The Antivirus Hacker’s Handbook
The Antivirus Hacker’s Handbook

Joxean Koret
Elias Bachaalany

WILEY
Joxean Koret has been working for the past +15 years in many different computing areas. He started as a database software developer and DBA, working with a number of different RDBMSs. Afterward he got interested in reverse-engineering and applied this knowledge to the DBs he was working with. He has discovered dozens of vulnerabilities in products from the major database vendors, especially in Oracle software. He also worked in other security areas, such as developing IDA Pro at Hex-Rays or doing malware analysis and anti-malware software development for an antivirus company, knowledge that was applied afterward to reverse-engineer and break over 14 AV products in roughly one year. He is currently a security researcher in Coseinc.

Elias Bachaalany has been a computer programmer, a reverse-engineer, an occasional reverse-engineering trainer, and a technical writer for the past 14 years. Elias has also co-authored the book *Practical Reverse Engineering*, published by Wiley (ISBN: 978-111-8-78731-1). He has worked with various technologies and programming languages including writing scripts, doing web development, working with database design and programming, writing Windows device drivers and low-level code such as boot loaders or minimal operating systems, writing managed code, assessing software protections, and writing reverse-engineering and desktop security tools. Elias has also presented twice at REcon Montreal (2012 and 2013).

While working for Hex-Rays SA in Belgium, Elias helped improve and add new features to IDA Pro. During that period, he authored various technical blog posts, provided IDA Pro training, developed various debugger plug-ins, amped up IDA Pro’s scripting facilities, and contributed to the IDAPython project. Elias currently works at Microsoft.
Credits

Project Editor
Sydney Argenta

Technical Editor
Daniel Pistelli

Production Editor
Saleem Hameed Sulthan

Copy Editor
Marylouise Wiack

Manager of Content Development & Assembly
Mary Beth Wakefield

Production Manager
Kathleen Wisor

Marketing Director
David Mayhew

Marketing Manager
Carrie Sherrill

Professional Technology & Strategy Director
Barry Pruett

Business Manager
Amy Knies

Associate Publisher
Jim Minatel

Project Coordinator, Cover
Brent Savage

Proofreader
Nicole Hirschman

Indexer
Nancy Guenther

Cover Designer
Wiley

Cover Image
Wiley; Shield © iStock.com/DSGpro
Acknowledgments

I would like to acknowledge Mario Ballano, Ruben Santamarta, and Victor Manual Alvarez, as well as all my friends who helped me write this book, shared their opinions and criticisms, and discussed ideas. I am most thankful to my girlfriend for her understanding and support during the time that I spent on this book. Many thanks to Elias Bachaalany; without his help, this book would not have been possible. Also, special thanks to everyone at Wiley; it has been a great pleasure to work with you on this book. I am grateful for the help and support of Daniel Pistelli, Carol Long, Sydney Argenta, Nicole Hirschman, and Marylouise Wiack.
## Contents at a Glance

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introduction</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part I</td>
<td>Antivirus Basics</td>
<td>1</td>
</tr>
<tr>
<td>Chapter 1</td>
<td>Introduction to Antivirus Software</td>
<td>3</td>
</tr>
<tr>
<td>Chapter 2</td>
<td>Reverse-Engineering the Core</td>
<td>15</td>
</tr>
<tr>
<td>Chapter 3</td>
<td>The Plug-ins System</td>
<td>57</td>
</tr>
<tr>
<td>Chapter 4</td>
<td>Understanding Antivirus Signatures</td>
<td>77</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>The Update System</td>
<td>87</td>
</tr>
<tr>
<td>Part II</td>
<td>Antivirus Software Evasion</td>
<td>103</td>
</tr>
<tr>
<td>Chapter 6</td>
<td>Antivirus Software Evasion</td>
<td>105</td>
</tr>
<tr>
<td>Chapter 7</td>
<td>Evading Signatures</td>
<td>117</td>
</tr>
<tr>
<td>Chapter 8</td>
<td>Evading Scanners</td>
<td>133</td>
</tr>
<tr>
<td>Chapter 9</td>
<td>Evading Heuristic Engines</td>
<td>165</td>
</tr>
<tr>
<td>Chapter 10</td>
<td>Identifying the Attack Surface</td>
<td>183</td>
</tr>
<tr>
<td>Chapter 11</td>
<td>Denial of Service</td>
<td>207</td>
</tr>
<tr>
<td>Part III</td>
<td>Analysis and Exploitation</td>
<td></td>
</tr>
<tr>
<td>Chapter 12</td>
<td>Static Analysis</td>
<td>217</td>
</tr>
<tr>
<td>Chapter 13</td>
<td>Dynamic Analysis</td>
<td>235</td>
</tr>
<tr>
<td>Chapter 14</td>
<td>Local Exploitation</td>
<td>269</td>
</tr>
<tr>
<td>Chapter 15</td>
<td>Remote Exploitation</td>
<td>297</td>
</tr>
</tbody>
</table>
### Contents at a Glance

