

# **BUG BOUNTY BOOTCAMP**

**The Guide to Finding and  
Reporting Web Vulnerabilities**

**Vickie Li**



**no starch  
press**

San Francisco

**BUG BOUNTY BOOTCAMP.** Copyright © 2021 by Vickie Li.

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13: 978-1-7185-0154-6 (print)

ISBN-13: 978-1-7185-0155-3 (ebook)

Publisher: William Pollock

Production Manager: Rachel Monaghan

Production Editors: Miles Bond and Dapinder Dosanjh

Developmental Editor: Frances Saux

Cover Design: Rick Reese

Interior Design: Octopod Studios

Technical Reviewer: Aaron Guzman

Copyeditor: Sharon Wilkey

Compositor: Jeff Lytle, Happenstance Type-O-Rama

Proofreader: James Fraleigh

For information on book distributors or translations, please contact No Starch Press, Inc. directly:

No Starch Press, Inc.

245 8th Street, San Francisco, CA 94103

phone: 1-415-863-9900; info@nostarch.com

www.nostarch.com

Names: Li, Vickie, author.

Title: Bug bounty bootcamp : the guide to finding and reporting web vulnerabilities / Vickie Li.

Description: San Francisco : No Starch Press, [2021] | Includes index. |

Identifiers: LCCN 2021023153 (print) | LCCN 2021023154 (ebook) | ISBN 9781718501546 (print) | ISBN 9781718501553 (ebook)

Subjects: LCSH: Web sites--Security measures. | Penetration testing (Computer security) | Debugging in computer science.

Classification: LCC TK5105.8855 .L523 2021 (print) | LCC TK5105.8855 (ebook) | DDC 025.042--dc23

LC record available at <https://lcn.loc.gov/2021023153>

LC ebook record available at <https://lcn.loc.gov/2021023154>

No Starch Press and the No Starch Press logo are registered trademarks of No Starch Press, Inc. Other product and company names mentioned herein may be the trademarks of their respective owners. Rather than use a trademark symbol with every occurrence of a trademarked name, we are using the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The information in this book is distributed on an “As Is” basis, without warranty. While every precaution has been taken in the preparation of this work, neither the author nor No Starch Press, Inc. shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in it.

## About the Author

Vickie Li is a developer and security researcher experienced in finding and exploiting vulnerabilities in web applications. She has reported vulnerabilities to firms such as Facebook, Yelp, and Starbucks and contributes to a number of online training programs and technical blogs. She can be found at <https://vickieli.dev/>, where she blogs about security news, techniques, and her latest bug bounty findings.

## About the Tech Reviewer

Aaron Guzman is co-author of *IoT Penetration Testing Cookbook* and product security lead with Cisco Meraki. He spends his days building security into IoT products and crafting designs that keep users safe from compromise. A co-chair of Cloud Security Alliance's IoT Working Group and a technical reviewer for several published security books, he also spearheads many open-source initiatives, raising awareness about IoT hacking and proactive defensive strategies under OWASP's IoT and Embedded Application Security projects. He has extensive public speaking experience, delivering conference presentations, training, and workshops globally. Follow Aaron on Twitter [@scriptingxss](#).

# BRIEF CONTENTS

|                        |     |
|------------------------|-----|
| Foreword . . . . .     | xix |
| Introduction . . . . . | xxi |

## **PART I: THE INDUSTRY . . . . .1**

|  |    |
|--|----|
| Chapter 1: Picking a Bug Bounty Program. . . . . | 3  |
| Chapter 2: Sustaining Your Success . . . . .     | 15 |

## **PART II: GETTING STARTED . . . . .31**

|   |    |
|---|----|
| Chapter 3: How the Internet Works . . . . .                       | 33 |
| Chapter 4: Environmental Setup and Traffic Interception . . . . . | 45 |
| Chapter 5: Web Hacking Reconnaissance . . . . .                   | 61 |

