IMMUNIO

Real-Time Web Application Security

Protect your customers and your reputation

EXECUTIVE SUMMARY

The internet is essential to almost every mobile phone; mobile applications connect to the cloud to store everything from photos to passwords and personal information. Now medical devices, home appliances, and cars are joining in, thanks to the Internet of Things – making application security on the web potentially truly a matter of life and death.

Everything happens so fast on the internet today that it often feels as if we can never catch up. Instant gratification is the order of the day, and if your app isn't ready and waiting, users will quickly move on to one that is. To meet these heightened user expectations, companies need to accelerate app delivery - often to the point of automating several new releases a day. This means new agile development processes - and intensified attention to threat prevention and response, because the bad guys are also operating at high speed, with automated processes and collaboration among attackers, hunting down and hammering on every vulnerability they can find. They're better organized - so developers must be, too.

This paper reviews the current threat landscape for web applications and the approaches available to organizations to secure those applications. It compares one of the most common approaches – Web Application Firewalls (WAFs) - and compares it with the most recent addition to the application protection arsenal, Run-Time Application Self-Protection (RASP). Finally, it takes a deeper dive into the IMMUNIO RASP solution, and concludes with some hints and tips on how to determine what application security solution would best suit your needs.

We hope you will find it useful, and welcome any comments or questions you have via email at contact@immun.io.

INTRODUCTION

Web applications are central to our lives today, both business and personal. Fortunately for us, building web apps has never been easier. There are thousands, perhaps even millions, of developers around the world using open-source platforms to build apps in weeks, not the months or years it takes to build traditional applications. According to NetCraft, there are over 175 million web apps publicly available – and probably a whole lot more behind corporate firewalls.

We work in the cloud, we store our valuable documents in the cloud, and of course we do our best to protect everything. But it is as cheap and easy for the bad guys to spin up servers in the cloud and throw out cyberattacks as it is to create those apps that we love, so protecting anything in the cloud is a Sisyphean challenge.

WhiteHat Security publishes an annual report tracking web vulnerabilities and time-to-fix; the 2015 report found that organizations in which the primary concern for website security was risk reduction (which is likely the case for the majority of app-hosting sites) had an average of 23 vulnerabilities per site, with a remediation rate of 18% - not statistics that would give those companies' customers a sense of security, especially when the report goes on to peg the average time-to-fix for those sites at 115 days.

The threats are changing and growing every day. Web applications are vulnerable to SQL injection attacks, cross-site scripting, cross-site request forgery, denial of service, and brute force assaults, any or all of which can lay individual user information or entire corporate databases open to theft and abuse. Add in the proliferation of mobile apps and the Internet of Things, mix in a little noncompliance with PCI and other mandated security requirements, and another information security nightmare is born.

But applying protection – whether it's basic anti-malware or enterprise-grade firewalls – to a production environment is a bit like shutting the stable door after the horse has bolted. What we really need is to build protection into the applications themselves. That's where Run-time Application Self-Protection (RASP) comes in.

RASP is designed specifically to provide the additional protection needed in the fast-moving, dynamic world of web application security by building that protection into the app without requiring any changes to the application code. This paper explores the current state of web application security and explains the beneficial role RASP technology, and specifically IMMUNIO, can play in keeping customers, partners, and the developers themselves safe.

TODAY'S THREAT LANDSCAPE

The 2015 Data Breach Investigation Report from Verizon found that, for the first time, organized crime was the leading bad actor in attacks on web applications, with financial gain being the primary driver of those attacks, 98% of which were opportunistic in nature and 95% of which involved harvesting credentials from user devices. Opportunistic attacks have one common factor – taking advantage of someone's inattention to detail. For the most part, that means users not being aware of a phish or vulnerabilities in application code.

That's not to say that cross-site scripting and SQL injection attacks aren't happening any more. In 2014, SQL injections were responsible for 8.1% of all data breaches, making them the third most popular attack method after malware and distributed denial-of-service attacks. In all of the top ten application attack types recorded by the Open Web Application Security Project (OWASP) described below, attackers were able to manipulate application input and obtain confidential data without being detected by network defense systems.

