

Bruce Dang, Alexandre Gazet, and Elias Bachaalany
with contributions from Sébastien Josse

PRACTICAL REVERSE ENGINEERING

X86, X64, ARM, WINDOWS® KERNEL,
REVERSING TOOLS, AND OBFUSCATION

WILEY

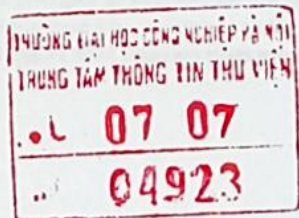
Practical Reverse Engineering

x86, x64, ARM, Windows® Kernel,
Reversing Tools, and Obfuscation


GIFT OF THE ASIA FOUNDATION
NOT FOR RE-SALE
QUÀ TẶNG CỦA QUỸ CHÂU Á
KHÔNG ĐƯỢC BÁN LẠI

Bruce Dang
Alexandre Gazet
Elias Bachaalany

with contributions from Sébastien Josse



WILEY

Practical Reverse Engineering: x86, x64, ARM, Windows® Kernel, Reversing Tools, and Obfuscation

Published by
John Wiley & Sons, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Copyright © 2014 by Bruce Dang
Published by John Wiley & Sons, Inc., Indianapolis, Indiana
Published simultaneously in Canada

ISBN: 978-1-118-78731-1
ISBN: 978-1-118-78725-0 (ebk)
ISBN: 978-1-118-78739-7 (ebk)

Manufactured in the United States of America

10 9 8 7 6

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

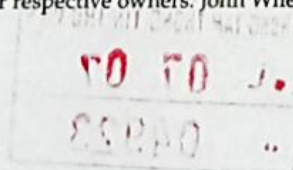
Limit of Liability/Disclaimer of Warranty: The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Web site is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information, the organization or website may provide or recommendations it may make. Further, readers should be aware that Internet websites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services please contact our Customer Care Department within the United States at (877) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at <http://booksupport.wiley.com>. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2013954099

Trademarks: Wiley and the Wiley logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.



About the Authors

This book is dedicated to those who relentlessly pursue knowledge and selflessly share it with others.

Robert Chappin is a senior software engineer at IBM, where he has been working for over 15 years. He has a master's degree in computer science from the University of Illinois at Urbana-Champaign. He is a frequent speaker at industry conferences and has published several papers on software engineering topics. He is also a member of the IEEE and the ACM.

Alexander Chavira is a senior software engineer at IBM, where he has been working for over 10 years. He has a master's degree in computer science from the University of Illinois at Urbana-Champaign. He is a frequent speaker at industry conferences and has published several papers on software engineering topics. He is also a member of the IEEE and the ACM.

Elia Bachvalov has been working at IBM for over 10 years. He has a master's degree in computer science from the University of Illinois at Urbana-Champaign. He is a frequent speaker at industry conferences and has published several papers on software engineering topics. He is also a member of the IEEE and the ACM.

While working for IBM, Robert Chappin and Alexander Chavira have been involved in the development of the IBM Pro scripting language. They have also been involved in the development of the IBM Pro security engine. They are currently working on the development of the IBM Pro security engine.

About the Authors

Bruce Dang is a senior security development engineering lead at Microsoft working on security technologies in unreleased Microsoft products. Previously, he worked on security vulnerabilities reported to Microsoft and was the first to publicly share analytical techniques for targeted attacks with Office documents. He and his team analyzed the famous Stuxnet malware, which supposedly attacked the Iranian uranium enrichment process. He has spoken at RSA, BlackHat Vegas, BlackHat Tokyo, Chaos Computer Club, REcon, and many other industry conferences.

Alexandre Gazet is a security researcher at Quarkslab. His interests focus on reverse engineering, software protections, and vulnerability research. Alexandre has presented at several conferences, including HITB Kuala Lumpur (2009) and REcon Montreal (2010 and 2011).