## **PART III: WEB VULNERABILITIES . . . . .109**

|  |     |
|--|-----|
| Chapter 6: Cross-Site Scripting . . . . .                                | 111 |
| Chapter 7: Open Redirects . . . . .                                      | 131 |
| Chapter 8: Clickjacking . . . . .  | 143 |
| Chapter 9: Cross-Site Request Forgery . . . . .                          | 155 |
| Chapter 10: Insecure Direct Object References . . . . .                  | 175 |
| Chapter 11: SQL Injection. . . . .                                       | 187 |
| Chapter 12: Race Conditions . . . . .                                    | 205 |
| Chapter 13: Server-Side Request Forgery . . . . .                        | 213 |
| Chapter 14: Insecure Deserialization . . . . .                           | 231 |
| Chapter 15: XML External Entity. . . . .                                 | 247 |
| Chapter 16: Template Injection . . . . .                                 | 261 |
| Chapter 17: Application Logic Errors and Broken Access Control . . . . . | 275 |

|   |            |
|---|------------|
| Chapter 18: Remote Code Execution . . . . .                           | 283        |
| Chapter 19: Same-Origin Policy Vulnerabilities . . . . .              | 295        |
| Chapter 20: Single-Sign-On Security Issues . . . . .                  | 307        |
| Chapter 21: Information Disclosure . . . . .                          | 323        |
| <b>PART IV: EXPERT TECHNIQUES . . . . .</b>                           | <b>333</b> |
| Chapter 22: Conducting Code Reviews . . . . .                         | 335        |
| Chapter 23: Hacking Android Apps. . . . .                             | 347        |
| Chapter 24: API Hacking . . . . .                                     | 355        |
| Chapter 25: Automatic Vulnerability Discovery Using Fuzzers . . . . . | 369        |
| Index . . . . .   | 381        |

# CONTENTS IN DETAIL

|                 |            |
|-----------------|------------|
| <b>FOREWORD</b> | <b>xix</b> |
|-----------------|------------|

|                     |            |
|---------------------|------------|
| <b>INTRODUCTION</b> | <b>xxi</b> |
|---------------------|------------|

|                                |      |
|--------------------------------|------|
| Who This Book Is For . . . . . | xxii |
| What Is In This Book. . . . .  | xxii |
| Happy Hacking! . . . . .       | xxiv |

## **PART I: THE INDUSTRY** **1**

|                                     |          |
|-------------------------------------|----------|
| <b>1</b>                            |          |
| <b>PICKING A BUG BOUNTY PROGRAM</b> | <b>3</b> |

|   |    |
|---|----|
| The State of the Industry . . . . .                       | 4  |
| Asset Types . . . . .                                     | 4  |
| Social Sites and Applications . . . . .                   | 5  |
| General Web Applications. . . . .                         | 5  |
| Mobile Applications (Android, iOS, and Windows) . . . . . | 6  |
| APIs . . . . .  | 6  |
| Source Code and Executables . . . . .                     | 7  |
| Hardware and IoT. . . . .                                 | 7  |
| Bug Bounty Platforms . . . . .                            | 8  |
| The Pros. . . . .   | 8  |
| and the Cons . . . . .                                    | 9  |
| Scope, Payouts, and Response Times . . . . .              | 9  |
| Program Scope. . . . .                                    | 9  |
| Payout Amounts . . . . .                                  | 10 |
| Response Time . . . . .                                   | 11 |
| Private Programs . . . . .                                | 11 |
| Choosing the Right Program . . . . .                      | 12 |
| A Quick Comparison of Popular Programs . . . . .          | 13 |

|                                |           |
|--------------------------------|-----------|
| <b>2</b>                       |           |
| <b>SUSTAINING YOUR SUCCESS</b> | <b>15</b> |

|   |    |
|---|----|
| Writing a Good Report. . . . .                  | 16 |
| Step 1: Craft a Descriptive Title. . . . .      | 16 |
| Step 2: Provide a Clear Summary . . . . .       | 16 |
| Step 3: Include a Severity Assessment . . . . . | 16 |
| Step 4: Give Clear Steps to Reproduce . . . . . | 18 |
| Step 5: Provide a Proof of Concept . . . . .    | 18 |