<table>
<thead>
<tr>
<th>Part</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part IV</td>
<td>Current Trends and Recommendations</td>
<td>321</td>
</tr>
<tr>
<td>Chapter 16</td>
<td>Current Trends in Antivirus Protection</td>
<td>323</td>
</tr>
<tr>
<td>Chapter 17</td>
<td>Recommendations and the Possible Future</td>
<td>331</td>
</tr>
<tr>
<td>Index</td>
<td></td>
<td>347</td>
</tr>
</tbody>
</table>
# Contents

## Introduction

<table>
<thead>
<tr>
<th>Part I</th>
<th>Antivirus Basics</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1</td>
<td>Introduction to Antivirus Software</td>
<td>3</td>
</tr>
<tr>
<td>What Is Antivirus Software?</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Antivirus Software: Past and Present</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Antivirus Scanners, Kernels, and Products</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Typical Misconceptions about Antivirus Software</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Antivirus Features</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Basic Features</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Making Use of Native Languages</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Scanners</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Signatures</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Compressors and Archives</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Unpackers</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Emulators</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Miscellaneous File Formats</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Advanced Features</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Packet Filters and Firewalls</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Self-Protection</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Anti-Exploiting</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>Chapter 2</td>
<td>Reverse-Engineering the Core</td>
<td>15</td>
</tr>
<tr>
<td>Reverse-Engineering Tools</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Command-Line Tools versus GUI Tools</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Debugging Symbols</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>Tricks for Retrieving Debugging Symbols</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>Debugging Tricks</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>
## Backdoors and Configuration Settings 21
- Kernel Debugging 23
- Debugging User-Mode Processes with a Kernel-Mode Debugger 25
- Analyzing AV Software with Command-Line Tools 27

## Porting the Core 28
- A Practical Example: Writing Basic Python Bindings for Avast for Linux 29
- A Brief Look at Avast for Linux 29
- Writing Simple Python Bindings for Avast for Linux 32
- The Final Version of the Python Bindings 37

## A Practical Example: Writing Native C/C++ Tools for Comodo Antivirus for Linux 37

## Other Components Loaded by the Kernel 55
- Summary 56

### Chapter 3 The Plug-ins System 57
- Understanding How Plug-ins Are Loaded 58
  - A Full-Featured Linker in Antivirus Software 58
  - Understanding Dynamic Loading 59
  - Advantages and Disadvantages of the Approaches for Packaging Plug-ins 60
- Types of Plug-ins 62
  - Scanners and Generic Routines 63
  - File Format and Protocol Support 64
  - Heuristics 65
    - Bayesian Networks 66
    - Bloom Filters 67
    - Weights-Based Heuristics 68
- Some Advanced Plug-ins 69
  - Memory Scanners 69
  - Non-native Code 70
  - Scripting Languages 72
  - Emulators 73
- Summary 74

### Chapter 4 Understanding Antivirus Signatures 77
- Typical Signatures 77
  - Byte-Streams 78
  - Checksums 78
  - Custom Checksums 79
  - Cryptographic Hashes 80
- Advanced Signatures 80
  - Fuzzy Hashing 81
  - Graph-Based Hashes for Executable Files 83
- Summary 85
# Contents