- **SQL Injection** At the top of the list, SQL injections let attackers modify a back-end command through unprotected user input.
- **Broken Authentication and Session Management** Several types of programming flaws allow attackers to bypass the authentication methods used by an application.
- **Cross-Site Scripting** lets attackers insert Javascript into trusted sites, altering the contents of those site to, for example, send a user's credentials to a remote server.
- **Insecure Direct Object References** such as path traversal enable the bad guys to extract data from a server by manipulating file names.
- **Security Misconfiguration** is a dangerous category of flaws that deals with the deliberate misconfiguration of a server or of the application itself.
- **Sensitive Data Exposure** If web applications do not properly protect sensitive data in motion and at rest, attackers can steal or modify the data to conduct credit card fraud, identity theft and other crimes.
- **Missing Function Level Access Control** covers situations in which higher-privilege

functionality is hidden from a lower-privilege or unauthenticated user, giving an unauthenticated user administrative access to a Web application.
- **Cross-Site Request Forgery**, often used in conjunction with social engineering, allows attackers to trick users into performing actions without their knowledge.
- **Using Components with Known Vulnerabilities** Attackers can easily exploit old third-party components because their vulnerabilities have been publicized, enabling cybercriminals and script kiddies alike to take advantage of these flaws with ease.
- **Unvalidated Redirects and Forwards** are used in phishing attacks in which the victim is tricked into visiting a malicious site through trusted site URL manipulation.

In Verizon's overview of confirmed data breaches, and setting aside those demonstrably caused by human error, web application attacks accounted for 9.4% of malicious data breaches, second only to the use of crimeware, which accounted for 18% of the breaches. In a breakdown of Critical Security Control preventative measures, Verizon recommended two-factor authentication and patching of web services (the speed at which vulnerabilities are remediated after discovery) as being the most critical courses of action in 48% of the data breaches analyzed.

WHY ARE WEB APPLICATIONS BEING TARGETED?

As we've already noted, web applications are the fastest-growing segment of software development, with the shortest Software Development Life Cycle (SDLC). Time-to-market is critically short, and what would in earlier days have been beta software is now routinely deployed as production code. That can often lead to short cuts in QA and testing which in turn leads to too many vulnerabilities in the field.

The application layer is also the hardest to defend in the field. Vulnerabilities encountered here often rely on complex patterns of user behavior that are not always easy to test for, and rely on knowledge that may not be available at every stage of the testing process. If testers do not know what represents expected user behavior, how can they identify what constitutes suspicious behavior?

This also means that vulnerabilities are frequently specific to each application – everyone is different, and everyone uses applications slightly differently – and so they are unknown to typical security systems like WAFs. A skilled attacker can easily find these vulnerabilities and exploit them without being detected.

The end result is that it's extremely difficult for application developers to track down how attackers are getting in. If there's a web server vulnerable to SQL injection, an open admin application login page, a flat network, API calls to old versions of third-party libraries, and little or no event logging, how can they determine the weakest point?

Half of firms hit by web application security breaches
A survey by Forrester Consulting revealed that 51% of organizations have had at least one web app security incident since the beginning of 2011.
More than 50% of organizations have experienced at least one web application security incident over an 18-month period. Forrester's Software Security Risk Report found that, of the 51% of web application development companies that suffered application security incidents, 18% lost more than $500,000. Another 8% saw losses in excess of $1 million, and two firms reported losses of over $10 million

Web applications are also prime targets because they are externally-facing and therefore present a larger attack surface. Unfortunately, Forrester also found that less than 42% of respondents followed secure coding guidelines, blaming a lack of resources and high demands for new code. 71% of the organizations that had experienced an attack said they lacked security technologies appropriate for development, and 79% said that security processes could not scale with the volume of code being produced.

THE PEOPLE FACTOR

This leads us to another major and continuing issue in web application security – silos. Developers and security people don't, on the whole, live in the same universe, and they are driven by different needs.

OWASP categorizes the players in the application protection space as Builders (developers) and Defenders (information security). While these groups are collaborating more closely than they have in the past, there's still a long way to go.

