The Web Application Defender’s Cookbook

Battling Hackers and Protecting Users

Ryan Barnett
This book is dedicated to my incredible daughter, Isabella. You are so full of imagination, kindness, and humor that I have a constant smile on my face. You are my Supergirl-flying, tae-kwon-do-kicking, fairy princess! I thank God every day for bringing you into my life and for allowing me the joy and privilege of being your father.

I love you Izzy, and I am so proud of you.
About the Author

**Ryan Barnett** is renowned in the web application security industry for his unique expertise. After a decade of experience defending government and commercial web sites, he joined the Trustwave SpiderLabs Research Team. He specializes in application defense research and leads the open source ModSecurity web application firewall project.

In addition to his commercial work at Trustwave, Ryan is also an active contributor to many community-based security projects. He serves as the Open Web Application Security Project (OWASP) ModSecurity Core Rule Set project leader and is a contributor on the OWASP Top Ten and AppSensor projects. He is a Web Application Security Consortium Board Member and leads the Web Hacking Incident Database and the Distributed Web Honeypot projects. At the SANS Institute, he is a certified instructor and contributor on the Top 20 Vulnerabilities and CWE/SANS Top 25 Most Dangerous Programming Errors projects.

Ryan is regularly consulted by news outlets that seek his insights into and analysis of emerging web application attacks, trends, and defensive techniques. He is a frequent speaker and trainer at key industry events including Black Hat, SANS AppSec Summit, and OWASP AppSecUSA.
Michael Gregg is the CEO of Superior Solutions, Inc. (www.thesolutionfirm.com), a Houston-based IT security-consulting firm. His organization performs security assessments and penetration testing for Fortune 1000 firms. He is frequently cited by major and trade print publications as a cyber security expert. He has appeared as an expert commentator for network broadcast outlets and print publications and has spoken at major security conferences such as HackerHalted, Fusion, and CA World.
must begin by thanking my wonderful wife, Linda. When I came to her with the idea of writing another book, she was fully supportive even though she understood the sacrifice it would require. I thank her for her continued patience and for enduring many late nights of “angry typing.” She has always encouraged and supported me both professionally and personally. The completion of this book is not my accomplishment alone but our whole family’s because it was truly a team effort. I love you Linda and am honored that you are my partner for life.

I would like to thank Nick Percoco, Senior VP of Trustwave SpiderLabs, for his unwavering support of ModSecurity and for appointing me as its manager. I am fortunate to work with intelligent, clever and funny people. Unfortunately I cannot list them all here, however, I must single out Breno Silva Pinto. Breno is the lead developer of ModSecurity and we have worked closely on the project for 2 years. I am constantly impressed with his insights, ingenuity and technical skill for web application security. This book would not have been possible without Breno’s contributions to ModSecurity features and capabilities.

I would also like to thank two specific ModSecurity community members who epitomize the “giving back” philosophy of the open source community. Thanks to Christian Bockermann for developing many ModSecurity support tools such as the AuditConsole and to Josh Zlatin for always helping users on the mail-list and for contributions to the OWASP ModSecurity CRS.

Last but not least, I want to specifically thank OWASP members: Tom Brennan, Jim Manico, and Sarah Baso. Your tireless work ethic and commitment to the OWASP mission is undeniable. I would also like to thank Michael Coates for starting the AppSensor project and both Colin Watson and John Melton for expanding its capabilities.
## Contents

**Foreword** .................................................. xix

**Introduction** ............................................... xxiii

### I  Preparing the Battle Space ......................... 1

1  Application Fortification ................................. 7

   *Recipe 1-1: Real-time Application Profiling* ............... 7

   *Recipe 1-2: Preventing Data Manipulation with Cryptographic Hash Tokens* ..................... 15

   *Recipe 1-3: Installing the OWASP ModSecurity Core Rule Set (CRS)* ......................... 19

   *Recipe 1-4: Integrating Intrusion Detection System Signatures* .......................... 33

   *Recipe 1-5: Using Bayesian Attack Payload Detection* ........................................ 38

   *Recipe 1-6: Enable Full HTTP Audit Logging* .................................................... 48

   *Recipe 1-7: Logging Only Relevant Transactions* .................................................. 52