|   |    |
|---|----|
| Step 6: Describe the Impact and Attack Scenarios . . . . .  | 19 |
| Step 7: Recommend Possible Mitigations . . . . .            | 19 |
| Step 8: Validate the Report . . . . .                       | 20 |
| Additional Tips for Writing Better Reports . . . . .        | 20 |
| Building a Relationship with the Development Team . . . . . | 21 |
| Understanding Report States . . . . .                       | 21 |
| Dealing with Conflict . . . . .                             | 23 |
| Building a Partnership . . . . .                            | 23 |
| Understanding Why You're Failing . . . . .                  | 24 |
| Why You're Not Finding Bugs . . . . .                       | 24 |
| Why Your Reports Get Dismissed . . . . .                    | 26 |
| What to Do When You're Stuck . . . . .                      | 27 |
| Step 1: Take a Break! . . . . .                             | 28 |
| Step 2: Build Your Skill Set . . . . .                      | 28 |
| Step 3: Gain a Fresh Perspective . . . . .                  | 28 |
| Lastly, a Few Words of Experience . . . . .                 | 29 |

## PART II: GETTING STARTED

## 31

### 3

#### HOW THE INTERNET WORKS

#### 33

|   |    |
|---|----|
| The Client-Server Model . . . . .             | 34 |
| The Domain Name System . . . . .              | 34 |
| Internet Ports . . . . .                      | 35 |
| HTTP Requests and Responses . . . . .         | 36 |
| Internet Security Controls . . . . .          | 38 |
| Content Encoding . . . . .                    | 38 |
| Session Management and HTTP Cookies . . . . . | 39 |
| Token-Based Authentication . . . . .          | 40 |
| JSON Web Tokens . . . . .                     | 41 |
| The Same-Origin Policy . . . . .              | 43 |
| Learn to Program . . . . .                    | 44 |

### 4

#### ENVIRONMENTAL SETUP AND TRAFFIC INTERCEPTION

#### 45

|  |    |
|--|----|
| Choosing an Operating System . . . . .                     | 46 |
| Setting Up the Essentials: A Browser and a Proxy . . . . . | 46 |
| Opening the Embedded Browser . . . . .                     | 47 |
| Setting Up Firefox . . . . .                               | 47 |
| Setting Up Burp . . . . .                                  | 49 |
| Using Burp . . . . .                                       | 51 |
| The Proxy . . . . .  | 52 |
| The Intruder . . . . .                                     | 54 |
| The Repeater . . . . .                                     | 56 |
| The Decoder . . . . .                                      | 57 |
| The Comparer . . . . .                                     | 58 |
| Saving Burp Requests . . . . .                             | 58 |
| A Final Note on Taking Notes . . . . .                     | 58 |

## 5

### WEB HACKING RECONNAISSANCE

61

|   |     |
|---|-----|
| Manually Walking Through the Target . . . . .       | 62  |
| Google Dorking . . . . .                            | 62  |
| Scope Discovery . . . . .                           | 65  |
| WHOIS and Reverse WHOIS . . . . .                   | 65  |
| IP Addresses . . . . .                              | 66  |
| Certificate Parsing . . . . .                       | 67  |
| Subdomain Enumeration . . . . .                     | 68  |
| Service Enumeration . . . . .                       | 69  |
| Directory Brute-Forcing . . . . .                   | 70  |
| Spidering the Site . . . . .                        | 71  |
| Third-Party Hosting . . . . .                       | 74  |
| GitHub Recon . . . . .                              | 75  |
| Other Sneaky OSINT Techniques . . . . .             | 77  |
| Tech Stack Fingerprinting . . . . .                 | 78  |
| Writing Your Own Recon Scripts . . . . .            | 80  |
| Understanding Bash Scripting Basics . . . . .       | 80  |
| Saving Tool Output to a File . . . . .              | 83  |
| Adding the Date of the Scan to the Output . . . . . | 84  |
| Adding Options to Choose the Tools to Run . . . . . | 84  |
| Running Additional Tools . . . . .                  | 85  |
| Parsing the Results . . . . .                       | 88  |
| Building a Master Report . . . . .                  | 90  |
| Scanning Multiple Domains . . . . .                 | 92  |
| Writing a Function Library . . . . .                | 96  |
| Building Interactive Programs . . . . .             | 97  |
| Using Special Variables and Characters . . . . .    | 100 |
| Scheduling Automatic Scans . . . . .                | 102 |
| A Note on Recon APIs . . . . .                      | 104 |
| Start Hacking! . . . . .                            | 104 |
| Tools Mentioned in This Chapter . . . . .           | 105 |
| Scope Discovery . . . . .                           | 105 |
| OSINT . . . . .                                     | 106 |
| Tech Stack Fingerprinting . . . . .                 | 106 |
| Automation . . . . .                                | 107 |

