FOREWORD BY
HOWARD SCHMIDT

# CORE SOFTWARE SECURITY

## SECURITY AT THE SOURCE

JAMES RANSOME
ANMOL MISRA

CRC Press
Taylor & Francis Group
AN AUERBACH BOOK

# CORE SOFTWARE SECURITY

## SECURITY AT THE SOURCE

JAMES RANSOME
ANMOL MISRA

CONTRIBUTING AUTHOR (CHAPTER 9): BROOK SCHOENFIELD

FOREWORD BY
HOWARD SCHMIDT


CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **Informa** business
AN AUERBACH BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742



Printed on acid-free paper
Version Date: 20131031

International Standard Book Number-13: 978-1-4665-6095-6 (Hardback)



**Library of Congress Cataloging-in-Publication Data**

Ransome, James F.
Core software security : security at the source / James Ransome and Anmol Misra.
pages cm
Includes bibliographical references and index.
ISBN 978-1-4665-6095-6 (hardback)
1. Computer security. I. Title.

QA76.9.A25R356 2013
005.8--dc23 2013042460

**Visit the Taylor & Francis Web site at**
**http://www.taylorandfrancis.com**

**and the CRC Press Web site at**
**http://www.crcpress.com**

# CORE SOFTWARE SECURITY

## SECURITY AT THE SOURCE

# Dedication

To Dr. Tony (Vern) Dubendorf, who passed away earlier this year. He was a true friend, co-worker, collaborator, confidant, co-researcher, co-author, and co-architect of the Getronics Wireless Integrated Security, Design, Operations & Management (WISDOM) solution.

—James Ransome

To Dad, Mom, Esu, Anu, Mausi, and Prince.

—Anmol Misra

# Foreword

The global cyber security threat is increasing on a regular basis, if not daily. The recurring question is how we address the current threat of global cyber security. The authors have aptly named their book in response to this question, in that the answer is to create software that has as minimal vulnerabilities as possible. In other words, focus on securing at the source first, instead of taking shortcuts by only trying to secure network infrastructure. Perimeter security and defense-in-depth have their place in security, but software security is the first line of defense and should come first. If you have fewer vulnerabilities at the source, it also takes out the financial benefit of nation states or organized crime stockpiling cyber weapons based on current vulnerabilities. Not only must we get better at it, we must make the solutions cost-effective, operationally relevant, and feasible, based on real-world experience, and worth the investment. Securing at the source requires securing the software, which is at the heart of cyber infrastructure. One of the things we have been constantly facing over the last 20 years is that software has become a critical component of every part of our critical infrastructure and everyday lives. We are already seeing software embedded within a vast variety of things we use in our daily lives—from smart meters in our home to cars we drive. Unfortunately, software security has not evolved at the same pace, and many software products are still developed in an environment with the intent that they fix the problem after release rather than doing it right the first time around. There are two major issues with this:

1. There are no shortages of threats out there today; therefore, people who are looking to exploit software vulnerabilities have a pretty

fertile field in which to work. As a consequence, we have to make sure we are doing better vulnerability management. We also have to look toward the future and ask ourselves, "How can we avoid having these types of vulnerabilities in future generations of software that we are increasingly dependent on?" The answer to this question is particularly important because it is very beneficial to companies to reduce these vulnerabilities and to stop them during the software development process. It is significantly less expensive to build security in through the use of a SDL than to come back and fix it post-release.

2. The second issue is that we need to start looking at a whole generation of what is referred to as "zero-day vulnerabilities." If we can eliminate the likelihood of finding a zero day by not allowing the vulnerabilities to take place from the very beginning by adhering to the best practices of a solid SDL, it will save companies money, make the software and its users more secure, the critical infrastructure more resilient, and overall, more beneficial to us all.

As the Executive Director of the Software Assurance Forum for Excellence in Code (SAFECode), a nonprofit organization dedicated exclusively to increasing trust in information and communications technology products and services through the advancement of effective software assurance methods, I currently have a major focus on security training for developers. The lack of security awareness and education among the software engineering workforce can be a significant obstacle to organizations working to implement software security programs. However, better training for software developers so they have the skills needed to write secure code is just one of the variables in the software security equation. Software projects are under the constraints of costs and tight timelines. In those situations, it is inevitable that security is sacrificed somewhere because of shortcuts taken. Cost, time, and resources are typically the triad of software development supporting security, and if you sacrifice one of the three, security and quality suffer. A software development environment is built around a programmer who is pressured on every side to work faster, to cut corners, and to produce more code at the expense of security and quality.