According to the System Administration, Networking, and Security (SANS) Institute 2105 report The State of Application Security: Closing the Gap, 47% of respondents felt that the effectiveness of their application security programs needed improvement. Builders are focused on delivering features and meeting time-to-market expectations rather than on making sure software is secure, and security people are focused on risk management; neither Builders nor Defenders are sufficiently familiar with each other's environment to feel confident they can fix vulnerabilities in the software without causing more problems.

Additionally, when it comes to the use of third-party libraries in web applications, a third of all Builders surveyed by SANS relied on "self-attestation by the vendor" for the security of that third-party code, a disturbing statistic given that almost 80% of the responding Builders rely on third-party libraries or open-source code.

These challenges are magnified by the rapid pace of web application development and lack of control over applications hosted in the cloud.

WHAT ARE ORGANIZATIONS DOING NOW TO PROTECT APPS?

One of the most important steps any organization can take to improve its application security profile is to break down the silos between development and information security, cross-training each to better understand the challenges and needs of the other. The SANS report indicates that this is becoming a more common practice, but it's not a luxury available to many of the smaller developers, who may well have just one multi-purpose development team.

To be fair, web application developers are not ignoring the security threat – far from it. They're using a plethora of testing techniques during the development cycle to identify as many potential vulnerabilities prior to release as they can.

Application security technologies like Dynamic Application Security Testing (DAST), Static Application Security Testing (SAST), and Interactive Application Security Testing (IAST) all test apps for security vulnerabilities and provide remediation guidance. But no test is going to find every vulnerability or be able to predict every attack vector encountered by an app in the field. Once an app is deployed, responsibility for security falls largely on external devices like Web Application Firewalls (WAFs) and Intrusion Prevention Systems (IPS). However, external protection sees only traffic and user sessions; it cannot detect whether application configuration, logic, or data flow are being tampered with. And the teams running network security have little or no insight into how the developers expect those applications to behave.

The pace of software development is accelerating rapidly as the demand for more and better web apps keeps on growing. Agile development, DevOps automation, and integration of deployment are all leading to new challenges that existing security technologies are ill-equipped to support:

- **Static Application Security Testing (SAST)** analyzes code for security vulnerabilities, but only when the code is not running. It's generally used to find security flaws that are hard to find through manual code reviews; to do this, it needs to perform complex and deep analysis of the code across many dependencies, making it very time-consuming – hardly an ideal solution for a dynamic web environment. Support for software development frameworks is often inadequate. To be more useful in the age of web applications, these tools need to focus on the simple, more error prone methods that can be delivered in near-real time, directly in the development environment.
- **Dynamic Application Security Testing (DAST)** requires a running application and access to an automation script for the proper authentication and authorization of test scripts. A rapid development cycle and adoption of dynamic technologies can break dynamic scanners, significantly reducing the value of the scans. Dynamic analysis tools need to become more resilient, provide instant feedback, and ideally start delivering protective services like RASP.
- **Interactive Application Security Testing (IAST)** tools are emerging as a result of DAST tool evolution, conducting behavioral analyses of applications and addressing areas such as logic and data flow. This kind of run-time analysis is much closer to the kind of vulnerability assessments required for web-based applications.
- **Web Application Firewalls (WAFs)** rely on signatures and application-specific communications and protocols – HTTP, XML, SOAP, etc; they (or rather the humans managing them) can't keep up with the frequent software releases and code changes of the web application world. For this reason, they are used primarily in monitoring, rather than intervention, mode to satisfy PCI compliance requirements. Specialist security research firm Securosis has undertaken much useful research in this area – see the Recommended Reading section at the end of this paper for links to WAF-specific documents.

Post-deployment approaches like penetration testing and bug bounty bashes will continue to grow, but simply can't scale with the volume and pace of software development in the web environment.

THERE IS A BETTER WAY

Run-time Application Self-Protection (RASP) is different. It protects applications from the inside – walling off vulnerabilities and preventing information leakage without impacting the functionality of the application. The beauty of this approach is that it's based on behavior rather than fixed physical characteristics; there are no lists to keep updated and no time-consuming database lookups.