Chapter 5 **The Update System**  
- Understanding the Update Protocols 87  
- Support for SSL/TLS 88  
- Verifying the Update Files 91  
- Dissecting an Update Protocol 92  
- When Protection Is Done Wrong 100  
- Summary 101  

Part II **Antivirus Software Evasion**  
Chapter 6 **Antivirus Software Evasion**  
- Who Uses Antivirus Evasion Techniques? 106  
- Discovering Where and How Malware Is Detected 107  
- Old Tricks for Determining Where Malware Is Detected: Divide and Conquer 107  
- Evading a Simple Signature-Based Detection with the Divide and Conquer Trick 108  
- Binary Instrumentation and Taint Analysis 113  
- Summary 114  

Chapter 7 **Evading Signatures**  
- File Formats: Corner Cases and Undocumented Cases 118  
- Evading a Real Signature 118  
- Evasion Tips and Tricks for Specific File Formats 124  
  - PE Files 124  
  - JavaScript 126  
    - String Encoding 127  
    - Executing Code on the Fly 128  
  - Hiding the Logic: Opaque Predicates and Junk Code 128  
  - PDF 129  
- Summary 131  

Chapter 8 **Evading Scanners**  
- Generic Evasion Tips and Tricks 133  
  - Fingerprinting Emulators 134  
  - Advanced Evasion Tricks 136  
    - Taking Advantage of File Format Weaknesses 136  
    - Using Anti-emulation Techniques 137  
    - Using Anti-disassembling Techniques 142  
- Disrupting Code Analyzers through Anti-analysis 144  
- More Anti-Anti-Anti… 147  
- Causing File Format Confusion 148  
- Automating Evasion of Scanners 148  
  - Initial Steps 149  
    - Installing ClamAV 150  
    - Installing Avast 150  
    - Installing AVG 151
## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Installing F-Prot</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>Installing Comodo</td>
<td>153</td>
</tr>
<tr>
<td></td>
<td>Installing Zoner Antivirus</td>
<td>154</td>
</tr>
<tr>
<td></td>
<td>MultiAV Configuration</td>
<td>154</td>
</tr>
<tr>
<td></td>
<td>peCloak</td>
<td>158</td>
</tr>
<tr>
<td></td>
<td>Writing the Final Tool</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>Summary</td>
<td>162</td>
</tr>
<tr>
<td></td>
<td><strong>Chapter 9</strong> Evading Heuristic Engines</td>
<td>165</td>
</tr>
<tr>
<td></td>
<td>Heuristic Engine Types</td>
<td>165</td>
</tr>
<tr>
<td></td>
<td>Static Heuristic Engines</td>
<td>166</td>
</tr>
<tr>
<td></td>
<td>Bypassing a Simplistic Static Heuristic Engine</td>
<td>166</td>
</tr>
<tr>
<td></td>
<td>Dynamic Heuristic Engines</td>
<td>173</td>
</tr>
<tr>
<td></td>
<td>Userland Hooks</td>
<td>173</td>
</tr>
<tr>
<td></td>
<td>Bypassing a Userland HIPS</td>
<td>176</td>
</tr>
<tr>
<td></td>
<td>Kernel-Land Hooks</td>
<td>178</td>
</tr>
<tr>
<td></td>
<td>Summary</td>
<td>180</td>
</tr>
<tr>
<td></td>
<td><strong>Chapter 10</strong> Identifying the Attack Surface</td>
<td>183</td>
</tr>
<tr>
<td></td>
<td>Understanding the Local Attack Surface</td>
<td>185</td>
</tr>
<tr>
<td></td>
<td>Finding Weaknesses in File and Directory Privileges</td>
<td>185</td>
</tr>
<tr>
<td></td>
<td>Escalation of Privileges</td>
<td>186</td>
</tr>
<tr>
<td></td>
<td>Incorrect Privileges in Files and Folders</td>
<td>186</td>
</tr>
<tr>
<td></td>
<td>Incorrect Access Control Lists</td>
<td>187</td>
</tr>
<tr>
<td></td>
<td>Kernel-Level Vulnerabilities</td>
<td>187</td>
</tr>
<tr>
<td></td>
<td>Exotic Bugs</td>
<td>188</td>
</tr>
<tr>
<td></td>
<td>Exploiting SUID and SGID Binaries on Unix-Based Platforms</td>
<td>189</td>
</tr>
<tr>
<td></td>
<td>ASLR and DEP Status for Programs and Binaries</td>
<td>190</td>
</tr>
<tr>
<td></td>
<td>Exploiting Incorrect Privileges on Windows Objects</td>
<td>193</td>
</tr>
<tr>
<td></td>
<td>Exploiting Logical Flaws</td>
<td>196</td>
</tr>
<tr>
<td></td>
<td>Understanding the Remote Attack Surface</td>
<td>197</td>
</tr>
<tr>
<td></td>
<td>File Parsers</td>
<td>198</td>
</tr>
<tr>
<td></td>
<td>Generic Detection and File Disinfection Code</td>
<td>199</td>
</tr>
<tr>
<td></td>
<td>Network Services, Administration Panels, and Consoles</td>
<td>199</td>
</tr>
<tr>
<td></td>
<td>Firewalls, Intrusion Detection Systems, and Their Parsers</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>Update Services</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>Browser Plug-ins</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>Security Enhanced Software</td>
<td>202</td>
</tr>
<tr>
<td></td>
<td>Summary</td>
<td>203</td>
</tr>
<tr>
<td></td>
<td><strong>Chapter 11</strong> Denial of Service</td>
<td>207</td>
</tr>
<tr>
<td></td>
<td>Local Denial-of-Service Attacks</td>
<td>208</td>
</tr>
<tr>
<td></td>
<td>Compression Bombs</td>
<td>208</td>
</tr>
<tr>
<td></td>
<td>Creating a Simple Compression Bomb</td>
<td>209</td>
</tr>
<tr>
<td></td>
<td>Bugs in File Format Parsers</td>
<td>212</td>
</tr>
<tr>
<td></td>
<td>Attacks against Kernel Drivers</td>
<td>213</td>
</tr>
<tr>
<td></td>
<td>Remote Denial-of-Service Attacks</td>
<td>214</td>
</tr>
<tr>
<td></td>
<td>Compression Bombs</td>
<td>214</td>
</tr>
<tr>
<td></td>
<td>Bugs in File Format Parsers</td>
<td>215</td>
</tr>
<tr>
<td></td>
<td>Summary</td>
<td>215</td>
</tr>
</tbody>
</table>
### Part III  Analysis and Exploitation 217

