# Contents

Preface xvii

How This Book Is Organized xviii

About the Authors xxi

Acknowledgments xxiii

Chapter 1 How Does Software Fail Thee? Let Us Count the Ways 1

1.1 Vulnerabilities Abound 2

1.1.1 Security Flaws Are Omnipresent 3

1.1.2 Cars Have Their Share of Computer Problems Too 5

1.2 Tracing the Roots of Defective Software 7

1.3 What Are the True Costs of Insecure Software to Global Enterprises? 8

1.4 Addressing Security Questions Addresses Resilience 10

1.5 References 11

Chapter 2 Characteristics of Secure and Resilient Software 13

2.1 Functional Versus Nonfunctional Requirements 13

2.2 Testing Nonfunctional Requirements 15

2.3 Families of Nonfunctional Requirements 17

2.4 Availability 18

2.5 Capacity 19

2.6 Efficiency 20

2.7 Interoperability 20

2.8 Manageability 21

2.9 Cohesion 21

2.10 Coupling 22

2.11 Maintainability 22
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.12</td>
<td>Performance</td>
<td>23</td>
</tr>
<tr>
<td>2.13</td>
<td>Portability</td>
<td>23</td>
</tr>
<tr>
<td>2.14</td>
<td>Privacy</td>
<td>24</td>
</tr>
<tr>
<td>2.15</td>
<td>Recoverability</td>
<td>25</td>
</tr>
<tr>
<td>2.16</td>
<td>Reliability</td>
<td>26</td>
</tr>
<tr>
<td>2.17</td>
<td>Scalability</td>
<td>27</td>
</tr>
<tr>
<td>2.18</td>
<td>Security</td>
<td>27</td>
</tr>
<tr>
<td>2.19</td>
<td>Serviceability/Supportability</td>
<td>29</td>
</tr>
<tr>
<td>2.20</td>
<td>Characteristics of Good Requirements</td>
<td>30</td>
</tr>
<tr>
<td>2.21</td>
<td>Eliciting Nonfunctional Requirements</td>
<td>32</td>
</tr>
<tr>
<td>2.22</td>
<td>Documenting Nonfunctional Requirements</td>
<td>33</td>
</tr>
<tr>
<td>2.23</td>
<td>References</td>
<td>34</td>
</tr>
</tbody>
</table>

### Chapter 3 Security and Resilience in the Software Development Life Cycle

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Resilience and Security Begin from Within</td>
<td>37</td>
</tr>
<tr>
<td>3.2</td>
<td>Requirements Gathering and Analysis</td>
<td>39</td>
</tr>
<tr>
<td>3.3</td>
<td>Systems Design and Detailed Design</td>
<td>40</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Functional Decomposition</td>
<td>41</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Categorizing Threats</td>
<td>41</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Ranking Threats</td>
<td>42</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Mitigation Planning</td>
<td>42</td>
</tr>
<tr>
<td>3.4</td>
<td>Design Reviews</td>
<td>42</td>
</tr>
<tr>
<td>3.5</td>
<td>Development (Coding) Phase</td>
<td>43</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Static Analysis</td>
<td>43</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Peer Review</td>
<td>44</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Unit Testing</td>
<td>44</td>
</tr>
<tr>
<td>3.6</td>
<td>Testing</td>
<td>44</td>
</tr>
<tr>
<td>3.7</td>
<td>Deployment</td>
<td>45</td>
</tr>
<tr>
<td>3.8</td>
<td>Security Training</td>
<td>47</td>
</tr>
<tr>
<td>3.9</td>
<td>References</td>
<td>48</td>
</tr>
</tbody>
</table>

### Chapter 4 Proven Best Practices for Resilient Applications

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Critical Concepts</td>
<td>49</td>
</tr>
<tr>
<td>4.2</td>
<td>The Security Perimeter</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>Attack Surface</td>
<td>52</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Mapping the Attack Surface</td>
<td>53</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Side Channel Attacks</td>
<td>54</td>
</tr>
<tr>
<td>4.4</td>
<td>Application Security and Resilience Principles</td>
<td>54</td>
</tr>
<tr>
<td>4.5</td>
<td>Practice 1: Apply Defense in Depth</td>
<td>55</td>
</tr>
<tr>
<td>4.6</td>
<td>Practice 2: Use a Positive Security Model</td>
<td>56</td>
</tr>
</tbody>
</table>
Chapter 5  Designing Applications for Security and Resilience 65

5.1 Design Phase Recommendations 65
5.1.1 Misuse Case Modeling 66
5.1.2 Security Design and Architecture Review 68
5.1.3 Threat and Risk Modeling 68
5.1.4 Risk Analysis and Modeling 70
5.1.5 Security Requirements and Test Case Generation 70

