Innocent Code

A Security Wake-Up Call for Web Programmers

Sverre H. Huseby

John Wiley & Sons, Ltd

This page intentionally left blank

"This book is much more than a wake-up call. It is also an eye-opener. Even for those who are already awake to the problems of Web server security, it is a serious guide for what to do and what not to do."

Peter G. Neumann, risks.org

This page intentionally left blank

Innocent Code

A Security Wake-Up Call for Web Programmers

Sverre H. Huseby

John Wiley & Sons, Ltd

John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester,
West Sussex PO19 8SQ, England

Telephone (+44) 1243 779777

Email (for orders and customer service enquiries): cs-books@wiley.co.uk
Visit our Home Page on www.wileyeurope.com or www.wiley.com

Other Wiley Editorial Offices

John Wiley & Sons Inc., 111 River Street, Hoboken, NJ 07030, USA

Jossey-Bass, 989 Market Street, San Francisco, CA 94103-1741, USA

Wiley-VCH Verlag GmbH, Boschstr. 12, D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 33 Park Road, Milton, Queensland 4064, Australia

John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop #02-01, Jin Xing Distripark, Singapore 129809

John Wiley & Sons Canada Ltd, 22 Worcester Road, Etobicoke, Ontario, Canada M9W 1L1

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Cataloging-in-Publication Data

Huseby, Sverre H.
Innocent code : a security wake-up call for Web programmers / Sverre H. Huseby.
p. cm.
"A Wiley-Interscience publication."
ISBN 0-470-85744-7
1. Computer security. 2. Computer networks--Security measures. 3. World Wide Web--Security measures. I. Title.
QA76.9.A25H88 2003
005.8--dc22

2003015774

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN 0-470-85744-7

Typeset in 10.5/13pt Sabon by Laserwords Private Limited, Chennai, India
Printed and bound in Great Britain by Biddles Ltd, Guildford and King's Lynn
This book is printed on acid-free paper responsibly manufactured from sustainable forestry in which at least two trees are planted for each one used for paper production.

Contents

Foreword

There has been a rude awakening for the IT industry in the last few years. For nearly a decade corporations have been told by the media and consultants that they needed firewalls, intrusion detection systems and network scanning tools to stop the barrage of cyber attacks that we all read about daily. Hackers are stealing credit cards, booking flights to exotic locations for free and downloading personal information about the latest politicians' affair with an actress. We have all seen the stories and those of us with an inquisitive mind have all wondered how it really happens.

As the information security market grew into a vast commercial machine pushing network and operating system security technology and processes as the silver bullet to cure all ills, the IT industry itself grew in a new direction. Business leaders and marketing managers discovered that the lowest common denominator to any user (or potential user) is the web browser, and quite frankly why in the world wouldn't they want to appeal to all the possible clients out there? Why would you want to restrict the possibility of someone signing up for your service? Web enabling applications and company data was not just a trend, it has been a phenomena. Today there are web interfaces to almost all major applications from development source code systems to human resources payroll systems and sales tracking databases. When we browse the Web and the local weather is displayed so conveniently in the side-menu, it's a web application that put it there. When we check our online bank balance, it's a system of complex web applications that compute and display the balance.

Creating these vast complex pieces of technology is no trivial task. From a technology stance, Microsoft and Sun are leading the charge with platforms

and supporting languages that provide flexible and extensible bases from which to build. With flexibility comes choice, and whilst it is true that these platforms can provide excellent security functionality, the security level is a choice of the designer and developer. All of the platforms on offer today can equally create secure and insecure applications, and as with many things in life, the devil is in the details. When building a web application the details are almost exclusively the responsibility of the developer.

This book takes a unique and highly effective approach to educating the people that can effect a change by addressing the people who are actually responsible for writing code; the developers themselves. It is written by a developer for developers, which means it speaks the developer lingo and explains issues in a way that as a developer you will understand. By taking a pragmatic approach to the issue, the author walks you, the reader, through an overview of the issues and then delves into the devilish details supporting issues with examples and real life scenarios that are both easy to understand and easy to realize in your own code.