## PART III: WEB VULNERABILITIES

109

## 6

### CROSS-SITE SCRIPTING

111

|                         |     |
|-------------------------|-----|
| Mechanisms . . . . .    | 112 |
| Types of XSS . . . . .  | 115 |
| Stored XSS . . . . .    | 115 |
| Blind XSS . . . . .     | 116 |
| Reflected XSS . . . . . | 117 |
| DOM-Based XSS . . . . . | 117 |
| Self-XSS . . . . .      | 119 |
| Prevention . . . . .    | 119 |



|  |     |
|--|-----|
| Hunting for XSS . . . . .                      | 120 |
| Step 1: Look for Input Opportunities . . . . . | 120 |
| Step 2: Insert Payloads . . . . .              | 122 |
| Step 3: Confirm the Impact. . . . .            | 125 |
| Bypassing XSS Protection . . . . .             | 126 |
| Alternative JavaScript Syntax . . . . .        | 126 |
| Capitalization and Encoding . . . . .          | 126 |
| Filter Logic Errors . . . . .                  | 127 |
| Escalating the Attack . . . . .                | 128 |
| Automating XSS Hunting . . . . .               | 129 |
| Finding Your First XSS! . . . . .              | 129 |

## **7**

### **OPEN REDIRECTS** **131**

|  |     |
|--|-----|
| Mechanisms . . . . .   | 131 |
| Prevention . . . . .   | 133 |
| Hunting for Open Redirects . . . . .                                     | 133 |
| Step 1: Look for Redirect Parameters . . . . .                           | 133 |
| Step 2: Use Google Dorks to Find Additional Redirect Parameters. . . . . | 134 |
| Step 3: Test for Parameter-Based Open Redirects. . . . .                 | 135 |
| Step 4: Test for Referer-Based Open Redirects . . . . .                  | 135 |
| Bypassing Open-Redirect Protection . . . . .                             | 136 |
| Using Browser Autocorrect . . . . .                                      | 136 |
| Exploiting Flawed Validator Logic . . . . .                              | 137 |
| Using Data URLs . . . . .  | 138 |
| Exploiting URL Decoding . . . . .  | 138 |
| Combining Exploit Techniques . . . . .                                   | 140 |
| Escalating the Attack . . . . .  | 140 |
| Finding Your First Open Redirect! . . . . .                              | 141 |

## **8**

### **CLICKJACKING** **143**

|  |     |
|--|-----|
| Mechanisms . . . . .                                     | 144 |
| Prevention . . . . .                                     | 149 |
| Hunting for Clickjacking . . . . .                       | 150 |
| Step 1: Look for State-Changing Actions . . . . .        | 150 |
| Step 2: Check the Response Headers . . . . .             | 151 |
| Step 3: Confirm the Vulnerability. . . . .               | 151 |
| Bypassing Protections . . . . .                          | 151 |
| Escalating the Attack . . . . .                          | 153 |
| A Note on Delivering the Clickjacking Payload . . . . .  | 154 |
| Finding Your First Clickjacking Vulnerability! . . . . . | 154 |