5.2 Design to Meet Nonfunctional Requirements 71
5.3 Design Patterns 73
5.4 Architecting for the Web 74
5.5 Architecture and Design Review Checklist 76
5.6 References 84

Chapter 6  Programming Best Practices 87

6.1 The Evolution of Software Attacks 88
6.2 The OWASP Top 10 89
6.2.1 A1: Injection 90
6.2.2 A2: Cross-Site Scripting 90
6.2.3 A3: Broken Authentication and Session Management 90
6.2.4 A4: Insecure Direct Object References 91
6.2.5 A5: Cross-Site Request Forgery 91
6.2.6 A6: Security Misconfiguration 91
6.2.7 A7: Failure to Restrict URL Access 92
8.8.1 IDE Integration for Developers 181  
8.8.2 Build Integration for Governance 182  
8.9 Regulatory Compliance 183  
8.10 Benefits of Using Source Code Analyzers 183  
8.11 Penetration (Pen) Testing 184  
8.11.1 Penetration Testing Tools 185  
8.11.2 Automated Black Box Scanning 185  
8.11.3 Deployment Strategy 186  
8.11.4 Gray Box Testing 187  
8.11.5 Limitations and Constraints of Pen Testing Tools 188  
8.12 References 189  

Chapter 9 Testing Commercial off-the-Shelf Systems 191  
9.1 The Problems with Shrink-Wrapped Software 191  
9.2 The Common Criteria for Information Technology Security Evaluation 192  
9.2.1 Harmonizing Evaluation Criteria 194  
9.2.2 Development 196  
9.2.3 Evaluation 197  
9.2.4 Operation 197  
9.2.5 Key Concepts of the Common Criteria 197  
9.2.6 The Security Framework 198  
9.2.7 The Common Criteria Approach 199  
9.2.8 The Security Environment 200  
9.2.9 The Common Criteria Portal 208  
9.2.10 Criticisms of the CC 208  
9.3 The Commercial Community Responds 210  
9.3.1 The BITS/FSTC Security Assurance Initiative 210  
9.4 ICSA Labs 211  
9.4.1 Evaluation Methodology 212  
9.4.2 Certification Criteria 212  
9.4.3 ICSA Labs Testing and Certification Process 212  
9.5 Veracode’s VerAfied Software Assurance 214  
9.5.1 Ratings Methodology 214  
9.5.2 Assessing Software for the VerAfied Mark 214  
9.6 References 216
Chapter 10  Implementing Security and Resilience Using CLASP  217

10.1 Comprehensive, Lightweight Application Security Process (CLASP)  218
10.2 CLASP Concepts  218
10.3 Overview of the CLASP Process  219
10.4 CLASP Key Best Practices  221
  10.4.1 Best Practice 1: Institute Awareness Programs  221
  10.4.2 Best Practice 2: Perform Application Assessments  223
  10.4.3 Best Practice 3: Capture Security Requirements  224
  10.4.4 Best Practice 4: Implement Secure Development Practices  225
  10.4.5 Best Practice 5: Build Vulnerability Remediation Procedures  226
  10.4.6 Best Practice 6: Define and Monitor Metrics  226
  10.4.7 Best Practice 7: Publish Operational Security Guidelines  227
10.5 CLASP Security Activities to Augment Software Development Processes  227
10.6 Applying CLASP Security Activities to Roles  228
10.7 Re-engineering Your SDLC for CLASP  232
  10.7.1 Business Objectives  232
  10.7.2 Process Milestones  232
  10.7.3 Process Evaluation Criteria  232
  10.7.4 Forming the Process Re-engineering Team  233
10.8 Sample CLASP Implementation Roadmaps  234
  10.8.1 Green-Field Roadmap  235
  10.8.2 Legacy Roadmap  235
10.9 References  236

Chapter 11  Metrics and Models for Security and Resilience Maturity  237

11.1 Maturity Models for Security and Resilience  237
11.2 Software Assurance Maturity Model—OpenSAMM  238
  11.2.1 Core Practice Areas  240
  11.2.2 Levels of Maturity  241
  11.2.3 Assurance  243
11.3 The Building Security In Maturity Model (BSIMM) 247
   11.3.1 BSIMM Software Security Framework 248
11.4 BSIMM Activities
   11.4.1 Governance: Strategy and Metrics 250
   11.4.2 Governance: Compliance and Policy 254
   11.4.3 Governance: Training 258
   11.4.4 Intelligence: Attack Models 261
   11.4.5 Intelligence: Security Features and Design 265
   11.4.6 Intelligence: Standards and Requirements 267
   11.4.7 SSDL Touchpoints: Architecture Analysis 271
   11.4.8 SSDL Touchpoints: Code Review 274
   11.4.9 SSDL Touchpoints: Security Testing 277
   11.4.10 Deployment: Penetration Testing 280
   11.4.11 Deployment: Software Environment 282
   11.4.12 Deployment: Configuration Management and Vulnerability Management 284
11.5 Measuring Results with BSIMM 286
11.6 Helpful Resources For Implementing BSIMM 287
11.7 Applying BSIMM to the Financial Services Domain 288
   11.7.1 Working Group Methodology 288
11.8 References 289