Elias Bachaalany has been a computer programmer, reverse engineer, freelance technical writer, and an occasional reverse engineering trainer for the past 14 years. Over his long career, Elias has worked with various technologies, including writing scripts, doing extensive web development, working with database design and programming, writing 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 to improve and add new features to IDA Pro. During that period, he authored various technical blog posts, provided IDA Pro training, developed various debugger plugins, amped up IDA Pro's scripting facilities, and contributed to the IDAPython project since version 1.2.0 and onwards. Elias currently works at Microsoft with a talented team of software security engineers.

Sébastien Josse is a security researcher at the French Ministry of Defense (Direction Générale de l'Armement). He has more than ten years of experience as an instructor, researcher, and consultant in the field of information systems security, in both the civilian and defense sectors. He dedicated his PhD dissertation (École Polytechnique, 2009) to the dynamic analysis of protected programs, focusing mainly on cryptographic mechanisms resistant to reverse engineering and on the use of a virtualization system to carry through analysis of protected programs. He has published in the journal *JICV* and several conferences proceedings, including ECRYPT (2004), EICAR (2006, 2008, 2011), AVAR (2007) and HICSS (2012, 2013 and 2014).



About the Technical Editor

Matt Miller is a principal security engineer in Microsoft's Trustworthy Computing organization, where he currently focuses on researching and developing exploit-mitigation technology. Prior to joining Microsoft, Matt was core developer for the Metasploit framework and a contributor to the journal *Uninformed*, where he wrote about topics related to exploitation, reverse engineering, program analysis, and operating system internals.

Production Editor

David Sullivan

Copy Editor

Leslie Smith

Editorial Manager

Mark Jeff Woodford

Executive Editorial Manager

Executive Editor

Associate Director of

Marketing

David Hargrave

Group Publisher

Richard Swadley

Associate Publisher

Jim Minetti

Project Coordinator, Press

Trish Alcorn

Proofreader

Just Chase, Wood, New York

Indexer

Ron Gorman

Cover Designer

Ryan Sneed



Credits

Executive Editor

Carol Long

Project Editor

John Sleeva

Technical Editor

Matt Miller

Production Editor

Daniel Scribner

Copy Editor

Luann Rouff

Editorial Manager

Mary Beth Wakefield

Freelancer Editorial Manager

Rosemarie Graham

**Associate Director of
Marketing**

David Mayhew

Marketing Manager

Ashley Zurcher

Business Manager

Amy Knies

**Vice President and Executive
Group Publisher**

Richard Swadley

Associate Publisher

Jim Minatel

Project Coordinator, Cover

Todd Klemme

Proofreader

Josh Chase, Word One New York

Indexer

Ron Strauss

Cover Designer

Ryan Sneed



Acknowledgments

Writing this book has been one of the most interesting and time-consuming endeavors we have ever gone through. The book represents something that we wish we had when we started learning about reverse engineering more than 15 years ago. At the time, there was a dearth of books and online resources (there were no blogs back then); we learned the art primarily through friends and independent trial-and-error experiments. The information security “industry” was also non-existent back then. Today, the world is different. We now have decompilers, web scanners, static source scanners, cloud (?), and APTs (unthinkable!). Numerous blogs, forums, books, and in-person classes aim to teach reverse engineering. These resources vary greatly in quality. Some are sub-standard but shamelessly published or offered to take advantage of the rise in demand for computer security; some are of extremely high quality but not well attended/read due to lack of advertising, specialization or because they are simply “too esoteric.” There is not a unifying resource that people can use as the foundation for learning reverse engineering. We hope this book is that foundation.

Now for the best part, acknowledging the people who helped us arrive at where we are today. All of the authors would like to acknowledge Rolf Rolles for his contributions to the obfuscation chapter. Rolf is a real pioneer in the field of reverse engineering. His seminal work on virtual machine deobfuscation, applying program analysis to reverse engineering, and binary analysis education influenced and inspired a new generation of reverse engineers. We hope that he will continue to contribute to the field and inspire others to do the same. Next, we would also like to thank Matt Miller, our comrade and technical reviewer. Matt is another true pioneer in our field and has made seminal contributions to exploit mitigations in Windows. His dedication to details and helping others learn should be a model for all. Finally, we would like to thank Carol Long, John Sleeva, Luann Rouff, and the staff at John Wiley & Sons for putting up with us through the publishing process.