## **9**

### **CROSS-SITE REQUEST FORGERY** **155**

|   |     |
|---|-----|
| Mechanisms . . . . .                                  | 156 |
| Prevention . . . . .                                  | 159 |
| Hunting for CSRFs . . . . .                           | 161 |
| Step 1: Spot State-Changing Actions . . . . .         | 161 |
| Step 2: Look for a Lack of CSRF Protections . . . . . | 161 |
| Step 3: Confirm the Vulnerability. . . . .            | 162 |

|   |     |
|---|-----|
| Bypassing CSRF Protection . . . . .               | 163 |
| Exploit Clickjacking . . . . .                    | 163 |
| Change the Request Method . . . . .               | 164 |
| Bypass CSRF Tokens Stored on the Server . . . . . | 165 |
| Bypass Double-Submit CSRF Tokens . . . . .        | 167 |
| Bypass CSRF Referer Header Check . . . . .        | 168 |
| Bypass CSRF Protection by Using XSS . . . . .     | 170 |
| Escalating the Attack . . . . .                   | 170 |
| Leak User Information by Using CSRF . . . . .     | 170 |
| Create Stored Self-XSS by Using CSRF . . . . .    | 171 |
| Take Over User Accounts by Using CSRF . . . . .   | 172 |
| Delivering the CSRF Payload . . . . .             | 173 |
| Finding Your First CSRF! . . . . .                | 174 |

## 10

### INSECURE DIRECT OBJECT REFERENCES 175

|   |     |
|---|-----|
| Mechanisms . . . . .  | 175 |
| Prevention . . . . .  | 177 |
| Hunting for IDORs . . . . .   | 178 |
| Step 1: Create Two Accounts . . . . .                                 | 178 |
| Step 2: Discover Features . . . . .                                   | 178 |
| Step 3: Capture Requests . . . . .                                    | 179 |
| Step 4: Change the IDs . . . . .                                      | 180 |
| Bypassing IDOR Protection . . . . .                                   | 181 |
| Encoded IDs and Hashed IDs . . . . .                                  | 181 |
| Leaked IDs . . . . .  | 182 |
| Offer the Application an ID, Even If It Doesn't Ask for One . . . . . | 182 |
| Keep an Eye Out for Blind IDORs . . . . .                             | 183 |
| Change the Request Method . . . . .                                   | 183 |
| Change the Requested File Type . . . . .                              | 184 |
| Escalating the Attack . . . . .                                       | 184 |
| Automating the Attack . . . . .                                       | 185 |
| Finding Your First IDOR! . . . . .                                    | 185 |

## 11

### SQL INJECTION 187

|  |     |
|--|-----|
| Mechanisms . . . . .   | 188 |
| Injecting Code into SQL Queries . . . . .                        | 189 |
| Using Second-Order SQL Injections . . . . .                      | 191 |
| Prevention . . . . .   | 192 |
| Hunting for SQL Injections . . . . .                             | 195 |
| Step 1: Look for Classic SQL Injections . . . . .                | 195 |
| Step 2: Look for Blind SQL Injections . . . . .                  | 196 |
| Step 3: Exfiltrate Information by Using SQL Injections . . . . . | 198 |
| Step 4: Look for NoSQL Injections . . . . .                      | 199 |
| Escalating the Attack . . . . .                                  | 201 |
| Learn About the Database . . . . .                               | 201 |
| Gain a Web Shell . . . . .                                       | 202 |
| Automating SQL Injections . . . . .                              | 202 |
| Finding Your First SQL Injection! . . . . .                      | 203 |