It is impossible to have 100 percent security, but the developers and their management should always strive to maximize the mitigation of risk. It is about making it so difficult to access in an unauthorized manner that adversaries:

- Have to utilize traceable and obvious means to gain access, so they are noticed right away
- Spend so much time trying to gain access that they eventually are noticed
- Give up and move on to easier targets

Ensuring that everyone touching the product development lifecycle has the knowledge they need to support an organization's software security process is a fundamental challenge for any organization committed to software security success. The goal is to remove the pain that organizations face in developing a custom program of their own resource constraints and knowledge vacuums.

Developers are often under intense pressure to deliver more features on time and under budget. Few developers get the time to review their code for potential security vulnerabilities. When they do get the time, they often don't have secure-code training and lack the automated tools, embedded processes and procedures, and resources to prevent hackers from using hundreds of common exploit techniques to trigger malicious attacks. Like it or not, the hard work of developers often takes the brunt of malicious hacker attacks. So what can software vendors do? A big part of the answer is relatively old-fashioned: The developers need to be provided with incentives, better tools, and proper training.

Unfortunately, it currently makes more business sense not to produce secure software products than it does to produce secure software products. Any solution needs to address this as a fundamental market failure instead of simply wishing it were not true. If security is to be a business goal, then it needs to make business sense. In the end, security requirements are in fact the same as any business goals and should be addressed as equally important. Employers should expect their employees to take pride in and own a certain level of responsibility for their work. And employees should expect their employers to provide the tools and training they need to get the job done. With these expectations established and goals agreed on, perhaps the software industry can do a better job of strengthening the security of its products by reducing software vulnerabilities.

This book discusses a process and methodology to develop software in such a way that security considerations play a key role in its development. It speaks to executives, to managers at all levels, and to technical leaders, and in that way it is unique. It also speaks to students and developers so they can understand the process of developing software with security in

mind and find resources to help them do so. The information in this book provides a foundation for executives, project managers, and technical leaders to improve the software they create and to improve the security, quality, and privacy of the software we all use.

Software developers know how to write software in a way that provides a high level of security and robustness. So why don't software developers practice these techniques? This book looks to answer this question in two parts:

1. Software is determined to be secure as a result of an analysis of how the program is to be used, under what conditions, and the security requirements it must meet in the environment in which it is to be deployed. The SDL must also extend beyond the release of the product in that if the assumptions underlying the software in an unplanned operational environment and their previously implied requirements do not hold, the software may no longer be secure and the SDL process may start over in part or as a whole if a complete product redesign is required. In this sense, the authors establish the need for accurate and meaningful security requirements and the metrics to govern them as well as examples of how to develop them. It also assumes that the security requirements are not all known prior to the development process and describes the process by which they are derived, analyzed, and validated.
2. Software executives, leaders, and managers must support the robust coding practices and required security enhancements as required by a business-relevant SDL as well supporting the staffing requirements, scheduling, budgeting, and resource allocations required for this type of work. The authors do an excellent job of describing the process, requirements, and management of metrics for people in these roles so they can accurately assess the impact and resources required for a SDL that is relevant to and work best in their organization and environment. Given that this is an approached designed from real-life, on-the-ground challenges and experiences, the authors describe how to think about issues in order to develop effective approaches and manage them as a business process.

I particularly like the addition of Brook Schoenfield to the book as the author of Chapter 9, bringing a seasoned principal enterprise and

software security architect with "in the trenches" experience to explain how security architecture fits into the SDL process in the "real world." In particular, he provides a unique and valuable approach to addressing the aspects of SDL and security architecture that has been field-tested and really works in the agile software development process.

I have known Dr. James Ransome for many years, and I am very pleased that he has chosen this topic for his 10th book on information security. Having recently served as the Special Assistant to the President and the Cyber Security Coordinator for the federal government, in addition to many senior leadership roles in the cyber security government and enterprise space, I can confidently say that this is currently the most critical area of information and global cyber security to fix. This has been and continues to be more of a business and process issue than it is technical. *Core Software Security: Security at the Source* adds great value to the typical training resources currently available in that it takes the elements of the best publically known SDLs and provides operational, business-relevant, cost-effective metrics. I believe that what James Ransome and Anmol Misra have written has hit the mark on this topic and will serve the community for many years to come as both a practical guide for professionals and as an academic textbook.

Hon. Howard A. Schmidt
Partner, Ridge Schmidt Cyber
Executive Director, The Software Assurance Forum for Excellence in Code (SAFECode)

About Hon. Howard A. Schmidt

Hon. Howard Schmidt serves as a partner in the strategic advisory firm Ridge Schmidt Cyber, an executive services firm that helps leaders in business and government navigate the increasing demands of cyber security. He serves in this position with Tom Ridge, the first Secretary of the U.S. Department of Homeland Security. Prof. Schmidt also serves as executive director of The Software Assurance Forum for Excellence in Code (SAFECode).