#### Chapter 12  Static Analysis 219

- Performing a Manual Binary Audit 219
  - File Format Parsers 220
  - Remote Services 228
- Summary 233

#### Chapter 13  Dynamic Analysis 235

- Fuzzing 235
  - What Is a Fuzzer? 236
  - Simple Fuzzing 237
  - Automating Fuzzing of Antivirus Products 239
    - Using Command-Line Tools 240
    - Porting Antivirus Kernels to Unix 243
    - Fuzzing with Wine 244
    - Problems, Problems, and More Problems 247
  - Finding Good Templates 248
  - Finding Template Files 250
  - Maximizing Code Coverage 252
    - Blind Code Coverage Fuzzer 253
  - Finding Template Files 250
  - Maximizing Code Coverage 252
    - Blind Code Coverage Fuzzer 253
  - Using Blind Code Coverage Fuzzer 254
  - Nightmare, the Fuzzing Suite 259
    - Configuring Nightmare 260
    - Finding Samples 262
    - Configuring and Running the Fuzzer 262
- Summary 266

#### Chapter 14  Local Exploitation 269

- Exploiting Backdoors and Hidden Features 270
- Finding Invalid Privileges, Permissions, and ACLs 274
- Searching Kernel-Land for Hidden Features 279
- More Logical Kernel Vulnerabilities 285
- Summary 295