|   |                                    |            |
|---|------------------------------------|------------|
| <b>12</b>   | <b>RACE CONDITIONS</b>             | <b>205</b> |
| Mechanisms . . . . .  |                                    | 206        |
| When a Race Condition Becomes a Vulnerability. . . . .                        |                                    | 207        |
| Prevention . . . . .  |                                    | 210        |
| Hunting for Race Conditions . . . . .   |                                    | 210        |
| Step 1: Find Features Prone to Race Conditions . . . . .                      |                                    | 210        |
| Step 2: Send Simultaneous Requests . . . . .                                  |                                    | 210        |
| Step 3: Check the Results . . . . .   |                                    | 211        |
| Step 4: Create a Proof of Concept . . . . .                                   |                                    | 211        |
| Escalating Race Conditions . . . . .  |                                    | 212        |
| Finding Your First Race Condition! . . . . .                                  |                                    | 212        |
| <b>13</b>   | <b>SERVER-SIDE REQUEST FORGERY</b> | <b>213</b> |
| Mechanisms . . . . .  |                                    | 213        |
| Prevention . . . . .  |                                    | 215        |
| Hunting for SSRFs. . . . .  |                                    | 216        |
| Step 1: Spot Features Prone to SSRFs. . . . .                                 |                                    | 216        |
| Step 2: Provide Potentially Vulnerable Endpoints with Internal URLs . . . . . |                                    | 218        |
| Step 3: Check the Results . . . . .   |                                    | 218        |
| Bypassing SSRF Protection. . . . .  |                                    | 220        |
| Bypass Allowlists. . . . .  |                                    | 220        |
| Bypass Blocklists . . . . .   |                                    | 221        |
| Escalating the Attack . . . . .   |                                    | 224        |
| Perform Network Scanning. . . . .   |                                    | 224        |
| Pull Instance Metadata. . . . .   |                                    | 226        |
| Exploit Blind SSRFs . . . . .   |                                    | 227        |
| Attack the Network . . . . .  |                                    | 228        |
| Finding Your First SSRF! . . . . .  |                                    | 229        |
| <b>14</b>   | <b>INSECURE DESERIALIZATION</b>    | <b>231</b> |
| Mechanisms . . . . .  |                                    | 232        |
| PHP . . . . .   |                                    | 232        |
| Java . . . . .  |                                    | 241        |
| Prevention . . . . .  |                                    | 244        |
| Hunting for Insecure Deserialization . . . . .                                |                                    | 244        |
| Escalating the Attack . . . . .   |                                    | 245        |
| Finding Your First Insecure Deserialization! . . . . .                        |                                    | 246        |
| <b>15</b>   | <b>XML EXTERNAL ENTITY</b>         | <b>247</b> |
| Mechanisms . . . . .  |                                    | 247        |
| Prevention . . . . .  |                                    | 249        |
| Hunting for XXEs . . . . .  |                                    | 250        |
| Step 1: Find XML Data Entry Points . . . . .                                  |                                    | 250        |
| Step 2: Test for Classic XXE . . . . .  |                                    | 251        |
| Step 3: Test for Blind XXE . . . . .  |                                    | 252        |

|  |     |
|--|-----|
| Step 4: Embed XXE Payloads in Different File Types . . . . . | 253 |
| Step 5: Test for XInclude Attacks . . . . .                  | 254 |
| Escalating the Attack . . . . .                              | 254 |
| Reading Files . . . . .                                      | 255 |
| Launching an SSRF . . . . .                                  | 255 |
| Using Blind XXEs . . . . .                                   | 256 |
| Performing Denial-of-Service Attacks . . . . .               | 258 |
| More About Data Exfiltration Using XXEs . . . . .            | 259 |
| Finding Your First XXE! . . . . .                            | 260 |

## 16

|   |            |
|---|------------|
| <b>TEMPLATE INJECTION</b>   | <b>261</b> |
| Mechanisms . . . . .  | 262        |
| Template Engines . . . . .  | 262        |
| Injecting Template Code . . . . .                                       | 263        |
| Prevention . . . . .  | 265        |
| Hunting for Template Injection . . . . .                                | 266        |
| Step 1: Look for User-Input Locations . . . . .                         | 266        |
| Step 2: Detect Template Injection by Submitting Test Payloads . . . . . | 266        |
| Step 3: Determine the Template Engine in Use . . . . .                  | 268        |
| Escalating the Attack . . . . .   | 268        |
| Searching for System Access via Python Code . . . . .                   | 269        |
| Escaping the Sandbox by Using Python Built-in Functions . . . . .       | 270        |
| Submitting Payloads for Testing . . . . .                               | 273        |
| Automating Template Injection . . . . .                                 | 273        |
| Finding Your First Template Injection! . . . . .                        | 274        |