   *Recipe 1-8: Ignoring Requests for Static Content* .............................................. 53

   *Recipe 1-9: Obscuring Sensitive Data in Logs* .................................................... 54

   *Recipe 1-10: Sending Alerts to a Central Log Host Using Syslog* ............................ 58

   *Recipe 1-11: Using the ModSecurity AuditConsole* ................................................ 60

2  Vulnerability Identification and Remediation ............... 67

   *Recipe 2-1: Passive Vulnerability Identification* .................................................. 70

   *Recipe 2-2: Active Vulnerability Identification* .................................................. 79

   *Recipe 2-3: Manual Scan Result Conversion* ......................................................... 88

   *Recipe 2-4: Automated Scan Result Conversion* .................................................... 92

   *Recipe 2-5: Real-time Resource Assessments and Virtual Patching* ......................... 99

3  Poisoned Pawns (Hacker Traps) ............................... 115

   *Recipe 3-1: Adding Honeypot Ports* ................................................................. 116

   *Recipe 3-2: Adding Fake robots.txt Disallow Entries* ........................................ 118

   *Recipe 3-3: Adding Fake HTML Comments* ......................................................... 123

   *Recipe 3-4: Adding Fake Hidden Form Fields* ..................................................... 128

   *Recipe 3-5: Adding Fake Cookies* ................................................................. 131
II

Asymmetric Warfare ......................... 137

4 Reputation and Third-Party Correlation ............... 139
Recipe 4-1: Analyzing the Client’s Geographic Location Data .............. 141
Recipe 4-2: Identifying Suspicious Open Proxy Usage .................. 147
Recipe 4-3: Utilizing Real-time Blacklist Lookups (RBL) .............. 150
Recipe 4-4: Running Your Own RBL ................................ 157
Recipe 4-5: Detecting Malicious Links ........................... 160

5 Request Data Analysis ........................... 171
Recipe 5-1: Request Body Access ............................. 172
Recipe 5-2: Identifying Malformed Request Bodies ..................... 178
Recipe 5-3: Normalizing Unicode ............................... 182
Recipe 5-4: Identifying Use of Multiple Encodings .................... 186
Recipe 5-5: Identifying Encoding Anomalies ........................ 189
Recipe 5-6: Detecting Request Method Anomalies ..................... 193
Recipe 5-7: Detecting Invalid URI Data .......................... 197
Recipe 5-8: Detecting Request Header Anomalies .................... 200
Recipe 5-9: Detecting Additional Parameters ........................ 209
Recipe 5-10: Detecting Missing Parameters ........................ 212
Recipe 5-11: Detecting Duplicate Parameter Names .................... 214
Recipe 5-12: Detecting Parameter Payload Size Anomalies .......... 216
Recipe 5-13: Detecting Parameter Character Class Anomalies .......... 219

