.

His articles have been republished and quoted in several blogs and whitepapers, including *eForensics* magazine. He is also the author of the book *Preventing Ransomware: Understand, Prevent, and Remediate Ransomware Attacks* (Packt Publishing, 2018).



Anoop Saldanha is one of the core authors of the Suricata Intrusion Detection and Prevention System, funded by the US Department of Homeland Security (DHS). He works as an independent security consultant and as a corporate security trainer. He designs and develops various detection technologies to secure both the host and the network, ranging from network security tools such as IDS/IPS to malware sandboxes, malware analysis tools, firewalls, endpoints, and IoT security tools. He holds multiple patents in the field of security and speaks at security conferences and workshops. He has previously worked in threat research

labs and detection engineering teams at RSA Security, Juniper Networks, Cyphort Cybersecurity, and various other cybersecurity startups.

About the Technical Reviewer

Ravikant Tiwari is a cybersecurity professional with in-depth knowledge of malware analysis and reverse engineering. He has more than nine years of experience in the antivirus industry. He has worked for cybersecurity firms such as Comodo Security Solutions, Norman ASA, McAfee, FireEye, and Acronis. He is a certified ethical hacker. His area of expertise includes malware analysis, reverse engineering, signature creation, and security research for designing and developing new features and solutions to enhance the detection capabilities of cybersecurity products. He has designed machine learning models for use in malware and exploit detection. He has been a member of the architect council at McAfee Labs, brainstorming and producing new solutions for McAfee. Currently leading the threat research lab, he is responsible for a multitude of tasks, including automation, producing malware detection rules, and developing a prototype for the Acronis Cyber Protect solution. He has written many blogs, articles, and threat reports. He is a speaker at RSA and Total Security conferences.

Occasionally, he provides expert comments and insights on security breaches and major hacks for media houses and newsrooms.

About the Foreword Author



Pedram Amini has spent much of his time in the shoes of a reverse engineer—developing automation tools and processes. In conjunction with his passion, he launched OpenRCE.org, a community website dedicated to the art and science of reverse engineering. He has presented at Black Hat, DEF CON, REcon, Ekoparty, BlueHat, ShmooCon, ToorCon, and Virus Bulletin, and taught numerous sold-out reverse engineering courses. He holds a computer science degree from Tulane University and is co-author of the book *Fuzzing: Brute Force Vulnerability Discovery*.

Pedram focuses the majority of his time on InQuest, whose product provides deep file inspection (DFI) for real-time threat detection and “retrohunting,” a novel approach that leverages the power of hindsight to apply today’s threat intelligence to yesterday’s data. Built by SOC analysts for SOC analysts, InQuest is designed to save enterprises their most limited resource, human cognition. He was formerly a director of software development at Avast after the acquisition of his startup, Jumpshot, a fully automated solution for the removal of deeply entrenched Windows malware infections. He is the founder of the Zero Day Initiative at TippingPoint (acquired by 3Com/HP). He has managed the world’s largest group of independent researchers, and he served as the assistant director and one of the founding members of iDEFENSE Labs (acquired by Verisign).

Acknowledgments

Thanks to our technical reviewer, Ravikant Tiwari, for his time and expertise reviewing more than 800 pages of our book and the various exercises and examples, making sure we have our content accurate to the dot. We'd like to thank Brad from Malware Traffic Analysis (www.malwaretrafficanalysis.com) for permitting us to use his samples. We'd like to thank Hex-rays for providing us licensed versions of the famous IDA Pro tool which we have covered in this book. Special thanks to the authors of various cybersecurity-related tools, without which writing this book would be impossible.

We'd also like to thank everybody at Apress, including the copyediting staff working hard behind the scenes, for their effort in helping with this book and making sure it meets the highest standards. Special thanks to Divya Modi, Matthew Moodie, Laura Berendson, Celestin Suresh John, and Nikhil Karkal in helping us through the various stages of this book development.

Abhijit Mohanta: To my dear father, thanks for your encouragement, without which I would never have been confident enough to write down my ideas into a book.