#### Chapter 15  Remote Exploitation 297

- Implementing Client-Side Exploitation 297
  - Exploiting Weakness in Sandboxing 297
  - Exploiting ASLR, DEP, and RWX Pages at Fixed Addresses 298
- Writing Complex Payloads 300
  - Taking Advantage of Emulators 301
  - Exploiting Archive Files 302
  - Finding Weaknesses in Intel x86, AMD x86_64, and ARM Emulators 303
  - Using JavaScript, VBScript, or ActionScript 303
  - Determining What an Antivirus Supports 304
  - Launching the Final Payload 306
- Exploiting the Update Services 307
  - Writing an Exploit for an Update Service 308
- Server-Side Exploitation 317
Differences between Client-Side and Server-Side Exploitation 317
Exploiting ASLR, DEP, and RWX Pages at Fixed Addresses 318
Summary 318

Part IV Current Trends and Recommendations 321

Chapter 16 Current Trends in Antivirus Protection 323
Matching the Attack Technique with the Target 324
The Diversity of Antivirus Products 324
Zero-Day Bugs 324
Patched Bugs 325
Targeting Home Users 325
Targeting Small to Medium-Sized Companies 326
Targeting Governments and Big Companies 326
The Targets of Governments 327
Summary 328

Chapter 17 Recommendations and the Possible Future 331
Recommendations for Users of Antivirus Products 331
Blind Trust Is a Mistake 332
Isolating Machines Improves Protection 337
Auditing Security Products 338
Recommendations for Antivirus Vendors 338
Engineering Is Different from Security 339
Exploiting Antivirus Software Is Trivial 339
Perform Audits 340
Fuzzing 340
Use Privileges Safely 341
Reduce Dangerous Code in Parsers 342
Improve the Safety of Update Services and Protocols 342
Remove or Disable Old Code 343
Summary 344

Index 347
Welcome to The Antivirus Hacker’s Handbook. With this book, you can increase your knowledge about antivirus products and reverse-engineering in general; while the reverse-engineering techniques and tools discussed in this book are applied to antivirus software, they can also be used with any other software products. Security researchers, penetration testers, and other information security professionals can benefit from this book. Antivirus developers will benefit as well because they will learn more about how antivirus products are analyzed, how they can be broken into parts, and how to prevent it from being broken or make it harder to break.

I want to stress that although this book is, naturally, focused on antivirus products, it also contains practical examples that show how to apply reverse-engineering, vulnerability discovery, and exploitation techniques to real-world applications.

Overview of the Book and Technology

This book is designed for individuals who need to better understand the functionality of antivirus products, regardless of which side of the fence they are on: offensive or defensive. Its objective is to help you learn when and how specific techniques and tools should be used and what specific parts of antivirus products you should focus on, based on the specific tasks you want to accomplish. This book is for you if any of the following statements are true:

- You want to learn more about the security of antivirus products.
- You want to learn more about reverse-engineering, perhaps with the aim of reverse-engineering antivirus products.
- You want to bypass antivirus software.
- You want to break antivirus software into pieces.
You want to write exploits for antivirus software.

You want to evaluate antivirus products.

You want to increase the overall security of your own antivirus products, or you want to know how to write security-aware code that will deal with hostile code.

You love to tinker with code, or you want to expand your skills and knowledge in the information security field.

How This Book Is Organized

The contents of this book are structured as follows:

- **Chapter 1, “Introduction to Antivirus Software”**—Guides you through the history of antivirus software to the present, and discusses the most typical features available in antivirus products, as well as some less common ones.

- **Chapter 2, “Reverse-Engineering the Core”**—Describes how to reverse-engineer antivirus software, with tricks that can be used to debug the software or disable its self-protection mechanisms. This chapter also discusses how to apply this knowledge to create Python bindings for Avast for Linux, as well as a native C/C++ tool and unofficial SDK for the Comodo for Linux antivirus.

- **Chapter 3, “The Plug-ins System”**—Discusses how antivirus products use plug-ins, how they are loaded, and how they are distributed, as well as the purpose of antivirus plug-ins.

- **Chapter 4, “Understanding Antivirus Signatures”**—Explores the most typical signature types used in antivirus products, as well as some more advanced ones.

- **Chapter 5, “The Update System”**—Describes how antivirus software is updated, how the update systems are developed, and how update protocols work. This chapter concludes by showing a practical example of how to reverse-engineer an easy update protocol.

