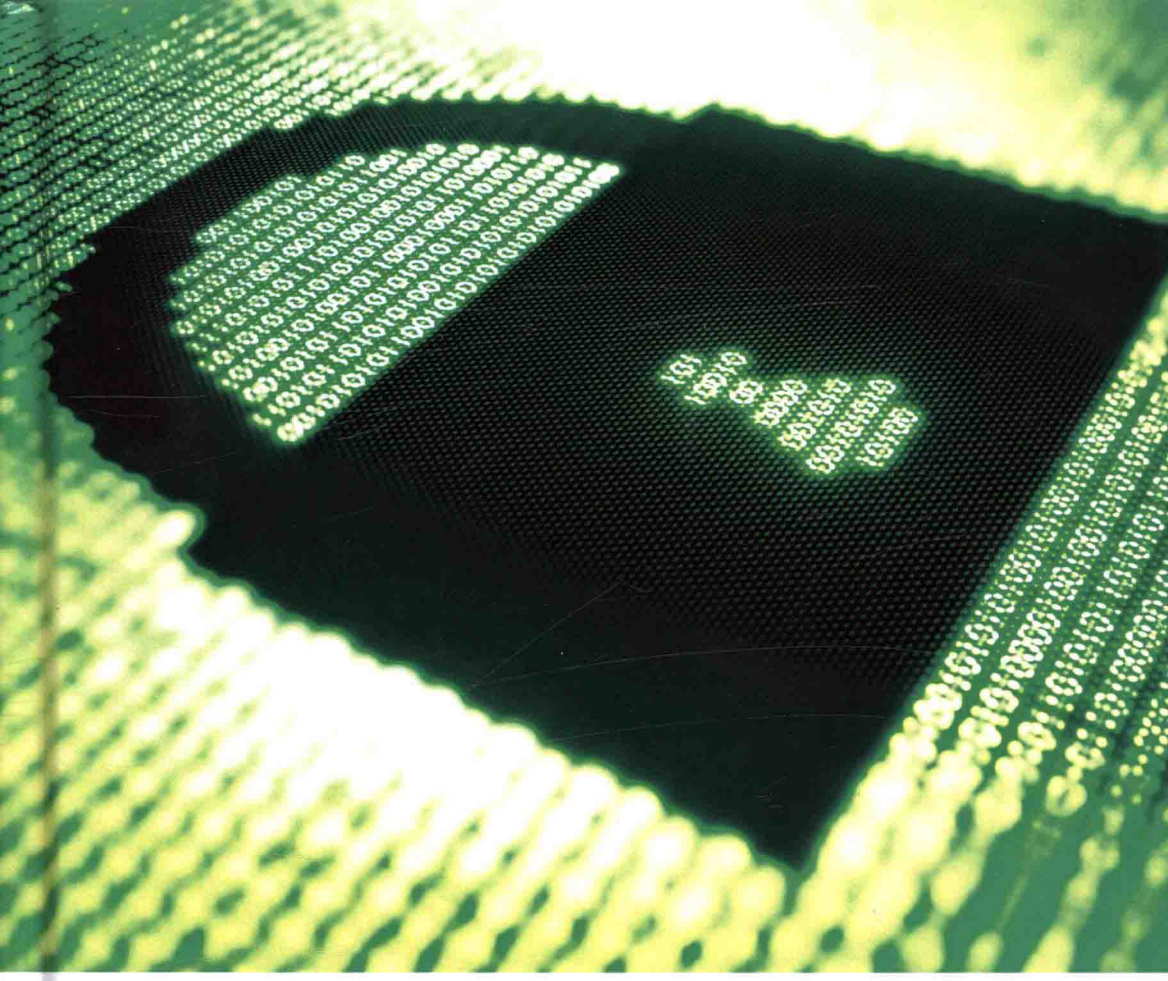

Safety of Web Applications

Éric Quinton

Risks, Encryption and Handling Vulnerabilities with PHP

ISTE PRESS

ELSEVIER

Web applications are prime targets for hackers. If a site has poor security, it is more susceptible to hacking, which could lead to sensitive information being leaked. Establishing appropriate security involves first analyzing risk, which consists of an evaluation of information confidentiality, integrity and availability along with a potential threat analysis, should security be breached.

An application must be structured correctly and the Model, View, Controller (MVC) design provides a good example to apply to the site architecture. This type of protection is based on correctly configured servers and encryption.

This book clearly explains how to test software security prior to going online as well as a comprehensive overview of the most common cyber-attacks and how to protect sites against them using PHP. Other sections include user information, rights management, encryption principles and advanced mechanisms to monitor completed actions.

Éric Quinton is a database administrator and responsible for the security of information systems at the National Research Institute of Science and technology for Environment and Agriculture in France.