Prof. Schmidt brings together talents in business, defense, intelligence, law enforcement, privacy, academia, and international relations, gained from a distinguished career spanning 40 years. He recently served as Special Assistant to the President and the Cyber Security Coordinator for the federal government. In this role, Mr. Schmidt was responsible for coordinating interagency cyber security policy development and implementation, and for coordinating engagement with federal, state, local, international, and private-sector cyber security partners.

Previously, Prof. Schmidt was the President and CEO of the Information Security Forum (ISF). Before serving on the ISF, he was Vice President and Chief Information Security Officer and Chief Security Strategist for eBay Inc., and formerly served as the Chief Security Officer for Microsoft Corp. He also served as Chief Security Strategist for the US-CERT Partners Program for the Department of Homeland Security. Mr. Schmidt also brings to bear over 26 years of military service. Beginning active duty with the Air Force, he later joined the Arizona Air National Guard. With the Air Force he served in a number of military and civilian roles, culminating as Supervisory Special Agent with the Office of Special Investigations (AFOSI). He finished his last 12 years as an Army Reserve Special Agent with the Criminal Investigation Division's Computer Crime Unit, all while serving for over a decade as a police officer with the Chandler, Arizona, Police Department.

Prof. Schmidt holds a bachelor's degree in business administration (BSBA) and a master's degree in organizational management (MAOM) from the University of Phoenix. He also holds an Honorary Doctorate degree in Humane Letters. Howard is a Professor of Research at Idaho State University, Adjunct Distinguished Fellow with Carnegie Mellon's CyLab, and a Distinguished Fellow of the Ponemon Privacy Institute.

Howard is also a ham radio operator (W7HAS), private pilot, outdoorsman, and avid Harley-Davidson rider. He is married to Raemarie J. Schmidt, a retired forensic scientist and researcher, and instructor in the field of computer forensics. Together, they are proud parents, and happy grandparents.

# Preface

The age of the software-driven machine has taken significant leaps over the last few years. Human tasks such as those of fighter pilots, stock-exchange floor traders, surgeons, industrial production and power-plant operators that are critical to the operation of weapons systems, medical systems, and key elements of our national infrastructure, have been, or are rapidly being taken over by software. This is a revolutionary step in the machine whose brain and nervous system is now controlled by software-driven programs taking the place of complex nonrepetitive tasks that formerly required the use of the human mind. This has resulted in a paradigm shift in the way the state, military, criminals, activists, and other adversaries can attempt to destroy, modify, or influence countries, infrastructures, societies, and cultures. This is true even for corporations, as we have seen increasing cases of cyber corporate espionage over the years. The previous use of large armies, expensive and devastating weapons systems and platforms, armed robberies, the physical stealing of information, violent protests, and armed insurrection are quickly being replaced by what is called cyber warfare, crime, and activism.

In the end, the cyber approach may have just as profound affects as the techniques used before in that the potential exploit of software vulnerabilities could result in:

- Entire or partial infrastructures taken down, including power grids, nuclear power plants, communication media, and emergency response systems
- Chemical plants modified to create large-yield explosions and/or highly toxic clouds

- Remote control, modification, or disablement of critical weapon systems or platforms
- Disablement or modification of surveillance systems
- Criminal financial exploitation and blackmail
- Manipulation of financial markets and investments
- Murder or harm to humans through the modification of medical support systems or devices, surgery schedules, or pharmaceutical prescriptions
- Political insurrection and special-interest influence through the modification of voting software, blackmail, or brand degradation though website defacement or underlying Web application takedown or destruction

A side effect of the cyber approach is that it has given us the ability to do the above at a scale, distance, and degree of anonymity previously unthought of from jurisdictionally protected locations through remote exploitation and attacks. This gives government, criminal groups, and activists abilities to proxy prime perpetuators to avoid responsibility, detection, and political fallout.

Although there is much publicity regarding network security, the real Achilles heel is the (insecure) software which provides the potential ability for total control and/or modification of a target as described above. The criticality of software security as we move quickly toward this new age of tasks previously relegated to the human mind being replaced by software-driven machines cannot be underestimated. It is for this reason that we have written this book. In contrast, and for the foreseeable future, software programs are and will be written by humans. This also means that new software will keep building on legacy code or software that was written prior to security being taken seriously, or before sophisticated attacks became prevalent. As long as humans write the programs, the key to successful security for these programs is in making the software development program process more efficient and effective. Although the approach of this book includes people, process, and technology approaches to software security, we believe the people element of software security is still the most important part to manage as long as software is developed, managed, and exploited by humans. What follows is a step-by-step process for software security that is relevant to today's technical, operational, business, and development environments, with a focus on what humans can

do to control and manage the process in the form of best practices and metrics. We will always have security issues, but this book should help in minimizing them when software is finally released or deployed. We hope you enjoy our book as much as we have enjoyed writing it.