- **Chapter 6, “Antivirus Software Evasion”**—Gives a basic overview of how to bypass antivirus software, so that files can evade detection. Some general tricks are discussed, as well as techniques that should be avoided.

- **Chapter 7, “Evading Signatures”**—Continues where Chapter 4 left off and explores how to bypass various kinds of signatures.

- **Chapter 8, “Evading Scanners”**—Continues the discussion of how to bypass antivirus products, this time focusing on scanners. This chapter looks at how to bypass some static heuristic engines, anti-disassembling, anti-emulation, and other “anti-” tricks, as well as how to write an automatic tool for portable executable file format evasion of antivirus scanners.
Chapter 9, “Evading Heuristic Engines”—Finishes the discussion on evasion by showing how to bypass both static and dynamic heuristic engines implemented by antivirus products.

Chapter 10, “Identifying the Attack Surface”—Introduces techniques used to attack antivirus products. This chapter will guide you through the process of identifying both the local and remote attack surfaces exposed by antivirus software.

Chapter 11, “Denial of Service”—Starts with a discussion about performing denial-of-service attacks against antivirus software. This chapter discusses how such attacks can be launched against antivirus products both locally and remotely by exploiting their vulnerabilities and weaknesses.

Chapter 12, “Static Analysis”—Guides you through the process of statically auditing antivirus software to discover vulnerabilities, including real-world vulnerabilities.

Chapter 13, “Dynamic Analysis”—Continues with the discussion of finding vulnerabilities in antivirus products, but this time using dynamic analysis techniques. This chapter looks specifically at fuzzing, the most popular technique used to discover vulnerabilities today. Throughout this chapter, you will learn how to set up a distributed fuzzer with central administration to automatically discover bugs in antivirus products and be able to analyze them.

Chapter 14, “Local Exploitation”—Guides you through the process of exploiting local vulnerabilities while putting special emphasis on logical flaws, backdoors, and unexpected usages of kernel-exposed functionality.

Chapter 15, “Remote Exploitation”—Discusses how to write exploits for memory corruption issues by taking advantage of typical mistakes in antivirus products. This chapter also shows how to target update services and shows a full exploit for one update service protocol.

Chapter 16, “Current Trends in Antivirus Protection”—Discusses which antivirus product users can be targeted by actors that use flaws in antivirus software, and which users are unlikely to be targeted with such techniques. This chapter also briefly discusses the dark world in which such bugs are developed.

Chapter 17, “Recommendations and the Possible Future”—Concludes this book by making some recommendations to both antivirus users and antivirus vendors, and discusses which strategies can be adopted in the future by antivirus products.

Who Should Read This Book

This book is designed for individual developers and reverse-engineers with intermediate skills, although the seasoned reverse-engineer will also benefit
Introduction

from the techniques discussed here. If you are an antivirus engineer or a malware reverse-engineer, this book will help you to understand how attackers will try to exploit your software. It will also describe how to avoid undesirable situations, such as exploits for your antivirus product being used in targeted attacks against the users you are supposed to protect.

More advanced individuals can use specific chapters to gain additional skills and knowledge. As an example, if you want to learn more about writing local or remote exploits for antivirus products, proceed to Part III, “Analysis and Exploitation,” where you will be guided through almost the entire process of discovering an attack surface, finding vulnerabilities, and exploiting them. If you are interested in antivirus evasion, then Part II, “Antivirus Software Evasion,” is for you. So, whereas some readers may want to read the book from start to finish, there is nothing to prevent you from moving around as needed.

Tools You Will Need

Your desire to learn is the most important thing you have as you start to read this book. Although I try to use open-source “free” software, this is not always possible. For example, I used the commercial tool IDA in a lot of cases; because antivirus programs are, with only one exception, closed-source commercial products, you need to use a reverse-engineering tool, and IDA is the de facto one. Other tools that you will need include compilers, interpreters (such as Python), and some tools that are not open source but that can be freely downloaded, such as the Sysinternals tools.

What’s on the Wiley Website

To make it as easy as possible for you to get started, some of the basic tools you will need are available on the Wiley website, which has been set up for this book at www.wiley.com/go/antivirushackershandbook.