6 Response Data Analysis ........................... 223
Recipe 6-1: Detecting Response Header Anomalies .................... 224
Recipe 6-2: Detecting Response Header Information Leakages .......... 234
Recipe 6-3: Response Body Access .............................. 238
Recipe 6-4: Detecting Page Title Changes .......................... 240
Recipe 6-5: Detecting Page Size Deviations ......................... 243
Recipe 6-6: Detecting Dynamic Content Changes ..................... 246
Recipe 6-7: Detecting Source Code Leakages ........................ 249
Recipe 6-8: Detecting Technical Data Leakages ...................... 253
Recipe 6-9: Detecting Abnormal Response Time Intervals .......... 256
Recipe 6-10: Detecting Sensitive User Data Leakages ................ 259
Recipe 6-11: Detecting Trojan, Backdoor, and Webshell Access Attempts .. 262
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Defending Authentication</td>
<td>265-280</td>
</tr>
<tr>
<td>7-1</td>
<td>Recipe 7-1: Detecting the Submission of Common/Default Usernames</td>
<td>266</td>
</tr>
<tr>
<td>7-2</td>
<td>Recipe 7-2: Detecting the Submission of Multiple Usernames</td>
<td>269</td>
</tr>
<tr>
<td>7-3</td>
<td>Recipe 7-3: Detecting Failed Authentication Attempts</td>
<td>272</td>
</tr>
<tr>
<td>7-4</td>
<td>Recipe 7-4: Detecting a High Rate of Authentication Attempts</td>
<td>274</td>
</tr>
<tr>
<td>7-5</td>
<td>Recipe 7-5: Normalizing Authentication Failure Details</td>
<td>280</td>
</tr>
<tr>
<td>7-6</td>
<td>Recipe 7-6: Enforcing Password Complexity</td>
<td>283</td>
</tr>
<tr>
<td>7-7</td>
<td>Recipe 7-7: Correlating Usernames with SessionIDs</td>
<td>286</td>
</tr>
<tr>
<td>8</td>
<td>Defending Session State</td>
<td>291-314</td>
</tr>
<tr>
<td>8-1</td>
<td>Recipe 8-1: Detecting Invalid Cookies</td>
<td>291</td>
</tr>
<tr>
<td>8-2</td>
<td>Recipe 8-2: Detecting Cookie Tampering</td>
<td>297</td>
</tr>
<tr>
<td>8-3</td>
<td>Recipe 8-3: Enforcing Session Timeouts</td>
<td>302</td>
</tr>
<tr>
<td>8-4</td>
<td>Recipe 8-4: Detecting Client Source Location Changes During Session Lifetime</td>
<td>307</td>
</tr>
<tr>
<td>8-5</td>
<td>Recipe 8-5: Detecting Browser Fingerprint Changes During Sessions</td>
<td>314</td>
</tr>
<tr>
<td>9</td>
<td>Preventing Application Attacks</td>
<td>323-352</td>
</tr>
<tr>
<td>9-1</td>
<td>Recipe 9-1: Blocking Non-ASCII Characters</td>
<td>323</td>
</tr>
<tr>
<td>9-2</td>
<td>Recipe 9-2: Preventing Path-Traversal Attacks</td>
<td>327</td>
</tr>
<tr>
<td>9-3</td>
<td>Recipe 9-3: Preventing Forceful Browsing Attacks</td>
<td>330</td>
</tr>
<tr>
<td>9-4</td>
<td>Recipe 9-4: Preventing SQL Injection Attacks</td>
<td>332</td>
</tr>
<tr>
<td>9-5</td>
<td>Recipe 9-5: Preventing Remote File Inclusion (RFI) Attacks</td>
<td>336</td>
</tr>
<tr>
<td>9-6</td>
<td>Recipe 9-6: Preventing OS Commanding Attacks</td>
<td>340</td>
</tr>
<tr>
<td>9-7</td>
<td>Recipe 9-7: Preventing HTTP Request Smuggling Attacks</td>
<td>342</td>
</tr>
<tr>
<td>9-8</td>
<td>Recipe 9-8: Preventing HTTP Response Splitting Attacks</td>
<td>345</td>
</tr>
<tr>
<td>9-9</td>
<td>Recipe 9-9: Preventing XML Attacks</td>
<td>347</td>
</tr>
<tr>
<td>10</td>
<td>Preventing Client Attacks</td>
<td>353-381</td>
</tr>
<tr>
<td>10-1</td>
<td>Recipe 10-1: Implementing Content Security Policy (CSP)</td>
<td>353</td>
</tr>
<tr>
<td>10-2</td>
<td>Recipe 10-2: Preventing Cross-Site Scripting (XSS) Attacks</td>
<td>362</td>
</tr>
<tr>
<td>10-3</td>
<td>Recipe 10-3: Preventing Cross-Site Request Forgery (CSRF) Attacks</td>
<td>371</td>
</tr>
<tr>
<td>10-4</td>
<td>Recipe 10-4: Preventing UI Redressing (Clickjacking) Attacks</td>
<td>377</td>
</tr>
<tr>
<td>10-5</td>
<td>Recipe 10-5: Detecting Banking Trojan (Man-in-the-Browser) Attacks</td>
<td>381</td>
</tr>
<tr>
<td>11</td>
<td>Defending File Uploads</td>
<td>387-390</td>
</tr>
<tr>
<td>11-1</td>
<td>Recipe 11-1: Detecting Large File Sizes</td>
<td>387</td>
</tr>
<tr>
<td>11-2</td>
<td>Recipe 11-2: Detecting a Large Number of Files</td>
<td>389</td>
</tr>
<tr>
<td>11-3</td>
<td>Recipe 11-3: Inspecting File Attachments for Malware</td>
<td>390</td>
</tr>
</tbody>
</table>
12 Enforcing Access Rate and Application Flows ................. 395
  Recipe 12-1: Detecting High Application Access Rates ........ 395
  Recipe 12-2: Detecting Request/Response Delay Attacks .... 405
  Recipe 12-3: Identifying Inter-Request Time Delay Anomalies .. 411
  Recipe 12-4: Identifying Request Flow Anomalies .......... 413
  Recipe 12-5: Identifying a Significant Increase in Resource Usage .... 414