## 17

|  |            |
|--|------------|
| <b>APPLICATION LOGIC ERRORS AND BROKEN ACCESS CONTROL</b>                      | <b>275</b> |
| Application Logic Errors . . . . .   | 276        |
| Broken Access Control . . . . .  | 278        |
| Exposed Admin Panels . . . . .   | 278        |
| Directory Traversal Vulnerabilities . . . . .                                  | 279        |
| Prevention . . . . .   | 279        |
| Hunting for Application Logic Errors and Broken Access Control . . . . .       | 280        |
| Step 1: Learn About Your Target . . . . .                                      | 280        |
| Step 2: Intercept Requests While Browsing . . . . .                            | 280        |
| Step 3: Think Outside the Box . . . . .  | 280        |
| Escalating the Attack . . . . .  | 281        |
| Finding Your First Application Logic Error or Broken Access Control! . . . . . | 281        |

## 18

|  |            |
|--|------------|
| <b>REMOTE CODE EXECUTION</b>                               | <b>283</b> |
| Mechanisms . . . . .                                       | 284        |
| Code Injection . . . . .                                   | 284        |
| File Inclusion . . . . .                                   | 286        |
| Prevention . . . . .                                       | 287        |
| Hunting for RCEs . . . . .                                 | 288        |
| Step 1: Gather Information About the Target . . . . .      | 289        |
| Step 2: Identify Suspicious User Input Locations . . . . . | 289        |

|  |     |
|--|-----|
| Step 3: Submit Test Payloads . . . . .     | 289 |
| Step 4: Confirm the Vulnerability. . . . . | 290 |
| Escalating the Attack . . . . .            | 291 |
| Bypassing RCE Protection . . . . .         | 291 |
| Finding Your First RCE! . . . . .          | 293 |

## 19

### **SAME-ORIGIN POLICY VULNERABILITIES 295**

|   |     |
|---|-----|
| Mechanisms . . . . .  | 296 |
| Exploiting Cross-Origin Resource Sharing. . . . .                 | 297 |
| Exploiting postMessage() . . . . .                                | 298 |
| Exploiting JSON with Padding . . . . .                            | 300 |
| Bypassing SOP by Using XSS. . . . .                               | 302 |
| Hunting for SOP Bypasses. . . . .                                 | 302 |
| Step 1: Determine If SOP Relaxation Techniques Are Used . . . . . | 302 |
| Step 2: Find CORS Misconfiguration . . . . .                      | 303 |
| Step 3: Find postMessage Bugs . . . . .                           | 304 |
| Step 4: Find JSONP Issues . . . . .                               | 305 |
| Step 5: Consider Mitigating Factors . . . . .                     | 305 |
| Escalating the Attack . . . . .                                   | 305 |
| Finding Your First SOP Bypass Vulnerability! . . . . .            | 306 |

## 20

### **SINGLE-SIGN-ON SECURITY ISSUES 307**

|  |     |
|--|-----|
| Mechanisms . . . . .                           | 308 |
| Cooking Sharing. . . . .                       | 308 |
| Security Assertion Markup Language. . . . .    | 309 |
| OAuth . . . . .                                | 312 |
| Hunting for Subdomain Takeovers . . . . .      | 316 |
| Step 1: List the Target's Subdomains . . . . . | 316 |
| Step 2: Find Unregistered Pages . . . . .      | 316 |
| Step 3: Register the Page . . . . .            | 317 |
| Monitoring for Subdomain Takeovers . . . . .   | 318 |
| Hunting for SAML Vulnerabilities . . . . .     | 319 |
| Step 1: Locate the SAML Response . . . . .     | 319 |
| Step 2: Analyze the Response Fields . . . . .  | 319 |
| Step 3: Bypass the Signature . . . . .         | 319 |
| Step 4: Re-encode the Message . . . . .        | 320 |
| Hunting for OAuth Token Theft. . . . .         | 320 |
| Escalating the Attack . . . . .                | 321 |
| Finding Your First SSO Bypass! . . . . .       | 321 |

