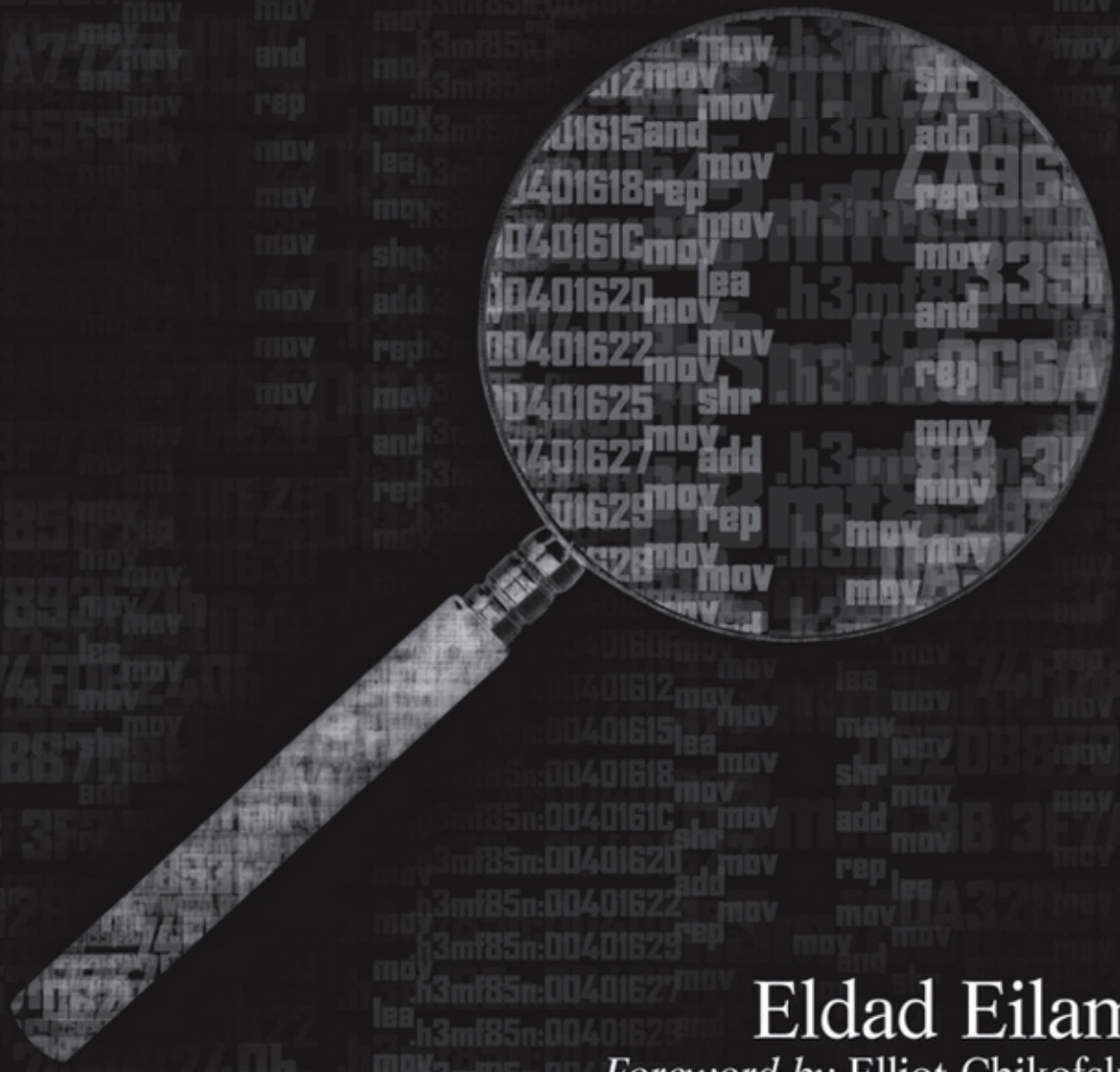
REVERSING

Secrets of Reverse Engineering



Eldad Eilam

Foreword by Elliot Chikofsky

Reversing: Secrets of Reverse Engineering

Reversing: Secrets of Reverse Engineering

Eldad Eilam

Wiley Publishing, Inc.