III Tactical Response ............................................. 419

13 Passive Response Actions ........................................ 421
  Recipe 13-1: Tracking Anomaly Scores ......................... 421
  Recipe 13-2: Trap and Trace Audit Logging .................... 427
  Recipe 13-3: Issuing E-mail Alerts ............................. 428
  Recipe 13-4: Data Sharing with Request Header Tagging ....... 436

14 Active Response Actions ......................................... 441
  Recipe 14-1: Using Redirection to Error Pages ................. 442
  Recipe 14-2: Dropping Connections ............................ 445
  Recipe 14-3: Blocking the Client Source Address .............. 447
  Recipe 14-4: Restricting Geolocation Access Through Defense Condition (DefCon) Level Changes ................. 452
  Recipe 14-5: Forcing Transaction Delays ...................... 455
  Recipe 14-6: Spoofing Successful Attacks ...................... 462
  Recipe 14-7: Proxying Traffic to Honeypots ................. 468
  Recipe 14-8: Forcing an Application Logout ................... 471
  Recipe 14-9: Temporarily Locking Account Access ............ 476

15 Intrusive Response Actions ..................................... 479
  Recipe 15-1: JavaScript Cookie Testing ......................... 479
  Recipe 15-2: Validating Users with CAPTCHA Testing ........ 481
  Recipe 15-3: Hooking Malicious Clients with BeEF ............ 485

Index ................................................................. 495
A defender, the person responsible for protecting IT systems from being compromised, could just as easily be the first line of defense as the last line. In fact, a defender working for an average organization might be the only line of defense—the only thing standing between the bad guy and a headline-making data breach. Worse yet, perhaps the incident doesn’t make headlines, and no one, including the defender, is the wiser.

Either way, when whatever crazy new Web 2.0 Ajax-laced HTML5-laden application has traversed the software development life cycle and successfully made it past the QA gate, when the third-party penetration testers are long gone, after management has signed off on all the security exceptions, and the application has been released to production, with or without the defender’s knowledge or consent, “security” then becomes entirely the defender’s responsibility. Rest assured that vulnerabilities will remain or will be introduced eventually. So, when all is said and done, a defender’s mission is to secure the insecure, to identify incoming attacks and thwart them, and to detect and contain breaches.

That’s why there should be no doubt about the importance of the role of a defender. Defenders often safeguard the personal data of millions of people. They may protect millions, perhaps billions, of dollars in online transactions and the core intellectual property of the entire business. You can bet that with so much on the line, with so much valuable information being stored, someone will want to steal it. And the bigger and more high profile the system, the more sustained and targeted the incoming attacks will be.

Making matters even more challenging, the bad guys have the luxury of picking their shots. They may attack a system whenever they want to, or not. A defender’s job is 24/7/365, holidays, weekends, vacation days. The system must be ready, and the defender must be ready, at all times.

A defender’s job description could read much like Ernest Shackleton’s famous advertisement when he was looking for men to accompany him on his next Antarctic expedition:

*Men wanted for hazardous journey. Low wages, bitter cold, long hours of complete darkness.*

*Safe return doubtful. Honour and recognition in event of success.*
A defender’s success really comes down to understanding a few key points about the operational environment in which he or she works:

- Web sites are often deployed in such a way that they cannot be adequately mirrored in development, QA, or even staging. This means that the real and true security posture, the real and true risk to the business, can be fully grasped only when it hits production and becomes an actual risk. As such, defenders must be able to think on their feet, be nimble, and react quickly.

