Contents

Preface xi

Chapter 1. Why Do Web Applications Need to be Secure? . . 1

1.1. What is a web application? 1
1.1.1. The Internet, a global network 1
1.1.2. Programs before the web 2
1.1.3. Web technology is gradually adopted by applications 3
1.1.4. Exchange is based on trust 4
1.1.5. Bad idea: trusting that the intranet is automatically secure 6
1.2. What is computer security? 6
1.2.1. Security relies on many different blocks 7
1.2.2. Not all applications are equal in terms of security needs . . 9
1.3. Examples of damage caused by security failures 10
1.3.1. Do not take anything for granted 13
1.3.2. Well-structured applications are easier to secure 14
1.3.3. The only type of security that matters is global security . . . 15
1.3.4. What security measures are required by applications with heavy clients? 16

Chapter 2. Estimating Risk 19

2.1. What is risk? 19
2.2. How can we protect ourselves from risk? 20
2.3. Determining the target 21
2.4. Determining the impact 22

2.4.1. Confidentiality 23
2.4.2. Integrity 24
2.4.3. Availability 26
2.4.4. Determining the level of risk associated with a project . . . 27
2.5. Which causes or scenarios should be considered? 29
2.5.1. ASVS requirements 30
2.5.2. Determining the relevant causes and their likelihoods of occurrence 32
2.5.3. Choosing the level of requirements 32
2.6. How should this study be performed in a company setting? . . . 33

Chapter 3. Encryption and Web Server Configuration 35

3.1. Examples of different web servers 35
3.2. Introduction to concepts in encryption 36
3.2.1. Symmetric encryption 36
3.2.2. Computing hashes and salting passwords 38
3.2.3. Asymmetric encryption 40
3.2.4. What is the ideal length for encryption keys? 42
3.2.5. Digital certificates and the chain of certification 43
3.3. Generating and managing encryption certificates 44
3.3.1. The OpenSSL library 44
3.3.2. Different types of certificates 45
3.3.3. Generating certificates 46
3.3.4. Where are keys and certificates stored? 49
3.3.5. Commands for viewing keys and certificates 50
3.4. Implementing the HTTPS protocol 52
3.4.1. Understanding the HTTPS protocol 52
3.4.2. Implementing the HTTPS protocol 54
3.4.3. Testing the SSL chain 55
3.5. Improving the security of the Apache server 56
3.5.1. Ensuring that the server hosting Apache has the latest security updates 56
3.5.2. Prohibiting low-security protocols 57
3.5.3. Preventing request flooding 58
3.5.4. Implementing a request filter 60
3.5.5. Allowing page header modifications 61
3.5.6. Authorizing .htaccess files 62

3.5.7. Hiding the version information of Apache and PHP 63
3.6. In summary 63

Chapter 4. Threats and Protecting Against Them 65

4.1. The threats associated with web-based environments 66
4.1.1. Limiting the types of authorized request 66
4.1.2. Preventing users from browsing the website file system . . . 67
4.1.3. Limiting the risk of session cookie hijacking 67
4.1.4. Hiding error messages 68
4.1.5. Asking browsers to enable safeguards 68
4.2. The top 10 most frequent attacks in 2013 70
4.2.1. Code injection 70
4.2.2. Circumventing the login process and session hijacking . . . 78
4.2.3. Executing code to redirect to another website, or Cross Site Scripting (XSS) 81
4.2.4. Insecure direct object references 84
4.2.5. Poorly configured application or environment security . . . 88
4.2.6. Leaking sensitive information 88
4.2.7. Lack of access-level control for certain functions 91
4.2.8. Tricking users into unknowingly running legitimate commands 92
4.2.9. Using components with known vulnerabilities 93
4.2.10. Refusing redirects 94
4.3. Other countermeasures 94
4.3.1. Checking UTF-8 encoding 94
4.3.2. Analyzing uploaded documents with an antivirus 96
4.3.3. Preventing the browser from storing the login and password 102
4.3.4. Encrypting database access 104
4.4. Implementing a resource controller 108
4.4.1. Managing user connections 109
4.4.2. Monitoring behavior 112
4.4.3. Managing alerts 117

Chapter 5. Managing User Logins and Assigning Permissions 119

5.1. Managing user logins 119
5.1.1. Managing accounts in a database 120

5.1.2. Locking passwords 131
5.1.3. Retrieving the login from the company directory 132
5.1.4. Delegating the login process to a CAS server 135
5.1.5. Doing more with CAS: identity federations with Shibboleth 143
5.1.6. Managing login offline using database storage 144
5.1.7. Managing the login process using a token encrypted with asymmetric keys 150
5.1.8. Creating tokens with the JWT protocol 160
5.1.9. Using the OAuth protocol to generate tokens 163
5.2. Managing permissions 164
5.2.1. What should we protect? 164
5.2.2. Managing user permissions with LDAP directory groups 169
5.2.3. Managing user permissions based on groups defined in the application 171
5.3. In summary 176

Chapter 6. Using the MVC Model to Structure the Application 177

6.1. Why does the application structure matter? 177
6.2. What is the MVC model? 178
6.2.1. Model 179
6.2.2. View 180
6.2.3. Controller 183
6.3. Conclusion 187

Chapter 7. Implementing a Suitable Technical Platform and Testing the Application 189

7.1. Designing a suitable technical architecture 189
7.1.1. Integrating security into the earliest stages of the project 189
7.1.2. Using code management systems such as GIT 190
7.1.3. Using software to design the database 191
7.1.4. Implementing separate architectures for development and production 192
7.2. Testing the security of the application 194

7.2.1. Analyzing vulnerabilities with ZAP Proxy 194
7.2.2. Certifying the application 199
7.2.3. Write the implementation documents 199
7.3. What options do we have if implementing security measures for an application seems an impossible task? 200

Bibliography 203

Index 207