Summary (From Here, Up Next, and So On)

*The Antivirus Hacker’s Handbook* is designed to help readers become aware of what antivirus products are, what they are not, and what to expect from them; this information is not usually available to the public. Rather than discussing how antivirus products work in general, it shows real bugs, exploits, and techniques for real-world products that you may be using right now and provides real-world techniques for evasion, vulnerability discovery, and exploitation. Learning how to break antivirus software not only helps attackers but also helps you to understand how antivirus products can be enhanced and how antivirus users can best protect themselves.
Part I

Antivirus Basics

In This Part

Chapter 1: Introduction to Antivirus Software
Chapter 2: Reverse-Engineering the Core
Chapter 3: The Plug-ins System
Chapter 4: Understanding Antivirus Signatures
Chapter 5: The Update System
Antivirus software is designed to prevent computer infections by detecting malicious software, commonly called malware, on your computer and, when appropriate, removing the malware and disinfecting the computer. Malware, also referred to as samples in this book, can be classified into various kinds, namely, Trojans, viruses (infectors), rootkits, droppers, worms, and so on.

This chapter covers what antivirus (AV) software is and how it works. It offers a brief history of AV software and a short analysis of how it evolved over time.

What Is Antivirus Software?

Antivirus software is special security software that aims to give better protection than that offered by the underlying operating system (such as Windows or Mac OS X). In most cases, it is used as a preventive solution. However, when that fails, the AV software is used to disinfect the infected programs or to completely clean malicious software from the operating system.

AV software uses various techniques to identify malicious software, which often self-protects and hides deep in an operating system. Advanced malware may use undocumented operating system functionality and obscure techniques in order to persist and avoid being detected. Because of the large attack surface these days, AV software is designed to deal with all kinds of malicious payloads coming from both trusted and untrusted sources. Some malicious inputs that
AV software tries to protect an operating system from, with varying degrees of success, are network packets, email attachments, and exploits for browsers and document readers, as well as executable programs running on the operating system.

**Antivirus Software: Past and Present**

The earliest AV products were simply called *scanners* because they were command-line scanners that tried to identify malicious patterns in executable programs. AV software has changed a lot since then. For example, many AV products no longer include command-line scanners. Most AV products now use graphical user interface (GUI) scanners that check every single file that is created, modified, or accessed by the operating system or by user programs. They also install firewalls to detect malicious software that uses the network to infect computers, install browser add-ons to detect web-based exploits, isolate browsers for safe payment, create kernel drivers for AV self-protection or sandboxing, and so on.

Since the old days of Microsoft DOS and other antiquated operating systems, software products have evolved alongside the operating systems, as is natural. However, AV software has evolved at a remarkable rate since the old days because of the incredible amount of malware that has been created. During the 1990s, an AV company would receive only a handful of malware programs in the space of a week, and these were typically file infectors (or viruses). Now, an AV company will receive thousands of unique malicious files (unique considering their cryptographic hash, like MD5 or SHA-1) daily. This has forced the AV industry to focus on automatic detection and on creating *heuristics* for detection of as-yet-unknown malicious software by both dynamic and static means. Chapters 3 and 4 discuss how AV software works in more depth.

The rapid evolution of malware and anti-malware software products is driven by a very simple motivator: money. In the early days, virus creators (also called *vXers*) used to write a special kind of file infector that focused on performing functions not previously done by others in order to gain recognition or just as a personal challenge. Today, malware development is a highly profitable business used to extort money from computer users, as well as steal their credentials for various online services such as eBay, Amazon, and Google Mail, as well as banks and payment platforms (PayPal, for example); the common goal is to make as much money as possible.

Some players in the malware industry can steal email credentials for your Yahoo or Gmail accounts and use them to send spam or malicious software to thousands of users in your name. They can also use your stolen credit card information to issue payments to other bank accounts controlled by them or to pay *mules* to move the stolen money from dirty bank accounts to clean ones, so their criminal activity becomes harder to trace.
Another increasingly common type of malware is created by governments, shady organizations, or companies that sell malware (spying software) to governments, who in turn spy on their own people’s communications. Some software is designed to sabotage foreign countries’ infrastructures. For example, the notorious Stuxnet computer worm managed to sabotage Iran’s Natanz nuclear plant, using up to five zero-day exploits. Another example of sabotage is between countries and companies that are in direct competition with another company or country or countries, such as the cyberattack on Saudi Aramco, a sabotage campaign attributed to Iran that targeted the biggest oil company in Saudi Arabia.