- Defenders will find themselves responsible for protecting web sites they did not create and have little or no insight into or control over. Management may not respect security and may be unwilling to fix identified vulnerabilities in a timely fashion, and that could be the long-term standard operating procedure. And maybe this is the right call, depending on business risk and the estimated cost of software security. Whatever the case may be, defenders must be able to identify incoming attacks, block as many exploits as they can, and contain breaches.

- Fighting fires and responding to daily threats must be an expected part of the role. Whether the business is fully committed to software security is immaterial, because software will always have vulnerabilities. Furthermore, everyone gets attacked eventually. A defender never wants to be late in seeing an attack and the last one to know about a breach. For a defender, attack identification and response time are crucial.

- Defenders, because they are on the front lines, learn a tremendous amount about the application’s risk profile and the necessary security readiness required to thwart attackers. This intelligence is like gold when communicated to developers who are interested in creating ever more resilient systems. This intelligence is also like gold when informing the security assessment teams about what types of vulnerabilities they should focus on first when testing systems in either QA or production. Everyone needs actionable data. The best defenders have it.

Putting these practices to use requires specialized skills and experience. Normally, aspiring defenders don’t get this type of how-to instruction from product README files or FAQs. Historically, the knowledge came from conversations with peers, blog posts, and mailing list conversations. Information scattered around the Internet is hard to cobble together into anything actionable. By the time you do, you might already have been hacked. Maybe that’s why you picked up this book. Clearly web-based attackers are becoming more active and brazen every day, with no signs of slowing.

For a defender to be successful, there is simply no substitute for experience. And this kind of experience comes only from hour after hour, day after day, and year after year of being on the battlefield, learning what strategies and tactics work in a given situation.
This kind of experience certainly doesn’t come quickly or easily. At the same time, this kind of information and the lessons learned can be documented, codified, and shared. This is what Ryan Barnett offers in this book: recipes for defense—recipes for success.

To all defenders, I leave you in Ryan’s accomplished and capable hands. His reputation speaks for itself. Ryan is one of the original defenders. He has contributed more than anyone else in web security to define the role of the defender. And he’s one of the best field practitioners I’ve ever seen. Good luck out there!

Jeremiah Grossman

Chief Technology Officer

WhiteHat Security, Inc.
Web Application Defender’s Cookbook is a consolidation of tutorials called recipes. They are designed to facilitate the expedient mitigation of the most critical vulnerabilities and attack methods that cyber criminals use today. Whether you’re mitigating a denial-of-service attack on your e-commerce application, responding to a fraud incident on your banking application, or protecting users of a newly deployed social networking application, this book gives you creative solutions to these challenges. This book shows you how to be as pragmatic as possible when performing remediation. Although each recipe includes adequate background information so that you understand how the web application attack or vulnerability works, theory is kept to a minimum. The recipes’ main intent is to provide step-by-step countermeasures you can use to thwart malicious activity within your web applications.

I obtained the information in this book through years of protecting government, educational, and commercial web sites from a wide range of adversaries who used an even wider array of attack methods. Because web attack methods employ multiple levels of complexity and sophistication, so do the remediation recipes presented in this book. All the recipes, however, have a foundation in the following skill sets and topics (listed in order of relevance):

- Web application vulnerabilities and attacks
- HTTP
- Perl Compatible Regular Expressions (PCRE)
- Programming (Lua, Python, and Perl)
- Web server administration (Apache)
- Web browsers
- UNIX operating system (shell commands and editing files with vim)
- HTML
- Incident response process
- Networking and TCP/IP
- Basic understanding of Structured Query Level (SQL)

Although the recipes in this book include the key elements of many web application attacks, this book is not meant as a comprehensive guide for describing web application weaknesses, attacks, or secure web application development. Numerous other books and public resources help developers secure their web applications.
Overview of This Book

Quite simply, the goal of this book is to make your web applications more difficult to hack. Web applications—or any software, for that matter—will never be completely secure and free from defects. It is only a matter of time before a determined attacker will find some vulnerability or misconfiguration to exploit and compromise either your site or one of its users. You should take a moment to come to terms with this truth before progressing. Many people wrongly assume that hiring “smart” developers or deploying commercial security products will magically make their sites “hacker proof.” Sadly, this is not reality. A more realistic goal for web application security is to gain visibility into your web transactions and to make your web applications more hacker resistant. If you can force any would-be attackers to spend a significant amount of time probing your site, looking for vulnerabilities, you will widen the window of opportunity for operational security personnel to initiate proper response methods.