Reversing: Secrets of Reverse Engineering
Published by
Wiley Publishing, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Published simultaneously in Canada

Library of Congress Control Number: 2005921595

ISBN-10: 0-7645-7481-7
ISBN-13: 978-0-7645-7481-8

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

1B/QR/QU/QV/IN

Credits

Executive Editor
Robert Elliott

Development Editor
Eileen Bien Calabro

Copy Editor
Foxxe Editorial Services

Editorial Manager
Mary Beth Wakefield

Vice President & Executive Group Publisher
Richard Swadley

Vice President and Publisher
Joseph B. Wikert

Project Editor
Pamela Hanley

Project Coordinator
Ryan Steffen

Graphics and Production Specialists
Denny Hager
Jennifer Heleine
Lynsey Osborn
Mary Gillot Virgin

Quality Control Technician
Leeann Harney

Proofreading and Indexing
TECHBOOKS Production Services

Cover Designer
Michael Trent

Foreword

It is amazing, and rather disconcerting, to realize how much software we run without knowing for sure what it does. We buy software off the shelf in shrink-wrapped packages. We run setup utilities that install numerous files, change system settings, delete or disable older versions and superceded utilities, and modify critical registry files. Every time we access a Web site, we may invoke or interact with dozens of programs and code segments that are necessary to give us the intended look, feel, and behavior. We purchase CDs with hundreds of games and utilities or download them as shareware. We exchange useful programs with colleagues and friends when we have tried only a fraction of each program's features.

Then, we download updates and install patches, trusting that the vendors are sure that the changes are correct and complete. We blindly hope that the latest change to each program keeps it compatible with all of the rest of the programs on our system. We rely on much software that we do not understand and do not know very well at all.

I refer to a lot more than our desktop or laptop personal computers. The concept of ubiquitous computing, or "software everywhere," is rapidly putting software control and interconnection in devices throughout our environment. The average automobile now has more lines of software code in its engine controls than were required to land the Apollo astronauts on the Moon.

Today's software has become so complex and interconnected that the developer often does not know all the features and repercussions of what has been created in an application. It is frequently too expensive and time-consuming to test all control paths of a program and all groupings of user options. Now, with multiple architecture layers and an explosion of networked platforms that the software will run on or interact with, it has become literally impossible for all

combinations to be examined and tested. Like the problems of detecting drug interactions in advance, many software systems are fielded with issues unknown and unpredictable.

Reverse engineering is a critical set of techniques and tools for understanding what software is really all about. Formally, it is "the process of analyzing a subject system to identify the system's components and their interrelationships and to create representations of the system in another form or at a higher level of abstraction"(IEEE 1990). This allows us to visualize the software's structure, its ways of operation, and the features that drive its behavior. The techniques of analysis, and the application of automated tools for software examination, give us a reasonable way to comprehend the complexity of the software and to uncover its truth.

Reverse engineering has been with us a long time. The conceptual Reversing process occurs every time someone looks at someone else's code. But, it also occurs when a developer looks at his or her own code several days after it was written. Reverse engineering is a discovery process. When we take a fresh look at code, whether developed by ourselves or others, we examine and we learn and we see things we may not expect.

While it had been the topic of some sessions at conferences and computer user groups, reverse engineering of software came of age in 1990. Recognition in the engineering community came through the publication of a taxonomy on reverse engineering and design recovery concepts in *IEEE Software* magazine. Since then, there has been a broad and growing body of research on Reversing techniques, software visualization, program understanding, data reverse engineering, software analysis, and related tools and approaches. Research forums, such as the annual international Working Conference on Reverse Engineering (WCRE), explore, amplify, and expand the value of available techniques. There is now increasing interest in binary Reversing, the principal focus of this book, to support platform migration, interoperability, malware detection, and problem determination.

As a management and information technology consultant, I have often been asked: "How can you possibly condone reverse engineering?" This is soon followed by: "You've developed and sold software. Don't you want others to respect and protect your copyrights and intellectual property?" This discussion usually starts from the negative connotation of the term reverse engineering, particularly in software license agreements. However, reverse engineering technologies are of value in many ways to producers and consumers of software along the supply chain.

A stethoscope could be used by a burglar to listen to the lock mechanism of a safe as the tumblers fall in place. But the same stethoscope could be used by your family doctor to detect breathing or heart problems. Or, it could be used by a computer technician to listen closely to the operating sounds of a sealed disk drive to diagnose a problem without exposing the drive to