— The authors

I would like to thank my parents for their sacrifices to give me better opportunities in life; my sister and brother, Ivy Dang and Donald Dang, for being a constant source of support and inspiration; and Rolf Rolles for being a good friend and source of reason all these years. I did not have many role models growing up, but the following people directly helped shape my perspectives: Le Thanh Sang, Vint Cerf, and Douglas Comer. At university, I learned the joy of Chinese literature from David Knetches, Buddhist studies from Kyoko Tokuno, Indian history from Richard Salomon (who would have thought so much can be learned from rocks and coins!), Central Asian history from Daniel Waugh, and Chinese language from Nyan-Ping Bi. While they are not reverse engineers, their enthusiasm and dedication forever inspired and made me a better human being and engineer. If I had met them earlier, my career path would probably be very different.

Through the journey of professional life, I was fortunate enough to meet intelligent people who influenced me (in no particular order): Alex Carp, rebel, Navin Pai, Jonathan Ness, Felix Domke, Karl J., Julien Tinnes, Josh Phillips, Daniel Radu, Maarten Boone, Yoann Guillot, Ivanle0u (thanks for hosting us), Richard van Eeden, Dan Ho, Andy Renk, Elia Florio, Ilfak Guilfanov, Matt Miller, David Probert, Damian Hasse, Matt Thomlinson, Shawn Hoffman, David Dittrich, Eloi Vanderbeken, LMH, Ali Rahbar, Fermin Serna, Otto Kivling, Damien Aumaitre, Tavis Ormandy, Ali Pezeshk, Gynvael Coldwind, anakata (a rare genius), Richard van Eeden, Noah W., Ken Johnson, Chengyun Yu, Elias Bachaalany, Felix von Leitner, Michal Chmielewski, sectorx, Son Pho Nguyen, Nicolas Pouvesle, Kostya Kortchinsky, Peter Viscerola, Torbjorn L., Gustavo di Scotti, Sergiusz Fonrobert, Peter W., Ilja van Sprundel, Brian Cavenah, upb, Maarten Van Horenbeeck, Robert Hensing, Cristian Craioveanu, Claes Nyberg, Igor Skorchinsky, John Lambert, Mark Wodrich (role model Buddhist), David Midturi, Gavin Thomas, Sebastian Porst, Peter Vel, Kevin Broas, Michael Sandy, Christer Oberg, Mateusz "j00ru" Jurczyk, David Ross, and Raphael Rigo. Jonathan Ness and Damian Hasse were always supportive of me doing things differently and constantly gave me opportunities to fail/succeed. If I forgot you, please forgive me.

The following people directly provided feedback and improved the initial drafts of my chapters: Michal Chmielewski, Shawn Hoffman, Nicolas Pouvesle, Matt Miller, Alex Ionescu, Mark Wodrich, Ben Byer, Felix Domke, Ange Albertini, Igor Skorchinsky, Peter Ferrie, Lien Duong, iZsh, Frank Boldewin, Michael Hale Ligh, Sebastien Renaud, Billy McCourt, Peter Viscerola, Dennis Elser, Thai Duong, Eloi Vanderbeken, Raphael Rigo, Peter Vel, and Bradley Spengler (a true over-achiever). Without their insightful comments and suggestions, most of the book would be unreadable. Of course, you can blame me for the remaining mistakes.

There are numerous other unnamed people that contributed to my knowledge and therefore this book.

I also want to thank Molly Reed and Tami Needham from The Omni Group for giving us a license of OmniGraffle to make illustrations in the earlier drafts.

Last but not least, I want to thank Alex, Elias, and Sébastien for helping me with this book. Without them, the book would have never seen the light of day.

— Bruce

First, I would like to thank Bruce Dang for inviting me to take part in this great project. It has been a long and enriching journey. Rolf Rolles was there at first, and I personally thank him for the countless hours we spent together imagining the obfuscation chapter and collecting material. Sébastien Josse then agreed to joined us; his contribution is invaluable and our chapter wouldn't be the same without him. Thank you, Seb.