This book arms you with information that will help you increase your web applications’ resistance to attacks. You will be able to perform the following critical web application security situational awareness tasks:

- Detect when web clients are acting abnormally
- Correlate web activity to the responsible user
- Figure out if your web application is not functioning properly
- Determine if your application is leaking sensitive user or technical information
- Detect new or misconfigured web application resources
- Create a process to quickly remediate identified vulnerabilities
- Create virtual patches to mitigate identified vulnerabilities
- Respond to different attack methods with various response actions

Who Should Read This Book

The target audience for this book is web application defenders. The term defender in this book refers to anyone who is charged with protecting live web applications from attacks. These people did not create the web application, but they are responsible for administering the application on the live network. Web application defenders are one of the three main communities who contribute to the overall security posture of web applications; the other two are builders and breakers. Builders are the development teams who are responsible for the actual source code and functionality of the web application. They are the initial creators of the application and are responsible for future enhancements and maintenance. Breakers, on the other hand, are the information security teams who assess applications by reviewing source code and attacking live web applications. All three communities
contribute to the overall security of web applications, but this book is solely focused on helping defenders with their appointed tasks.

How This Book Is Organized

This book is organized as a set of recipes that solve specific web application security problems. Many of the recipes are self-contained, meaning that the vulnerability or attack is presented and a solution is outlined that mitigates the entire attack surface. Other recipes, however, individually address only a portion of the larger problem, so it may be necessary to implement multiple recipes together to fully reduce the risks involved.

From the highest conceptual view, this book is organized into three logical parts:

- Part I, “Preparing the Battle Space”
- Part II, “Asymmetric Warfare”
- Part III, “Tactical Response”

The order of these parts is based on the logical flow and dependencies of each topic. If you are initially deploying a new web application, we recommend that you read this book from start to finish. On the other hand, if you already have web applications deployed in production, you can easily jump directly to a recipe that addresses your specific concern. When appropriate, cross-references to other chapters, recipes, or external resources are provided. The following sections describe each part and its chapters.

Part I, “Preparing the Battle Space”

This book begins with the concept of preparing your web application platform for the attacks that will eventually occur. You should complete the recipes in this part either right before or right after you put a new application into production.

- Chapter 1, “Application Fortification,” lists the initial steps you should take to implement attack awareness, gain visibility, and conduct proper audit logging of web transactions. You’ll also learn how to implement centralized data storage to facilitate shared attack intelligence for your organization.
- Chapter 2, “Vulnerability Identification and Remediation,” describes the critical methods of proactively identifying vulnerabilities within your web applications. After these vulnerabilities are identified, you learn how to remediate them through a process known as virtual patching.
- Chapter 3, “Poisoned Pawns (Hacker Traps),” covers various methods of “booby-trapping” your web application to help you quickly and accurately identify malicious clients.
Part II, “Asymmetric Warfare”

After the web application is live on the production network and is exposed to real users, the recipes in this part will come into play. All of these recipes deal with analyzing web transactions for malicious activity.