RASP technology protects web apps when and where it matters - in the application's run-time environment. In effect, it picks up where xAST and WAFs leave off, turning suspected vulnera-bilities into affected code segment identification and informing developers exactly what they need to fix. What's more, because it's operating at run-time, it can "fence off" that vulnerable segment until remediation has taken place, providing real-time protection against potential exploits leveraging that vulnerability.

Using RASP does not negate or cancel out the use of those other technologies; developers still need to test for and fix as many vulnerabilities as possible before deployment. Firewalls and other perimeter defenses should still monitor network access, along with identity or access management systems. RASP adds that extra layer of protection to enable applications to secure themselves in real time, in the production environment, protecting customers and the organization itself.

In fact, for organizations already using WAFs, the benefits of adding RASP will quickly become evident. Where WAFs take a broadbrush approach to security, issuing an alert whenever any untoward action is suspected, RASP will take that suspicion and zero in on it, enabling developers to take any necessary remediating action quickly. Working as it does in real time, RASP really is an ideal security layer for cloud-driven environments, mitigating the security concerns that remain one of the biggest barriers to wholesale cloud adoption.

According to WhiteHat Security, the best way to lower vulnerabilities, speed up time-to-fix, and increase remediation rates is to feed vulnerability results back to development as quickly as possible. By pinpointing the exact place in the code where a vulnerability exists, RASP can reduce that vulnerability gap to a matter of hours or days rather than the weeks or months that are today's reality.

Be aware, though, that not all RASP implementations are created equal. While all RASPs are embedded inside the applications they're protecting, some rely on API calls, which make it difficult to protect code "in the field", because they require code changes. Others come with their own virtual machines, which can get problematic if operations are built around Java. Others, including IMMUNIO, use an agent that's installed alongside the web server, making it accessible to admins, and work by hooking functions and methods in the web server code, not by changing the code itself.

COMPARING RASP AND WAFS

In late 2015, IMMUNIO set out with application security expert John Stauffacher and his team at Caffeinated Networks to compare WAFs and RASP. John's team set up a prototype real-world environment named DoJo College, an online provider of a multi-tenanted hosted Learning Management System (LMS), to evaluate the strengths and weaknesses of the two approaches. The scenario was as follows:

- DoJo's software is designed to enable educational institutions and corporations to host their own LMSs within the larger DoJo.College system. The software is all developed in-house by an Agile team.
- DoJo has limited funding and a laundry list of projects to complete in a finite amount of time, the first of which was a complete overhaul of their website, through which their application interfaces with the rest of the world.
- Since Dojo needed to focus on providing customer functionality, not becoming security experts, they made the decision to use third-party technology to embed application security into the site.

To meet their needs, DoJo came up with the following core requirements for a solution to secure their web application environment:

- Must not impact normal operation and functionality
- Performance must remain well within standard response times
- Must provide actionable near real-time intelligence and reports to enable rapid threat response
- Must address the run-time portion of the OWASP Top Ten
- Does not require specialized skills for installation or ongoing maintenance

The shortlist of solutions with the potential to meet these requirements were two well-regarded WAFs -CloudFlare and Incapsula – and IMMUNIO's RASP solution. Here's a quick overview of the results; you can view the full case study video here.

Objective #1: Ease of installation
While IMMUNIO only required the team to create an account and deploy a Gem file, CloudFlare required moving the entire DNS infrastructure to the CloudFlare servers. Incapsula, while simpler than CloudFlare, still required administrator-level access to install DNS records, potentially requiring another team to get involved.

Objective #2: Clear, concise, actionable reporting
CloudFlare classified every detected attacks as a Bad Browser, no matter what it was – just like the security guard who's only looking for guns – and did not show what action had been taken. Incapsula was unable to provide information in real-time, rendering it useless for securing a hosting environment. IMMUNIO's reporting pinpointed in real time the template filename, line number and even block of code containing the vulnerability; information on the attackers was lacking, but this failing has since been addressed.

