



Contents

Foreword **xix**

Preface **xxiii**

Acknowledgments **xxxi**

About the Author **xxxv**

Part I: Software Security Fundamentals **1**

1 **Defining a Discipline** **3**

The Security Problem 4

The Trinity of Trouble: Why the Problem Is Growing 5

Basic Science 10

Security Problems in Software 14

Bugs and Flaws and Defects, Oh My! 14

The Range of Defects 18

The Problem with Application Security 20

Software Security and Operations 23

Solving the Problem: The Three Pillars of Software Security 25

Pillar I: Applied Risk Management 26

Pillar II: Software Security Touchpoints 27

Pillar III: Knowledge 35

The Rise of Security Engineering 37

Software Security Is Everyone's Job 38

2	A Risk Management Framework	39
	Putting Risk Management into Practice	40
	How to Use This Chapter	41
	The Five Stages of Activity	42
	<i>Stage 1: Understand the Business Context</i>	43
	<i>Stage 2: Identify the Business and Technical Risks</i>	43
	<i>Stage 3: Synthesize and Rank the Risks</i>	44
	<i>Stage 4: Define the Risk Mitigation Strategy</i>	45
	<i>Stage 5: Carry Out Fixes and Validate</i>	45
	<i>Measuring and Reporting on Risk</i>	46
	The RMF Is a Multilevel Loop	46
	Applying the RMF: KillerAppCo's iWare 1.0 Server	48
	<i>Understanding the Business Context</i>	49
	<i>Identifying the Business and Technical Risks</i>	50
	<i>Synthesizing and Ranking the Risks</i>	63
	<i>Defining the Risk Mitigation Strategy</i>	69
	<i>Carrying Out Fixes and Validating</i>	73
	The Importance of Measurement	73
	<i>Measuring Return</i>	74
	<i>Measurement and Metrics in the RMF</i>	75
	The Cigital Workbench	76
	Risk Management Is a Framework for Software Security	79

Part II: Seven Touchpoints for Software Security **81**

3	Introduction to Software Security Touchpoints	83
	Flyover: Seven Terrific Touchpoints	86
	1. <i>Code Review (Tools)</i>	86
	2. <i>Architectural Risk Analysis</i>	86
	3. <i>Penetration Testing</i>	87
	4. <i>Risk-Based Security Testing</i>	87
	5. <i>Abuse Cases</i>	88
	6. <i>Security Requirements</i>	88
	7. <i>Security Operations</i>	88
	*. <i>External Analysis</i>	88
	<i>Why Only Seven?</i>	89
	Black and White: Two Threads Inextricably Intertwined	89
	Moving Left	91
	Touchpoints as Best Practices	94

Who Should Do Software Security?	96
<i>Building a Software Security Group</i>	97
Software Security Is a Multidisciplinary Effort	100
Touchpoints to Success	103

4 Code Review with a Tool 105

Catching Implementation Bugs Early (with a Tool)	106
Aim for Good, Not Perfect	108
Ancient History	109
Approaches to Static Analysis	110
<i>The History of Rule Coverage</i>	112
<i>Modern Rules</i>	114
Tools from Researchland	114
Commercial Tool Vendors	123
<i>Commercial Source Code Analyzers</i>	124
<i>Key Characteristics of a Tool</i>	125
<i>Three Characteristics to Avoid</i>	127
<i>The Fortify Source Code Analysis Suite</i>	127
<i>The Fortify Knowledge Base</i>	132
<i>Using Fortify</i>	134
Touchpoint Process: Code Review	135
Use a Tool to Find Security Bugs	137

5 Architectural Risk Analysis 139

Common Themes among Security Risk Analysis Approaches	140
Traditional Risk Analysis Terminology	144
Knowledge Requirement	147
The Necessity of a Forest-Level View	148
A Traditional Example of a Risk Calculation	152
Limitations of Traditional Approaches	153
Modern Risk Analysis	154
<i>Security Requirements</i>	155
<i>A Basic Risk Analysis Approach</i>	156
Touchpoint Process: Architectural Risk Analysis	161
<i>Attack Resistance Analysis</i>	163
<i>Ambiguity Analysis</i>	165
<i>Weakness Analysis</i>	167
Getting Started with Risk Analysis	169
Architectural Risk Analysis Is a Necessity	170