This book is a serious must have for all developers who are building web sites. I know you will enjoy it as much as I did.

Mark Curphey

Mark Curphey has a Masters degree in Information Security and runs the Open Web Application Security Project. He moderates the sister security mailing list to Bugtraq called webappsec that specializes in web application security. He is a former Director of Information Security for Charles Schwab, consulting manager for Internet security Systems and veteran of more banks and consulting clients than he cares to remember.

Acknowledgments

This book would have been less readable, less consistent, and more filled with bugs if it wasn't for a handful of smart friends and colleagues that helped me pinpoint troublesome areas along the way. All I did was to promise them a beer and honorable mention in this section, and they started spending hours and days (and some even weeks) helping me out.

First of all, Jan Ingvoldstad has spent an amazing amount of time reading, commenting, and suggesting improvements to almost every paragraph.

In addition, the following people have spent quite some time reading and commenting on early versions of the text: Lars Preben S. Arnesen, Erik Assum, Jon S. Bratseth, Per Otto Christensen, Per Kristian Gjermshus, Morten Grimnes, Leif John Korshavn, Rune Offerdal, Frode Sandnes, Frank Solem, Rune Steinberg, Kent Vilhelmsen and Sigmund Øy.

Kjetil Valstadsve made me rethink some sections, and Tore Anderson, Kjetil Barvik, Maja Bratseth, Lasse G. Dahl, Dennis Groves, Jan Kvile, Filip van Laenen, Glenn T. Lines, Kevin Spett, Thorkild Stray and Bjørn Stærk gave valuable feedback and ideas to parts of the text.

Please note that none of the people on this list of gratitude should be blamed for any errors or omissions whatsoever in this book. I was stupid enough not to follow all the advice given to me by these kind and experienced people, so I'm the only one to blame if you feel like blaming anyone for anything (concerning this book, that is).

I would also like to thank my editor Gaynor Redvers-Mutton and her friends at Wiley for believing in my book proposal even though most of their reviewers wanted to turn the book into a traditional infrastructure security thing. :-)

As I find book dedications quite meaningless, I'd rather say "hi" to Markus and Matilde in this section. Thanks for giving me good memories while you keep me busy throughout the days.

And last, but certainly not least, I bow deeply for my beloved wife, Hanne S. Finstad. She always makes me feel safe and free of worries. Without that kind of support (which I'm not sure she knows she's giving me), I would never have been able to write a book (cliche, but true anyway). She's the most creative, intelligent, beautiful, … oh, sorry. I'll tell her face to face instead.

S. H. H.

Introduction

This book is kind of weird. It's about the security of a web site, but it hardly mentions firewalls. It's about the security of information, but it says very little about encryption. So what's this book all about? It describes a small, and often neglected, piece of the web site security picture: Program code security.

Many people think that a good firewall, encrypted communication and staying up to date on software patches is all that is needed to make a web site secure. They're wrong. Many of today's web sites contain program code that make them dynamic. Code written using tools such as Java, PHP, Perl, ASP/VBScript, Zope, ColdFusion, and many more. Far too often, this code is written by programmers who seem to think that security is handled by the administrators. The effect is that an enormous number of dynamic web sites have logical holes in them that make them vulnerable to all kinds of nasty attacks. Even with both firewall and encryption in place.

Current programmer education tends to see security as off topic. Something for the administrators, or for some elite of security specialists. We learn how to program. Period. More specifically, to make programs that please the customers by offering the requested functionality. Some years ago, that would probably suffice. Back then, programs were internal to organizations. Every person with access to our program wanted it to operate correctly, so that they could do their day to day job.

In the age of the Web, however, most of us get to create programs that are available to the entire world. Legitimate users still just want the program to do its job for them. Unfortunately, our program is also available to lots of people who find amusement in making programs break. Or better, making them do things they were not supposed to do.

Until recently, those who find joy in breaking programs have put most of their effort in mass-produced software, creating exploits that will work on thousands of systems. In the last couple of years, however, focus on custom-made web applications has increased. International security mailing lists have been created to deal with the web application layer only, many good white papers have been written, and we have seen reports of the first few application level attacks in the media. With increased focus, chances are that more attackers will start working on application exploits. While the security people tend to keep up, the programmers are far behind. It's about time *we* started focusing on security too.