## 21

### **INFORMATION DISCLOSURE 323**

|   |     |
|---|-----|
| Mechanisms . . . . .                              | 324 |
| Prevention . . . . .                              | 324 |
| Hunting for Information Disclosure . . . . .      | 325 |
| Step 1: Attempt a Path Traversal Attack . . . . . | 325 |
| Step 2: Search the Wayback Machine. . . . .       | 326 |

|  |     |
|--|-----|
| Step 3: Search Paste Dump Sites . . . . .                                | 327 |
| Step 4: Reconstruct Source Code from an Exposed .git Directory . . . . . | 328 |
| Step 5: Find Information in Public Files. . . . .                        | 331 |
| Escalating the Attack . . . . .  | 332 |
| Finding Your First Information Disclosure! . . . . .                     | 332 |

## **PART IV: EXPERT TECHNIQUES 333**

### **22 CONDUCTING CODE REVIEWS 335**

|   |     |
|---|-----|
| White-Box vs. Black-Box Testing . . . . .                           | 336 |
| The Fast Approach: grep Is Your Best Friend . . . . .               | 336 |
| Dangerous Patterns . . . . .  | 336 |
| Leaked Secrets and Weak Encryption . . . . .                        | 338 |
| New Patches and Outdated Dependencies. . . . .                      | 340 |
| Developer Comments. . . . .   | 340 |
| Debug Functionalities, Configuration Files, and Endpoints . . . . . | 340 |
| The Detailed Approach. . . . .                                      | 341 |
| Important Functions . . . . .                                       | 341 |
| User Input . . . . .  | 342 |
| Exercise: Spot the Vulnerabilities . . . . .                        | 344 |

### **23 HACKING ANDROID APPS 347**

|  |     |
|--|-----|
| Setting Up Your Mobile Proxy . . . . . | 348 |
| Bypassing Certificate Pinning. . . . . | 349 |
| Anatomy of an APK . . . . .            | 350 |
| Tools to Use. . . . .                  | 351 |
| Android Debug Bridge. . . . .          | 351 |
| Android Studio . . . . .               | 352 |
| Apktool . . . . .                      | 352 |
| Frida . . . . .                        | 353 |
| Mobile Security Framework . . . . .    | 353 |
| Hunting for Vulnerabilities . . . . .  | 353 |

### **24 API HACKING 355**

|  |     |
|--|-----|
| What Are APIs?. . . . .                                    | 355 |
| REST APIs. . . . .   | 357 |
| SOAP APIs . . . . .  | 358 |
| GraphQL APIs . . . . .                                     | 358 |
| API-Centric Applications. . . . .                          | 361 |
| Hunting for API Vulnerabilities . . . . .                  | 362 |
| Performing Recon . . . . .                                 | 362 |
| Testing for Broken Access Control and Info Leaks . . . . . | 364 |
| Testing for Rate-Limiting Issues. . . . .                  | 365 |
| Testing for Technical Bugs. . . . .                        | 366 |

## **AUTOMATIC VULNERABILITY DISCOVERY USING FUZZERS 369**

|   |     |
|---|-----|
| What Is Fuzzing? . . . . .                            | 370 |
| How a Web Fuzzer Works . . . . .                      | 370 |
| The Fuzzing Process . . . . .                         | 371 |
| Step 1: Determine the Data Injection Points . . . . . | 371 |
| Step 2: Decide on the Payload List . . . . .          | 372 |
| Step 3: Fuzz. . . . .                                 | 372 |
| Step 4: Monitor the Results . . . . .                 | 374 |
| Fuzzing with Wfuzz . . . . .                          | 374 |
| Path Enumeration . . . . .                            | 374 |
| Brute-Forcing Authentication . . . . .                | 376 |
| Testing for Common Web Vulnerabilities . . . . .      | 377 |
| More About Wfuzz . . . . .                            | 378 |
| Fuzzing vs. Static Analysis . . . . .                 | 378 |
| Pitfalls of Fuzzing . . . . .                         | 378 |
| Adding to Your Automated Testing Toolkit . . . . .    | 379 |

## **INDEX 381**