Software can also be created simply to spy on government networks, corporations, or citizens; organizations like the National Security Agency (NSA) and Britain’s Government Communications Headquarters (GCHQ), as well as hackers from the Palestine Liberation Army (PLA), engage in these activities almost daily. Two examples of surveillance software are FinFisher and Hacking Team. Governments, as well as law enforcement and security agencies, have purchased commercial versions of FinFisher and Hacking Team to spy on criminals, suspects, and their own citizens. An example that comes to mind is the Bahrain government, which used FinFisher software to spy on rebels who were fighting against the government.

Big improvements and the large amounts of money invested in malware development have forced the AV industry to change and evolve dramatically over the last ten years. Unfortunately, the defensive side of information security, where AV software lies, is always behind the offensive side. Typically, an AV company cannot detect malware that is as yet unknown, especially if there is some quality assurance during the development of the malware software piece. The reason is very simple: AV evasion is a key part of malware development, and for attackers it is important that their malware stay undetected as long as possible. Many commercial malware packages, both legal and illegal, are sold with a window of support time. During that support period, the malware product is updated so it bypasses detection by AV software or by the operating system. Alternatively, malware may be updated to address and fix bugs, add new features, and so on. AV software can be the target of an attack, as in the case of The Mask, which was government-sponsored malware that used one of Kaspersky’s zero-day exploits.

**Antivirus Scanners, Kernels, and Products**

A typical computer user may view the AV software as a simple software suite, but an attacker must be able to view the AV on a deeper level.

This chapter will detail the various components of an AV, namely, the kernel, command-line scanner, GUI scanner, daemons or system services, file system filter drivers, network filter drivers, and any other support utility that ships with it.
ClamAV, the only open-source AV software, is an example of a scanner. It simply performs file scanning to discover malicious software patterns, and it prints a message for each detected file. ClamAV does not disinfect or use a true (behavioral-based) heuristic system.

A kernel, on the other hand, forms the core of an AV product. For example, the core of ClamAV is the libclam.so library. All the routines for unpacking executable programs, compressors, cryptors, protectors, and so on are in this library. All the code for opening compressed files to iterate through all the streams in a PDF file or to enumerate and analyze the clusters in one OLE2 container file (such as a Microsoft Word document) are also in this library. The kernel is used by the scanner clamscan, by the resident (or daemon) clamd, or by other programs and libraries such as its Python bindings, which are called PyClamd.

**NOTE** AV products often use more than one AV core or kernel. For example, F-Secure uses its own AV engine and the engine licensed from BitDefender.

An antivirus product may not always offer third-party developers direct access to its core; instead, it may offer access to command-line scanners. Other AV products may not give access to command-line scanners; instead, they may only allow access to the GUI scanner or to a GUI program to configure how the real-time protection, or another part of the product, handles malware detection and disinfection. The AV product suite may also ship with other security programs, such as browsers, browser toolbars, drivers for self-protection, firewalls, and so on.

As you can see, the product is the whole software package the AV company ships to the customer, while the scanners are the tools used to scan files and directories, and the kernel includes the core features offered to higher-level software components such as the GUI or command-line scanners.

**Typical Misconceptions about Antivirus Software**

Most AV users believe that security products are bulletproof and that just installing AV software keeps their computers safe. This belief is not sound, and it is not uncommon to read comments in AV forums like, “I’m infected with XXX malware. How can it be? I have YYY AV product installed!”

To illustrate why AV software is not bulletproof, let’s take a look at the tasks performed by modern AV products:

- Discovering known malicious patterns and bad behaviors in programs
- Discovering known malicious patterns in documents and web pages
- Discovering known malicious patterns in network packets
- Trying to adapt and discover new bad behaviors or patterns based on experience with previously known ones