This book is written for the coders, those of us programming dynamic web applications. The book explains many common mistakes that coders tend to make, and how these mistakes may be exploited to the benefit of the attackers.

When reading the book, you may get the impression that the main focus is on how to abuse a web site rather than on how to build a site that can't be abused. The focus on destruction is deliberate: to build secure applications, one will need to know how programming mistakes may be abused. One will need to know how the attacker thinks when he snoops around looking for openings. To protect our code, we'll need to know the enemy. The best way to stop an attacker is to think like one.

The goal of this book is not to tell you everything about how to write secure web applications. Such a cover-it-all book would span thousands of pages, and be quite boring: it would contain lots of details on every web programming language out there, most of which you would never use. And it would contain lots of details on problems you will never try to solve. Every programming platform and every type of problem have their own gotchas.

The goal of this book is to make you aware that the code you write may be exploited, and that there are many pitfalls, regardless of which platform you use. Hopefully, you will see this book as a teaser, or a wake-up call, that will make you realize that the coding you do for a living is in fact a significant part of the security picture. If you end up being a little bit more paranoid when programming, this book has reached its goal.

0.1 The Rules

When reading the book, you'll come across a good handful of "rules" or "best practices". The rules highlight points that are particularly worthy of understanding or remembering. As with most other rules, these are not absolute. Some of the rules can be bent, others can be broken. Before you start

bending and breaking a rule, you should have a very clear understanding of the security problem the rule tries to prevent. And you should have an equally clear understanding of why your application will not be vulnerable, or why it doesn't matter if it is vulnerable, once you start bending and breaking the rule.

Deciding that an application will not be vulnerable is not necessarily a simple task. It's easy to think that "if I can't find a way to exploit my code, nobody else can". That view is extremely dangerous. The average developer is not trained in destructive thinking. She works by constructing things. There may always be an intruder that is more creative when it comes to malicious thinking than the developer is herself. To remember that, and at the same time see what the rules look like, we introduce the first rule:

Rule 1

Do not underestimate the power of the dark side

The rule encourages us not to take short cuts, and not to set a security mechanism aside, no matter what program we create and no matter what part of the program we are working on at the moment. It also tells us to be somewhat paranoid. This rule in itself is not particularly convincing, but paired with the contents of this book, it hopefully is. The Web has a dark side. Someone is out there looking for an opportunity to abuse a web site, either for fun or for profit. No matter what their intentions are, they may ruin the web site you have spent months creating. Even if they're not able to do direct harm, symptoms of poor security may give very bad press both for the web site and for the company that made it.

0.2 The Examples

This book contains lots and lots of examples. The author believes that next to experimenting, seeing examples is the best way to learn. In the security context, the two learning mechanisms don't always combine. Please do not use the examples in this book to experiment on sites on which you haven't got explicit permission to do so. Depending on the laws in your country, you may end up in jail.

Many of the examples will tell stories that make it seem as if they describe real life applications. And that's exactly what they do. The examples that

sound real are based on code reviews and testing done by various people, including the author. Some examples are even based on unauthorized, non-destructive experiments (luckily, I'm still not in jail). I have anonymized the sites by not mentioning their name, and often by showing pieces of code in another programming language than the site actually uses.

Examples are mainly small snippets of code written in Java, PHP, Perl or VBScript. These languages should be quite easy to read for most programmers. If you are new to one of these languages, you may find the following table useful. It lists a few syntactical differences:

	Java	PHP	Perl	VBScript
String concatenation	+	.	.	&
Variable prefix		$	$	
Subroutine prefix			&	
Line continuation				_

Domain names used in the examples follow the directions given in RFC 2606 [1]. None of them are valid in the real world. The IP addresses are private addresses according to RFC 1918 [2]. They are not valid on the Internet. (RFCs, short for Request For Comments, are technical and organizational documents about the Internet, maintained by the RFC Editor [3] on behalf of IETF [4], the Internet Engineering Task Force. Every official Internet protocol is defined in one or more RFCs.)