I also want to thank my friends Fabrice Desclaux, Yoann Guillot, and Jean-Philippe Luyten for their invaluable feedback.

Finally, thanks to Carol Long for making this book possible, and to John Sleeva for keeping us on track.

— Alexandre

I want to start by thanking Bruce Dang, my friend and colleague, for giving me the chance to participate in this endeavor. I also want to thank all my friends and colleagues for their support and help. In particular, I would like to thank Daniel Pistelli (CEO of Cerbero GmbH), Michal Chmielewski, Swamy Shivaganga Nagaraju, and Alexandre Gazet for their technical input and feedback during the writing of the book.

I want to thank Mr. Ilfak Guilfanov (CEO of Hex-Rays SA). I learned a lot from him while working at Hex-Rays. His hard work, patience, and perseverance to create IDA Pro will always be an inspiration to me.

A big thanks to John Wiley & Sons for giving us the opportunity to publish this book. Thanks also to the acquisition editor Carol Long for her prompt and professional assistance, and to the project editor John Sleeva and copy editor Luann Rouff for their energy, patience, and hard work.

— Elias

I want to thank Alexandre, Elias, and Bruce for giving me the opportunity to contribute to this book. I also want to thank Jean-Philippe Luyten for putting us in touch. Finally, thanks to Carol Long and John Sleeva for their help and professionalism in the realization of this project.

— Sébastien



Contents at a Glance

Introduction	xxiii
Chapter 1 x86 and x64	1
Chapter 2 ARM	39
Chapter 3 The Windows Kernel	87
Chapter 4 Debugging and Automation	187
Chapter 5 Obfuscation	267
Appendix Sample Names and Corresponding SHA1 Hashes	341
Index	343

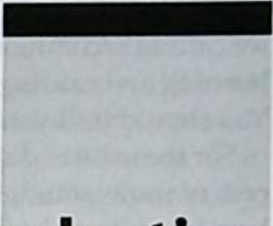
Contents

Introduction	xxiii
Chapter 1 x86 and x64	1
Register Set and Data Types	2
Instruction Set	3
Syntax	4
Data Movement	5
Exercise	11
Arithmetic Operations	11
Stack Operations and Function Invocation	13
Exercises	17
Control Flow	17
System Mechanism	25
Address Translation	26
Interrupts and Exceptions	27
Walk-Through	28
Exercises	35
x64	36
Register Set and Data Types	36
Data Movement	36
Canonical Address	37
Function Invocation	37
Exercises	38
Chapter 2 ARM	39
Basic Features	40
Data Types and Registers	43
System-Level Controls and Settings	45
Introduction to the Instruction Set	46

Loading and Storing Data	47
LDR and STR	47
Other Usage for LDR	51
LDM and STM	52
PUSH and POP	56
Functions and Function Invocation	57
Arithmetic Operations	60
Branching and Conditional Execution	61
Thumb State	64
Switch-Case	65
Miscellaneous	67
Just-in-Time and Self-Modifying Code	67
Synchronization Primitives	67
System Services and Mechanisms	68
Instructions	70
Walk-Through	71
Next Steps	77
Exercises	78
Chapter 3 The Windows Kernel	87
Windows Fundamentals	88
Memory Layout	88
Processor Initialization	89
System Calls	92
Interrupt Request Level	104
Pool Memory	106
Memory Descriptor Lists	106
Processes and Threads	107
Execution Context	109
Kernel Synchronization Primitives	110
Lists	111
Implementation Details	112
Walk-Through	119
Exercises	123
Asynchronous and Ad-Hoc Execution	128
System Threads	128
Work Items	129
Asynchronous Procedure Calls	131
Deferred Procedure Calls	135
Timers	140
Process and Thread Callbacks	142
Completion Routines	143
I/O Request Packets	144
Structure of a Driver	146
Entry Points	147
Driver and Device Objects	149