-
- 6 Software Penetration Testing 171**
- Penetration Testing Today 173
 - Software Penetration Testing—a Better Approach 178
 - Make Use of Tools* 179
 - Test More Than Once* 182
 - Incorporating Findings Back into Development 183
 - Using Penetration Tests to Assess the Application Landscape 184
 - Proper Penetration Testing Is Good 185
- 7 Risk-Based Security Testing 187**
- What’s So Different about Security? 191
 - Risk Management and Security Testing 192
 - How to Approach Security Testing 193
 - Who* 193
 - How* 194
 - Thinking about (Malicious) Input 201
 - Getting Over Input 203
 - Leapfrogging the Penetration Test 204
- 8 Abuse Cases 205**
- Security Is Not a Set of Features 209
 - What You Can’t Do 210
 - Creating Useful Abuse Cases 211
 - But No One Would Ever Do That!* 212
 - Touchpoint Process: Abuse Case Development 213
 - Creating Anti-Requirements* 213
 - Creating an Attack Model* 216
 - An Abuse Case Example 217
 - Abuse Cases Are Useful 222
- 9 Software Security Meets Security Operations 223**
- Don’t Stand So Close to Me 224
 - Kumbaya (for Software Security) 225
 - Come Together (Right Now) 232
 - Future’s So Bright, I Gotta Wear Shades 235

Part III: Software Security Grows Up 237

10 An Enterprise Software Security Program 239

- The Business Climate 240
- Building Blocks of Change 242
- Building an Improvement Program 246
- Establishing a Metrics Program 247
 - A Three-Step Enterprise Rollout* 248
- Continuous Improvement 250
- What about COTS (and Existing Software Applications)? 251
 - An Enterprise Information Architecture* 253
- Adopting a Secure Development Lifecycle 256

11 Knowledge for Software Security 259

- Experience, Expertise, and Security 261
- Security Knowledge: A Unified View 262
- Security Knowledge and the Touchpoints 268
- The Department of Homeland Security Build
 - Security In Portal 269
- Knowledge Management Is Ongoing 274
- Software Security Now 275

12 A Taxonomy of Coding Errors 277

- On Simplicity: Seven Plus or Minus Two 279
 - Input Validation and Representation* 279
 - API Abuse* 279
 - Security Features* 280
 - Time and State* 280
 - Error Handling* 281
 - Code Quality* 281
 - Encapsulation* 281
 - Environment* 282
- The Phyla 282
 - More Phyla Needed* 289
- A Complete Example 290
- Lists, Piles, and Collections 292
 - Nineteen Sins Meet Seven Kingdoms* 296
 - Seven Kingdoms and the OWASP Ten* 297
- Go Forth (with the Taxonomy) and Prosper 297

13	Annotated Bibliography and References	299
	Annotated Bibliography: An Emerging Literature	299
	<i>Required Reading: The Top Five</i>	299
	<i>References Cited in Software Security: Building</i>	
	<i>Security In</i>	300
	<i>Government and Standards Publications Cited</i>	312
	<i>Other Important References</i>	313
	Software Security Puzzle Pieces	318
	<i>Basic Science: Open Research Areas</i>	319

Appendices 321

A	Fortify Source Code Analysis Suite Tutorial	323
	1. Introducing the Audit Workbench	324
	2. Auditing Source Code Manually	326
	3. Ensuring a Working Build Environment	328
	4. Running the Source Code Analysis Engine	329
	5. Exploring the Basic SCA Engine Command Line	
	Arguments	332
	6. Understanding Raw Analysis Results	333
	7. Integrating with an Automated Build Process	335
	8. Using the Audit Workbench	339
	9. Auditing Open Source Applications	342
B	ITS4 Rules	345
C	An Exercise in Risk Analysis: Smurfware	385
	SmurfWare SmurfScanner Risk Assessment Case Study	385
	SmurfWare SmurfScanner Design for Security	390
D	Glossary	393
	Index	395