## About the Book

This book outlines a step-by-step process for software security that is relevant to today's technical, operational, business, and development environments. The authors focus on what humans can do to control and manage a secure software development process in the form of best practices and metrics. Although security issues will always exist, this book will teach you how to maximize your organization's ability to minimize vulnerabilities in your software products before they are released or deployed, by building security into the development process. The authors have worked with Fortune 500 companies and have often seen examples of the breakdown of security development lifecycle (SDL) practices. In this book, we take an experience-based approach to applying components of the best available SDL models in dealing with the problems described above, in the form of a SDL software security best practices model and framework. *Core Software Security: Security at the Source* starts with an overview of the SDL and then outlines a model for mapping SDL best practices to the software development lifecycle, explaining how you can use this model to build and manage a mature SDL program. Although security is not a natural component of the way industry has been building software in recent years, the authors believe that security improvements to development processes are possible, practical, and essential. They trust that the software security best practices and model presented in this book will make this clear to all who read the book, including executives, managers, and practitioners.

## Audience

This book is targeted toward anyone who is interested in learning about software security in an enterprise environment, including product security and quality executives, software security architects, security consultants, software development engineers, enterprise SDLC program managers,

chief information security officers, chief technology officers, and chief privacy officers whose companies develop software. If you want to learn about how software security should be implemented in developing enterprise software, this is a book you don't want to skip.

## Support

Errata and support for this book are available on the CRC Press book website.

## Structure

This book is divided into three different sections and 10 chapters. Chapter 1 provides an introduction to the topic of software security and why it is important that we get it right the first time. Chapter 2 introduces challenges of making software secure and the SDL framework. Chapters 3 through 8 provide mapping of our SDL with its associated best practices to a generic SDLC framework. Chapter 9 provides a seasoned software security architect's view on the successful application of the solutions proposed in Chapters 3 through 8. Chapter 9 also explains real-world approaches to the typical challenges that are presented when making secure software. We conclude, in Chapter 10, by describing real-world security threats that a properly architected, implemented, and managed SDL program will mitigate against.

## Assumptions

This book assumes that a reader is familiar with basics of software development (and methodologies) and basic security concepts. Knowledge of the SDL, different types of security testing, and security architecture is recommended but not required. For most topics, we gently introduce readers to the topic before diving deep into that particular topic.

# Acknowledgments

Writing a book is a journey, and without support from mentors, friends, colleagues, and family, it can be a difficult one. Many people have been instrumental in helping us write this book. First, we would like to thank our editor, John Wyzalek, at CRC Press, for his patience, support, and commitment to the project. We would also like to thank the production team at DerryField Publishing: Theron Shreve, Lynne Lackenbach, and Marje Pollack.

Both authors would like to thank the Hon. Howard A. Schmidt [Partner, Ridge Schmidt Cyber; Executive Director, The Software Assurance Forum for Excellence in Code (SAFECode); and former Special Assistant to the President and the Cyber Security Coordinator for the federal government], and Dena Haritos Tsamitis (Director, Information Networking Institute; Director of Education, Training, and Outreach, CyLab Carnegie Mellon University) for their support with this project. We would also like to thank Brook Schoenfield, who has joined us in this journey to prove there is another way to architect, implement, and manage software security than what is the current status quo, and for his contribution in writing a chapter of this book as a contributing author. We would like to thank the security community to which we both belong and are proud of. Finally, we would like to thank the people with whom we have worked and interacted over the years.

—James Ransome and Anmol Misra

I would like to take this opportunity to give thanks to my wife, Gail, for her patience and understanding and for serving as my preliminary proofreader. A special thanks to my co-author, Anmol Misra, who has joined me as a partner in developing this critical message over the last three years that has resulted in the book you are about to read. A special thanks to Howard Schmidt for writing the foreword for this book on a subject and message for which we both share a passion, and that we both want to get out to practitioners and decision makers alike. And finally, I leave you with the following quote by Walter Bagehot: "The greatest pleasure in life is doing that which people say we cannot do."

—James Ransome

Over the years, many people have mentored and helped me. I would like to thank them all, but space is limited. You know who you are, and I would like to thank you for your patience, encouragement, and support. I would like to thank my co-author, James Ransome. He has been a mentor and a partner over the years and has helped me in more ways than I can mention. Finally, I would like to take this opportunity to thank my family—Mom, Dad, Sekhar, Anupam, and Mausi. Nothing would be possible without their unquestioning support and love. You have been asking me if I am going to take a break from my writing and if I will finally have a "normal" schedule. Yes, I will now—hopefully.

—Anmol Misra