IRP Handling	150
A Common Mechanism for User-Kernel Communication	150
Miscellaneous System Mechanisms	153
Walk-Throughs	155
An x86 Rootkit	156
An x64 Rootkit	172
Next Steps	178
Exercises	180
Building Confidence and Solidifying Your Knowledge	180
Investigating and Extending Your Knowledge	182
Analysis of Real-Life Drivers	184
Chapter 4 Debugging and Automation	187
The Debugging Tools and Basic Commands	188
Setting the Symbol Path	189
Debugger Windows	189
Evaluating Expressions	190
Process Control and Debut Events	194
Registers, Memory, and Symbols	198
Breakpoints	208
Inspecting Processes and Modules	211
Miscellaneous Commands	214
Scripting with the Debugging Tools	216
Pseudo-Registers	216
Aliases	219
Language	226
Script Files	240
Using Scripts Like Functions	244
Example Debug Scripts	249
Using the SDK	257
Concepts	258
Writing Debugging Tools Extensions	262
Useful Extensions, Tools, and Resources	264
Chapter 5 Obfuscation	267
A Survey of Obfuscation Techniques	269
The Nature of Obfuscation: A Motivating Example	269
Data-Based Obfuscations	273
Control-Based Obfuscation	278
Simultaneous Control-Flow and Data-Flow Obfuscation	284
Achieving Security by Obscurity	288
A Survey of Deobfuscation Techniques	289
The Nature of Deobfuscation: Transformation Inversion	289
Deobfuscation Tools	295
Practical Deobfuscation	312

Case Study	328
First Impressions	328
Analyzing Handlers Semantics	330
Symbolic Execution	333
Solving the Challenge	334
Final Thoughts	336
Exercises	336
Appendix	
Sample Names and Corresponding SHA1 Hashes	341
Index	343



Introduction

The reverse engineering learning process is similar to that of foreign language acquisition for adults. The first phase of learning a foreign language begins with an introduction to letters in the alphabet, which are used to construct words with well-defined semantics. The next phase involves understanding the grammatical rules governing how words are glued together to produce a proper sentence. After being accustomed to these rules, one then learns how to stitch multiple sentences together to articulate complex thoughts. Eventually it reaches the point where the learner can read large books written in different styles and still understand the thoughts therein. At this point, one can read reference books on the more esoteric aspects of the language—historical syntax, phonology, and so on.

In reverse engineering, the language is the architecture and assembly language. A word is an assembly instruction. Paragraphs are sequences of assembly instructions. A book is a program. However, to fully understand a book, the reader needs to know more than just vocabulary and grammar. These additional elements include structure and style of prose, unwritten rules of writing, and others. Understanding computer programs also requires a mastery of concepts beyond assembly instructions.

It can be somewhat intimidating to start learning an entirely new technical subject from a book. However, we would be misleading you if we were to claim that reverse engineering is a simple learning endeavor and that it can be completely mastered by reading this book. The learning process is quite involved because it requires knowledge from several disparate domains of knowledge. For example, an effective reverse engineer needs to be knowledgeable in computer architecture, systems programming, operating systems, compilers, and so on; for certain areas, a strong mathematical background is necessary. So how do you

know where to start? The answer depends on your experience and skills. Because we cannot accommodate everyone's background, this introduction outlines the learning and reading methods for those without any programming background. You should find your "position" in the spectrum and start from there.

For the sake of discussion, we loosely define reverse engineering as the process of understanding a system. It is a problem-solving process. A system can be a hardware device, a software program, a physical or chemical process, and so on. For the purposes of the book, the system is a software program. To understand a program, you must first understand how software is written. Hence, the first requirement is knowing how to program a computer through a language such as C, C++, Java, and others. We suggest first learning C due to its simplicity, effectiveness, and ubiquity. Some excellent references to consider are *The C Programming Language*, by Brian Kernighan and Dennis Ritchie (Prentice Hall, 1988) and *C: A Reference Manual*, by Samuel Harbison (Prentice Hall, 2002). After becoming comfortable with writing, compiling, and debugging basic programs, consider reading *Expert C Programming: Deep C Secrets*, by Peter van der Linden (Prentice Hall, 1994). At this point, you should be familiar with high-level concepts such as variables, scopes, functions, pointers, conditionals, loops, call stacks, and libraries. Knowledge of data structures such as stacks, queues, linked lists, and trees might be useful, but they are not entirely necessary for now. To top it off, you might skim through *Compilers: Principles, Techniques, and Tools*, by Alfred Aho, Ravi Sethi, and Jeffrey Ullman, (Prentice Hall, 1994) and *Linkers and Loaders*, by John Levine (Morgan Kaufmann, 1999), to get a better understanding of how a program is really put together. The key purpose of reading these books is to gain exposure to basic concepts; you do not have to understand every page for now (there will be time for that later). Overachievers should consider *Advanced Compiler Design and Implementation*, by Steven Muchnick (Morgan Kaufmann, 1997).