Chapter 12 Taking It to the Streets 291
12.1 Getting Educated 291
   12.1.1 DEVELOPER 530: Defending Web Applications 292
   12.1.2 DEVELOPER 530: Essential Secure Coding in Java/JEE 293
   12.1.3 DEVELOPER 541: Secure Coding in Java/JEE: Developing Defensible Applications 293
   12.1.4 DEVELOPER 542: Web App Penetration Testing and Ethical Hacking 293
   12.1.5 DEVELOPER 544: Secure Coding in .NET: Developing Defensible Applications 294
12.1.6 DEVELOPER 545: Secure Coding in PHP: Developing Defensible Applications 294
12.1.7 DEVELOPER 534: Secure Code Review for Java Web Apps 295
12.1.8 DEVELOPER 543: Secure Coding in C/C++: Developing Defensible Applications 295
12.1.9 Aspect Security Inc. 296
12.1.10 CERT Software Engineering Institute (SEI) 298
12.1.11 SEI Secure Coding in C and C++ Course 299
12.2 Getting Certified 300
12.2.1 Certified Secure Software Lifecycle Professional (CSSLP) 301
12.2.2 Why Obtain the CSSLP? 302
12.2.3 Benefits of Certification to the Professional 302
12.2.4 Benefits of Certification to the Enterprise 302
12.3 Getting Involved 303
12.3.1 Web Application Security Consortium 303
12.4 Reaching Out for Research 308
12.4.1 DHS Research Program Areas 308
12.4.2 The U.S. Treasury and the FSSCC 310
12.5 Last Call 314
12.6 Conclusion 315
12.7 References 316

Glossary 319

Appendix A 2010 CWE/SANS Top 25 Most Dangerous Programming Errors 335
A.1 Brief Listing of the Top 25 336
  A.1.1 Insecure Interaction Between Components 336
  A.1.2 Risky Resource Management 336
  A.1.3 Porous Defenses 337
A.2 Detailed CWE Descriptions 338
A.2.1 CWE-79: Failure to Preserve Web Page Structure ("Cross-Site Scripting") 338
A.2.2 CWE-89: Improper Sanitization of Special Elements Used in an SQL Command ("SQL Injection") 338
A.2.3 CWE-120: Buffer Copy Without Checking Size of Input ("Classic Buffer Overflow") 339
A.2.4 CWE-352: Cross-Site Request Forgery (CSRF) 339
A.2.5 CWE-285: Improper Access Control (Authorization) 339
A.2.6 CWE-807: Reliance on Un-trusted Inputs in a Security Decision 340
A.2.7 CWE-22: Improper Limitation of a Pathname to a Restricted Directory ("Path Traversal") 340
A.2.8 CWE-434: Unrestricted Upload of File with Dangerous Type 340
A.2.9 CWE-78: Improper Sanitization of Special Elements Used in an OS Command ("OS Command Injection") 341
A.2.10 CWE-311: Missing Encryption of Sensitive Data 341
A.2.11 CWE-798: Use of Hard-Coded Credentials 341
A.2.12 CWE-805: Buffer Access with Incorrect Length Value 342
A.2.13 CWE-98: Improper Control of Filename for Include/Require Statement in PHP Program ("PHP File Inclusion") 342
A.2.14 CWE-129: Improper Validation of Array Index 342
A.2.15 CWE-754: Improper Check for Unusual or Exceptional Conditions 342
A.2.16 CWE-209: Information Exposure Through an Error Message 343
A.2.17 CWE-190: Integer Overflow or Wraparound 343
A.2.18 CWE-131: Incorrect Calculation of Buffer Size 343
A.2.19 CWE-306: Missing Authentication for Critical Function 344
A.2.20 CWE-494: Download of Code Without Integrity Check 344
A.2.21 CWE-732: Incorrect Permission Assignment for Critical Resource 344
A.2.22 CWE-770: Allocation of Resources Without Limits or Throttling 344
A.2.23 CWE-601: URL Redirection to Site ("Open Redirect") 345
A.2.24 CWE-327: Use of a Broken or Risky Cryptographic Algorithm 345
A.2.25 CWE-362: Race Condition 345

Appendix B Enterprise Security API 347

B.1 Interface Encoder 348
B.2 Interface User 349
B.3 Interface Authenticator 350
B.4 Interface AccessController 351
B.5 Interface AccessReferenceMap 352
B.6 Interface Encryptor 355
B.7 Interface HTTPUtilities 355
B.8 Interface Logger 357

Index 361