THE EXPERT'S VOICE® IN SECURITY

Foreword by Vint Cerf, a Founding Father of the Internet

Foundations of Security

What Every Programmer Needs to Know

What every programmer needs to know about security, illustrated with running examples of web applications and stories of what's gone wrong in the past.

Neil Daswani, Christoph Kern, and Anita Kesavan

Foreword by Vinton G. Cerf

Apress®

Foundations of Security

What Every Programmer Needs to Know

Neil Daswani, Christoph Kern,
and Anita Kesavan

Apress®

Foundations of Security: What Every Programmer Needs to Know

ISBN-13 (pbk): 978-1-59059-784-2

ISBN-10 (pbk): 1-59059-784-2

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Jonathan Gennick
Technical Reviewer: Dan Pilone
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Gennick, Jonathan Hassell, James Huddleston, Chris Mills, Matthew Moodie, Dominic Shakeshaft, Jim Sumser, Matt Wade
Project Manager: Kylie Johnston
Copy Edit Manager: Nicole Flores
Copy Editor: Damon Larson
Assistant Production Director: Kari Brooks-Copony
Production Editor: Ellie Fountain
Compositor: Dina Quan
Proofreader: Liz Welch
Indexer: Julie Grady
Artist: Kinetic Publishing Services, LLC
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `http://www.springeronline.com`.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.

The source code for this book is available to readers at `http://www.apress.com` in the Source Code/Download section.

This book is dedicated to Dad, who provided me my foundations,
and Mom, who taught me what I needed to know.
—N. Daswani

Contents at a Glance

PART 3 Introduction to Cryptography

PART 4 Appendixes

Contents

Foreword

When Neil Daswani and Christoph Kern invited me to write a foreword to the book you are reading now, I accepted without hesitation and with a good deal of pleasure. This timely volume is solidly grounded in theory and practice and is targeted at helping programmers increase the security of the software they write. Despite the long history of programming, it seems as if bug-free and resilient software continues to elude us. This problem is exacerbated in networked environments because attacks against the vulnerabilities in software can come from any number of other computers and, in the Internet, that might mean millions of potential attackers. Indeed, the computers of the Internet that interact with each other are in some sense performing unplanned and unpredictable tests between the software complements of pairs of machines. Two machines that start out identically configured will soon become divergent as new software is downloaded as a consequence of surfing the World Wide Web or as updates are applied unevenly among the interacting machines. This richly diverse environment exposes unexpected vulnerabilities, some of which may be exploited deliberately by hackers intent on causing trouble or damage and who may even have pecuniary motivations for their behavior. So-called bot armies are available in the millions to be directed against chosen targets, overwhelming the defenses of some systems by the sheer volume of the attack. In other cases, known weaknesses are exploited to gain control of the target machines or to introduce viruses, worms, or Trojan horses that will do further damage.

Programmers writing for networked environments have a particularly heavy responsibility to be fully aware of the way in which these vulnerabilities may come about and have a duty to do everything they can to discover and remove them or to assure that they are eliminated by careful design, implementation, and testing. It takes discipline and a certain amount of paranoia to write secure software. In some ways it is like driving defensively. You must assume you are operating in a hostile environment where no other computer can be trusted without demonstrating appropriate and verifiable credentials. Even this is not enough. In a kind of nightmare scenario, someone with a USB memory stick can bypass all network defenses and inject software directly into the computer. Such memory sticks emulate disks and can easily pick up viruses or worms when they are used on unprotected computers, and when reused elsewhere, can propagate the problem. All input must be viewed with suspicion until cleared of the possibility of malformation.

Vulnerability can exist at all layers of the Internet protocol architecture and within the operating systems. It is naive to imagine that simply encrypting traffic flowing between pairs of computers on the Internet is sufficient to protect against exploitation. An obvious example is a virus attached to an e-mail that is sent through the Internet fully encrypted at the IP layer using IPsec. Once the message is decrypted packet by packet and reassembled, the virus will be fully ready to do its damage unless it is detected at the application layer by the e-mail client, or possibly by the mail transport agent that delivers the e-mail to the target recipient.

It is vital to understand not only how various attacks are carried out, but also how the vulnerabilities that enable these attacks arise. Programs that fail to check that inputs are properly