ELSEVIER

ISTE PRESS

www.iste.co.uk

9 781785 482281

Éric Quinton

Safety of Web Applications

Series Editor
Jean-Charles Pomerol

Safety of Web Applications

Risks, Encryption and Handling Vulnerabilities with PHP

Éric Quinton

ISTE PRESS

ELSEVIER

First published 2017 in Great Britain and the United States by ISTE Press Ltd and Elsevier Ltd

ISTE Press Ltd
27-37 St George's Road
London SW19 4EU
UK

www.iste.co.uk

Elsevier Ltd
The Boulevard, Langford Lane
Kidlington, Oxford, OX5 1GB
UK

www.elsevier.com

Notices

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

British Library Cataloguing-in-Publication Data
A CIP record for this book is available from the British Library
Library of Congress Cataloging in Publication Data
A catalog record for this book is available from the Library of Congress
ISBN 978-1-78548-228-1

Safety of Web Applications

Preface

When I first began my career in the 1980s, personal computing was in its infancy. The first computers had very little internal memory, so files and even entire booting routines were stored on floppy disks with very low storage capacity. Graphical interfaces did not yet exist, or in some cases were just beginning to be introduced. To exchange information with other users, we used floppy disks, and more powerful computer systems used magnetic tapes. Mainframe computers were accessed by passive terminals connected together with dedicated cables (one cable per terminal). These were called *serial links*.

During the 1990s, as computing power increased, direct wired connections were gradually replaced by networks, and terminals were replaced by PCs. Computers then began to start dialoguing with each other internally, within companies. However, exchanges between different locations were limited, since communication costs were exorbitant and only very low bandwidths were possible. Applications were written for specific operating systems, which were in perfect control of their environments: all monitors had the same characteristics, as did all computers (PCs).

In the late 1990s, the development of the Internet and the creation of the first websites open to the general public profoundly changed the lay of the land. The shift from dedicated applications on specific devices to the more general form of web applications deeply disrupted approaches to programming. Now, anyone can connect to any software, access information, order products, manage their emails, and so on.

In the 2000s, with the proliferation of wireless connections (WiFi only really began to become widespread in France in the mid-2000s), the developer community also began to become aware of the security problems associated with this new environment. Before then, problems had been relatively well-isolated, since IT professionals were rare and also because systems enjoyed perfect control of their technical environments. This does not mean that security was completely neglected (encryption has always existed), but rather that, as a general rule, only companies or organizations that handled sensitive information (banks, military, etc.) paid any special attention to security. For most companies, the risk of data loss was very often the only factor taken into consideration, and even then only very loose protective measures were taken. The identify of users was also typically verified using unencrypted logins and passwords, since the risk of this information being intercepted was still relatively low. The available computer equipment was not yet powerful enough to implement "real-time" encryption anyway.

But individuals, and in some cases organizations, quickly realized the potential financial and later political advantages that could be obtained by targeting these web applications: the Internet was no longer static as in its early days (initially, web pages could not be modified), but had now become dynamic. Now, users themselves can request specific behavior from the server, and influence the information that is presented to them.

Web applications thus acquired a new dimension of risk: the information submitted by the browser needs to be systematically checked. Unanticipated behavior can occur if the user sends data specially modified with the objective of causing the server to react differently. This is, for example, the case with SQL injection, which, without the appropriate security measures, tricks the server into executing operations that were not planned by the programmer.

Today, web software development requires an extremely varied spectrum of skills (programming languages such as PHP, HTML for drawing pages, CSS for stylizing them uniquely, JavaScript and its extensions such as JQuery for building interactivity, SQL for accessing databases, etc.).

Many programmers learn these languages without necessarily being made aware of the risks to which they are exposed. Basic training rarely focuses on these aspects (with the exception of specialized training), or only addresses some of them, preferring to focus on more rudimentary skills. It is all too

common to meet newly qualified developers with very little knowledge of how to secure their software. This is also often true with self-taught students, who learn to program to meet a specific need, such as setting up a website for an organization, or who write interfaces to control small databases.

Yet the risks are ever-present: without special protective measures, data can be corrupted, and the website can be used as a gateway by an attacker seeking to take control of the company information system, or might be defaced for politically motivated reasons.

Taking the time to examine the security of an application is therefore particularly important. The first step of this process is to perform risk assessment: a banking application is more sensitive than a system for booking meeting rooms.

The next step is to consider the runtime environment of the program. This might not be intrinsically safe: if the machines that host the application are not well protected themselves, any protective measures built into the code will be essentially useless. Similarly, exchanges between the server and users must be encrypted to prevent undesirable eavesdropping, or malicious modifications to the exchanged information performed in real time. Of course, these considerations begin to exceed the scope of just the program code, but one should note that encryption mechanisms are widely used by a large number of routines, in particular for the guarantees that they can provide for certain types of operation.

Understanding the risks to which the code is exposed is a complex topic, simply due to the sheer variety of the potential mechanisms by which it could be attacked. Fortunately, there exist projects such as OWASP that regularly publish lists of the most common types of attack, each of which can often be thwarted by a few lines of code. ANSSI, the French National Agency for Information Systems Security, also regularly publishes advice and risk analysis methodology in the form of the EBIOS method.

One important aspect of security is managing users and their access rights. Several different mechanisms can be used to identify users and allow them to access the information that they need.

Designing an application is a complex task, and the only way to ensure that security measures are correctly applied everywhere is to structure the code

accordingly. Organizational methods, such as the MVC model (model, view, controller), can be used to fulfill these needs, and ensure that the application operates both reliably and securely.

Finally, testing the developed software with special tools will allow us to identify the most obvious shortcomings. This is an important step before beginning production, and provides users with a guarantee that the necessary precautions have been taken to anticipate risks.

This book presents a large number of examples. Most of them are written in PHP, which is one of the most common languages for creating web applications. These examples can of course be adapted to fit other languages. Often, only a few lines of code is required to patch a vulnerability, and the algorithm or approach used to tackle the problem is more interesting than the actual code itself.

Éric QUINTON
February 2017

Contents