Note that some example text has had white space added for readability. Long URLs, error messages and text strings that would have been on a single line in their natural habitats, may span several lines in this book. And they do so without further notice.

0.3 The Chapters

Although this book is written with sequential reading of the entire text in mind, it should be possible to read single chapters as well. A chapter summary follows:

- Chapter 1 gives an introduction to HTTP and related web technologies, such as cookies and sessions, along with examples on what can go wrong if we fail to understand how it all works.

- Chapter 2 talks about metacharacter problems that may show up whenever we pass data to another system. The famous SQL Injection problem is described in great detail in this chapter.
- Chapter 3 addresses input handling such as spotting invalid input, how to deal with it, and why one should not blindly trust what comes from the client.
- Chapter 4 shows how data we send to our users' browsers may cause major trouble if left unfiltered. The Cross-site Scripting problem is described in this chapter.
- Chapter 5 explains how easy it may be to trick a user into performing a web task he never intended to do, just by pointing him to a web page or sending him an E-mail.
- Chapter 6 deals with password handling, secret identifiers and other things we may want to hide from the intruder. Includes the world's shortest introduction to cryptography.
- Chapter 7 discusses reasons why the code of web applications often ends up being insecure.
- Chapter 8 lists all the rules given throughout the book, including short summaries.
- Finally, there are appendixes on web server bugs, packet sniffing, E-mail forging, and sources of more information. Notorious appendix skippers should at least consider reading the "More Information" part.

The book also has a References chapter. Throughout the book, you'll see numbers in [angle brackets]. These numbers refer to entries in the References. The entries point to books, articles and web sites with more information on the topics discussed.

0.4 What is Not in This Book?

As this book is for programmers, most infrastructure security is left out. Also, security design, such as what authentication methods to use, how to separate logic in multiple tiers on multiple servers and so on is mostly missing. When coding, these decisions have already been made. Hopefully. If you're not only coding, but designing security mechanisms too, I urge you to read Ross Anderson's *Security Engineering* [5], which shows how easy it is to get things wrong (and how not to).

One important topic that should be high on the list of C/C++ coders is left out: the buffer overflow problem. This problem is hard to understand for people who are not seasoned C/C++ programmers. If you program C, C++ or any other language that lacks pointer checks, index checks and so on, make sure you fully understand the importance of protecting your memory areas. I suggest you take a look at Aleph One's classical article "Smashing the Stack for Fun and Profit" [6], or pick up a book on secure programming in general, which typically explains it all in great detail. I recommend *Building Secure Software* [7] by John Viega and Gary McGraw.

While talking about books on secure programming, I could also mention *Writing Secure Code* [8] by Michael Howard and David LeBlanc, and David Wheeler's on-line "Secure Programming for Linux and Unix HOWTO" [9]. Although the former is skewed towards the Microsoft platform and the latter favors Unix and Linux, both contain major parts that are relevant no matter what your platform is.

This book focuses on server-side programming. It does not address Java Applets, ActiveX objects and other technologies that allow programs to be run on the client-side. If you create client-side programs, you should understand that the program runs under full control of whoever operates the computer. It's probably also a good idea to read one of those books on general code security.

And finally, most platform-dependent security gotchas are left out to make the entire book readable for everyone. After reading this book, I urge you to spend some time browsing the Web for "security best practices" for your platform of choice.

0.5 A Note from the Author

You may like to know that I'm a web programmer myself. I've made my (far from neglectable) share of security holes, and even if I've spent every single day the last three years focusing only on such holes, I still make them. I like to think that I make fewer holes now than before, though. Not because I've become a better programmer, but because I've realized that every single line I write counts when it comes to security, and—even more importantly—that it's far too easy to make mistakes.