To the love of my life—my dear wife, Shreeti, thank you for being patient with me all these months while I spent hours writing the book.

Anoop Saldanha: I'd like to thank my wife, Sonia Carrol, without whose love and patience it would have been impossible for me to write this book. Thanks again to my wife and my daughters, Sadhana and Suvidha, for putting up with my insane schedule while writing this book!

I'd also like to thank my dad, William, and my mom, Nayana, for easing my load and helping me manage my various everyday tasks to free my time up to write this book. Not to mention the often underrated guidance and wisdom they have given me throughout the years.

Foreword

This book is a beast! If you're looking to master the ever-widening field of malware analysis, look no further. This is the definitive guide for you.

Reverse engineering (or reversing) is a fascinating subject and one that I've always had a love affair with. Puzzle lovers and tinkerers alike will find appeal in the art of reversing. Talented practitioners can discover and exploit software vulnerabilities, dissect the intent behind a novel malware sample, and hack a toy like a Big Mouth Billy Bass to operate as an Amazon Echo.

When approached by newcomers looking for advice on how to get started with reversing, I generally recommend that they start with malware analysis. The software targets are smaller than enterprise software and, therefore, more digestible. While code volume is lower, malware can, and will, employ any number of tricks that add hurdles for the analyst. Overcoming these challenges will quickly improve your skillset, and there are fresh malware samples for one to play with daily.

Malware analysts are needed now more than ever. The volume of unique malware, similar to the general volume of Internet-transmitted data, is growing rapidly every year. When I first got into the industry almost 20 years ago, there were hundreds to thousands of samples daily. Today, it's well into the millions. This increase in volume is of some benefit to defenders. Large volumes of data are a requisite for data science. There's tremendous value in machine learning, but it's no silver bullet. Manual analysis is still mandatory and will be for some time to come.

The stakes have never been higher. In 2010, Stuxnet was first discovered, and, to date, it's the most technically impressive piece of software I've ever seen. It is a modular and air-gap jumping worm, armed with four zero-day exploits and targeted toward Iranian nuclear enrichment centrifuges (reportedly ruining almost 20% of them). It is a clear sign of the military-industrial complex engaging on a new frontier. With today's large budgets and a shifting focus to digital, we can certainly expect some similarly sensational headlines in the future.

FOREWORD

Authors Abhijit and Anoop have done an incredible job putting together a truly all-encompassing work. I mean, wow, Chapter 16 is a book unto itself! I admire these two seasoned practitioners for making an effort to create such an incredible guide through such a wide field.

Another piece of advice I'm quick to share with folks looking to delve into reversing and malware analysis: you must truly be passionate and be willing to put in the time. To the reader: master the materials in this book, and you'll be ready to join the global resistance against malware.

—Pedram Amini

InQuest CTO and Founder of OpenRCE.org and Zero Day Initiative

Introduction

As cybersecurity specialists and corporate trainers, we are often contacted by people who say that their organization has been infected by malware, and they want to know what they should do to contain the infection, or they ask how they should secure their systems and network to prevent such attacks. The stories that we hear often follow the same storyline: *There was a malware infection, which our anti-malware product caught, we quarantined the system, cleaned it up, updated our IDS signatures, but now the infection is back, affecting our other systems and our staff.*

When we cross-question, some important questions are often left unanswered.

- Did you figure out the entry point of attack?
- Did you check for any infection spread (a.k.a. lateral movement/spread across systems from the malware infection point)?
- Did you make sure you were able to figure out all the artifacts from the malware infection and cleaned all of them up?
- Were you able to understand the intention of the cyberattack and the threat actor behind the cyberattack?
- Did you inform your management and give them a full report of the true damage caused by the malware infection?

In most cases, the answers to these questions are not ascertained, leaving holes in the SOC, IR, and forensic stages, which can lead to the infection remaining present in your network. Not knowing the intentions behind the attack and the attacker means that IT and SOC teams are not fully aware of the true impact of the infection, leaving management in a plight to build a plan to prepare for the potential damage to their business and brand because of this infection.