Once you have a good understanding of how programs are generally written, executed, and debugged, you should begin to explore the program's execution environment, which includes the processor and operating system. We suggest first learning about the Intel processor by skimming through *Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture* by Intel, with special attention to Chapters 2–7. These chapters explain the basic elements of a modern computer. Readers interested in ARM should consider *Cortex-A Series Programmer's Guide* and *ARM Architecture Reference Manual ARMv7-A and ARMv7-R Edition* by ARM. While our book covers x86/x64/ARM, we do not discuss every architectural detail. (We assume that the reader will refer to these manuals, as necessary.) In skimming through these manuals, you should have a basic appreciation of the technical building blocks of a computing system. For a more conceptual understanding, consider *Structured Computer Organization* by Andrew Tanenbaum (Prentice Hall, 1998). All readers should also consult the *Microsoft PE*

and COFF Specification. At this point, you will have all the necessary background to read and understand Chapter 1, “x86 and x64,” and Chapter 2, “ARM.”

Next, you should explore the operating system. There are many different operating systems, but they share many common concepts including processes, threads, virtual memory, privilege separation, multi-tasking, and so on. The best way to understand these concepts is to read *Modern Operating Systems*, by Andrew Tanenbaum (Prentice Hall, 2005). Although Tanenbaum’s text is excellent for concepts, it does not discuss important technical details for real-life operating systems. For Windows, you should consider skimming through *Windows NT Device Driver Development*, by Peter Viscarola and Anthony Mason (New Riders Press, 1998); although it is a book on driver development, the background chapters provide an excellent and concrete introduction to Windows. (It is also excellent supplementary material for the Windows kernel chapter in this book.) For additional inspiration (and an excellent treatment of the Windows memory manager), you should also read *What Makes It Page? The Windows 7 (x64) Virtual Memory Manager*, by Enrico Martignetti (CreateSpace Independent Publishing Platform, 2012).

At this point, you would have all the necessary background to read and understand Chapter 3 “The Windows Kernel.” You should also consider learning Win32 programming. *Windows System Programming*, by Johnson Hart (Addison-Wesley Professional, 2010), and *Windows via C/C++*, by Jeffrey Richter and Christophe Nasarre (Microsoft Press, 2007), are excellent references.

For Chapter 4, “Debugging and Automation,” consider *Inside Windows Debugging: A Practical Guide to Debugging and Tracing Strategies in Windows*, by Tarik Soulati (Microsoft Press, 2012), and *Advanced Windows Debugging*, by Mario Hewardt and Daniel Pravat (Addison-Wesley Professional, 2007).

Chapter 5, “Obfuscation,” requires a good understanding of assembly language and should be read after the x86/x64/ARM chapters. For background knowledge, consider *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection*, by Christian Collberg and Jasvir Nagra (Addison-Wesley Professional, 2009).

NOTE This book includes exercises and walk-throughs with real, malicious viruses and rootkits. We intentionally did this to ensure that readers can immediately apply their newly learned skills. The malware samples are referenced in alphabetical order (Sample A, B, C, ...), and you can find the corresponding SHA1 hashes in the Appendix. Because there may be legal concerns about distributing such samples with the book, we decided not to do so; however, you can download these samples by searching various malware repositories, such as www.malware.lu, or request them from the forums at <http://kernelmode.info>. Many of the samples are from famous hacking incidents that made worldwide news, so they should be interesting. Perhaps some enthusiastic readers will gather all the samples in a package and share them on BitTorrent.