0.6 Feedback

If this book makes you angry, happy, curious, scared, nervous, comfortable, or anything, please tell me by sending an E-mail to `innocentcode@thathost.com`. If you find errors, please direct them to the same address. If you happen

to be in Oslo (the capitol of Norway) and want to discuss the topics of the book over a beer or something (I must warn you that beer is quite expensive in Norway), feel free to invite me. :-)

This book has a companion web site at `http://innocentcode.thathost.com/`. Any corrections or additions to the text will appear on this site.

This page intentionally left blank

1 The Basics

We don't have to go all the way back to the old Romans, but we'll step back to 1989–1990. That's when Tim Berners-Lee [10] and his friends at CERN "invented" the World Wide Web [11]. The Internet was already old [12], but with the birth of the Web, information was far more easily available.

Three specifications are central to the Web. One is the definition of URLs [13, 14, 15, 16], or Uniform Resource Locators, which specifies how to communicate, well, locations of resources (Standard documents usually refer to *URIs* [17, 16], Uniform Resource Identifiers, rather than URLs. URLs are a subset of URIs. This book will use the term URL even where standard documents mention URI, as most people think in terms of URLs.). Another specification is HTML [18], HyperText Markup Language, which gives us a way to structure textual information. And finally, there is HTTP [19], or Hypertext Transfer Protocol. HTTP tells us how nodes in the Web exchange information.

Most developers have good knowledge of URLs and HTML, but many know very little about HTTP. I truly believe that one needs a good understanding of the underlying infrastructure to be able to create more secure programs. This chapter will bring you up to speed on the basics of HTTP, and at the same time describe some security problems that may show up if one doesn't understand the basics.

1.1 HTTP

When a web browser wants to display a web page, it connects to the server mentioned in the URL to retrieve the page contents. As soon as the

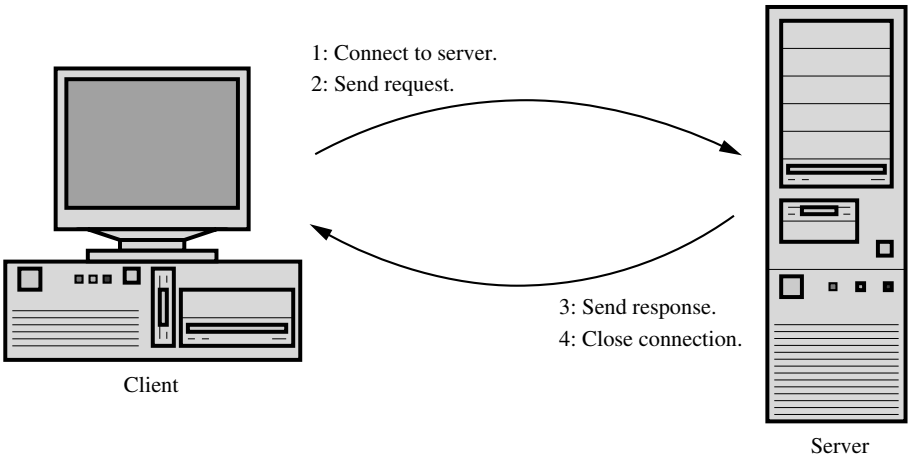

Figure 1.1 The client-server model of the web. The client connects and sends a request. The server responds and closes the connection

TCP connection is established, the browser sends a HTTP request asking the web server to provide the wanted document. The web server sends a reply containing the page contents, and closes the connection. If a *persistent connection* is used, the connection may remain open for some (normally short) time to allow multiple requests with less TCP overhead. Persistent connections typically speed up access to pages containing lots of images. If the document contains hypertext that references embedded contents, such as images and Java applets, the browser will need to send multiple requests to display all the contents.

The browser is always the initiating party—the server never "calls back". This means that HTTP is a *client/server protocol* (see Figure 1.1). The client will typically be a web browser, but it need not be. It may be any program capable of sending HTTP requests to a web server.

1.1.1 Requests and responses

HTTP is line oriented, just like many other Internet protocols. Communication takes place using strings of characters, separated by carriage return (ASCII 13) and line feed (ASCII 10). When you instruct your web browser to go to the URL `http://www.someplace.example/`, it will look up the IP address of the host named `www.someplace.example`, connect to it, and send the