This is exactly where *Malware Analysis and Detection Engineering* comes in. It does not only help you learn how to detect, analyze, and reverse engineer malware, but it also teaches you the importance of effective and efficient workflows.

This book was described by Pedram Amini, the founder of Zero Day Initiative and OpenRCE, as a “beast!”. And a beast it is indeed, with more than 900 comprehensive pages of content and exercises. With this book at your fingertips, you should be able to take on any malware that comes your way.

Malware Analysis and Reverse Engineering

Pretty much any cyberattack involves malware, and the number of such attacks is increasing every day, and attackers are getting bolder as well. Millions of pieces of malware are seen every day, but there aren’t enough analysts out there to deal with it all. Malware analysis is an esoteric field that is mastered by only a few. It involves dissecting all types of malware efficiently and masterfully, with minimum expenditure of time and effort, high accuracy, and absolute inference of the malware’s intentions. Today, there are various analysts out there, but not all of them have the requisite skill to dissect a piece of malware.

This book incorporates our combined multiyear experiences in the field of cybersecurity. It translates myriad questions and cases and converts them into efficient and understandable material, which should help any analyst learn how to analyze malware systematically by using various unspoken tricks used by industry researchers. The samples in this book largely focus on Windows executables, but we also cover how to analyze and reverse other types of malware, including Microsoft Office macro malware, PowerShell and JavaScript malware, and other scripting malware.

We also introduce in this book a new, open source tool—APIMiner, which we developed while writing this book. It should be a gamechanger for malware analysts and reverse engineers around the world, which should greatly increase the speed and accuracy with which you can analyze malware.

But malware analysis may not be enough for most cases, and we understand this well based on our experience. And this is why we dedicate a section of this on the esoteric topic of reverse engineering. In Chapter 16, which deserves to be a book on its own, we introduce you to the world of x86 assembly and debuggers. We walk you through various tricks to quickly reverse and debug malware. We don’t treat reversing as a standalone topic, but instead, teach you how to combine various tools and tricks from malware analysis to make reverse engineering easier.

Detection Engineering: The Lonely Stepchild

The first thing we discussed when we devised the content for this book was, why hasn't anyone covered how to detect malware? The first part of dealing with any malware infection is to detect the malware infection itself. Then comes analyzing and reverse engineering samples. In our experience with various cybersecurity companies, we have seen that there is a huge gap between detection engineers and malware researchers, which in the end translates to poor detection products. But if you combine the knowledge from these two areas, you will have the skill set to apply the tricks from malware analysis to detect malware samples. At the same time, detection engineering uses various automation and development tools, which, if used effectively, can help malware analysts speedily analyze and reverse malware samples.

To that end, we dedicate Part 6 of this book to detection engineering, taking you through the internals of the most important cybersecurity tools used in this industry: antiviruses, malware sandboxes, network intrusion detection and prevention systems, and binary instrumentation. By covering various detection tools and frameworks, which range from host-based anti-malware tools like antiviruses and binary instrumentation frameworks, to network security tools like IDS/IPS and Suricata, we teach you how to apply the intricate workings of these detection tools to automate your everyday analysis and reversing workflow.

Hands-on

Ever seen kids take homework home and come back to school the next day with their work completed? That's the exact story of labs at the end of each chapter. And this is precisely why we don't use labs at the end of the chapter and instead incorporate the lab exercises as hands-on exercises in the chapters. You run and inspect examples under our supervision to make sure that you understand every aspect of what you might encounter.

The trouble with real-world malware exercises is that they rush you and place you in a state of panic when you are learning how to analyze them—because malware waits for no one. Our exercises are samples that were developed in-house and exhibit malware behavior under controlled conditions to let you analyze them at your own pace. At the same time, to prepare you for the real world, we have a ton of hands-on, real-world malware exercises throughout the book, allowing you to test the tricks you learned from the simulated samples against real-world samples.