- **Chapter 4, “Reputation and Third-Party Correlation,”** demonstrates how to leverage third-party geographic data and IP address blacklists to identify known malicious sources or restrict access.
- **Chapter 5, “Request Data Analysis,”** illustrates various methods of data acquisition, normalization, and analysis for inbound HTTP requests. It also discusses how to identify common request anomalies that indicate abnormal request construction.
- **Chapter 6, “Response Data Analysis,”** shows techniques similar to those in Chapter 5, but this time we inspect the HTTP response data. Anomalies found in these recipes often indicate application errors or successful application attacks.
- **Chapter 7, “Defending Authentication,”** describes how to identify brute-force attacks to enumerate user credentials, track successful and failed authentication attempts, and track user applications.
- **Chapter 8, “Defending Session State,”** addresses security issues related to application session management. It discusses attacks such as cookie tampering and session hijacking.
- **Chapter 9, “Preventing Application Attacks,”** highlights the problem of accepting user input to your web application without any type of validation check. It provides protection information for attacks such as SQL Injection, remote file inclusion, and OS commanding.
- **Chapter 10, “Preventing Client Attacks,”** shifts the focus from protecting the web application to defending its users against various attacks, such as cross-site scripting, cross-site request forgery, and clickjacking. It also highlights cutting-edge defenses against banking trojans such as Zeus. Furthermore, it shows you how your web application and the browser can collaborate on security policies.
- **Chapter 11, “Defending File Uploads,”** describes how allowing users to upload files to your web application provides an obvious window for attackers to insert malicious code onto your site. These recipes demonstrate how to analyze file attachment uploads to block malicious data.
- **Chapter 12, “Enforcing Access Rate and Application Flows,”** shows you how to correlate attributes from multiple client requests to identify abnormal request rates, application flow, and usage anomalies.
Part III, “Tactical Response”

After you have identified malicious behavior within your web application, the next logical question is what do you want to do about it. The final part of this book highlights how to best utilize the different response options at your disposal.

- **Chapter 13, “Passive Response Actions,”** shows various application changes that may be dispatched that are imperceptible to the end user, such as increases in internal logging.
- **Chapter 14, “Active Response Actions,”** lays out more aggressive, disruptive actions that may be used against the transaction or end user. These include dropping the connection, temporarily blacklisting the source IP, and forcibly logging out the application user.
- **Chapter 15, “Intrusive Response Actions,”** describes intrusive methods of inspecting information in the client’s web browser to validate functionality or even enumerate additional source information.

Tools You Will Need

This book demonstrates hands-on techniques you can use to prevent web application attacks. Keep in mind that, because you are a web application defender, your tool set is not identical to that which is afforded to the web application developer. Your main tool is the web application platform itself, such as Apache’s web server or Microsoft’s Internet Information Services (IIS) web server. Unfortunately, the effective mitigation of web application vulnerabilities and attacks relies on advanced logic and analysis capabilities that normally are not present in standard web server software. To achieve the advanced defensive capabilities outlined in this book, you need to install additional software within your web server as an additional module or filter.

The recipes in this book use the open source web application firewall (WAF) module called ModSecurity,¹ which is available for the Apache, Microsoft IIS, and Nginx web server platforms. You have other options for achieving basic input filtering for your web application, but no other defensive module has the same level of advanced capabilities as ModSecurity. These capabilities include the robust rules language, data modification, content injection, and even a Lua application programming interface (API), which facilitates custom logic and integration with other tools. By the end of this book, I am confident you will agree that ModSecurity is an outstanding tool for web application defense.
Nevertheless, we include appropriate pointers and references to other tools that may provide similar functionality.

Although the recipes use ModSecurity, the underlying detection techniques may certainly be adapted for use within applications that use other tools or libraries. For instance, the OWASP AppSensor project (to which I am a contributor) includes Java code examples that implement many of the same concepts presented in this book.

The target web applications that you will protect throughout this book are taken from the OWASP Broken Web Application (OWASP BWA) project. The virtual machine image available for download provides a large number of intentionally broken web applications used for testing and learning about web application attacks and vulnerabilities. It also provides many real-world applications (such as WordPress) that have real flaws. If you want to follow along with the recipes in the book and practice hands-on with implementing the protection mechanisms, download the virtual machine image.

1 http://www.modsecurity.org/
3 http://code.google.com/p/owaspbwa/

Conventions

To help you get the most out of this book and to keep track of the information contained in the recipes, we use several conventions throughout the book.

RECIPE X-X: RECIPE TITLE

The recipes solve specific web application security problems, demonstrate the use of a tool or script, and usually provide sample debug and alert messages. Each recipe starts with a list of ingredients like the following:

- External software reference
- Some rules files to download
- A third-party data file to install

A recipe may include endnotes that reference further reading material on the topic.

NOTE

Notes include tips, hints, and tricks to consider with each recipe.