Objective #3: Attack body of knowledge
Both IMMUNIO and Incapsula performed well; IMMUNIO was uniquely able to detect LFI/RFI and RCE attacks, while Incapsula provided comprehensive support for anti-DDoS and offered IP reputation services; however, Incapsula was still dependent on signature sets for detection. CloudFlare security was significantly less comprehensive, relying mostly on Mod_Security rulesets.

Objective #4: Completeness of solution
To determine the full extent of coverage, we measured how each solution responded to the

OWASP Top 10. Both CloudFlare's and Incapsula's detection rates were very low, with much manual tuning required to register even those levels. IMMUNIO, on the other hand, rebuffed every attack and, importantly, did not offer attackers any way to evade detection, since it does not rely on signatures.

Objective #5: Performance
Performance was monitored over a period of normal traffic flow, with no attack vectors. IMMUNIO delivered far fewer spikes in performance, again a desirable attribute in a hosting environment.

Objective #6: Overall testing results based on the OWASP Top 10
When Dojo crunched the numbers and ran back through each attack scenario, they found that only IMMUNIO not only caught the majority of attacks but also did not waste time on false positives. There's a detailed breakdown of how each solution fared in the full report.

The DoJo team could not have been happier with the results of their evaluation. They needed a broad and deep solution that would give them the intelligence they needed to remediate vulnerabilities quickly, and they found it in IMMUNIO. The solution provided all the assurances of a WAF and comprehensive coverage of the OWASP Top Ten, as well as effectively handling false positives. The level of insight into vulnerabilities was on a par with specialized penetration testing and static source code analysis, neither of which was an option for a small team like DoJo.

Relying on a WAF to protect your apps is not unlike protecting a building against intruders by telling the security guard to keep out anyone not on the approved list. But if an individual looks trustworthy, the guard has nothing on which to base a smart admit/exclude decision, no way to fine-tune the list on the fly, and no way to apply that include/exclude decision on the fly across multiple entry points. The better approach would be to have motion detectors in every room, monitoring movement and perceived intention by how the intruder moves about the room. When the intruder attempts to enter a designated private space, the alarms go off and the gates come down. That's how RASP does application security.

THE IMMUNIO SOLUTION

At IMMUNIO, we believe the time is long past when you can rely on whitelists and blacklists to keep applications secure. – it's simply not a viable solution in a warp-speed web-based world. IMMUNIO even goes further than first-generation RASP technologies by providing:

- More extensive coverage of different vulnerabilities
- Code-level visibility into attacks
- Breadth of platform support

The IMMUNIO service is designed specifically to secure web assets in the cloud and within local networks, protecting your customers' data and your business. A self-contained agent runs inside the process of your application, without requiring any code changes in the application itself. Whenever your app is exposed to a malicious attack, the agent identifies the attacker and type of attack and blocks it. User data is protected and you have the insight you need into the coding vulnerabilities you need to fix.

HOW IT WORKS

IMMUNIO is based on patented runtime self-protection technology. The agent is self-contained and independently protects its application, even if it becomes disconnected from the IMMUNIO service. User data is never exposed outside the agent, ensuring your apps remain in compliance with data protection mandates.

In addition to securing your customers and your applications, IMMUNIO enables development teams to quickly identify and prioritize vulnerability remediation efforts by providing vital information about the identity and severity of attackers.

The service

- Disrupts automated brute force attacks by serving up captchas
- Blocks sensitive data from being exposed by injection attacks
- Secures applications with known vulnerabilities until remediation resources are available
- Secures hard-to-monitor/hard-to-manage applications, for example when hundreds of web

apps are running simultaneously on an internal network

KNOWLEDGE IS POWER

IMMUNIO agents deliver information on the time, origin, and type of every attack on your apps to a central reporting point. This information builds over time into a broad profile of attacks impacting your networks, enabling your organization to map trends and deploy appropriate resources.

The service lets you monitor and review exploitation attempts across an unlimited number of applications. Attack details are propagated across your infrastructure, so an attack detected on one application is immediately flagged on every app server and for every monitored app in your account.

IMMUNIO also gives your developers full visibility into how the vulnerability in your code would have been exploited, including a stack trace down to the line of code (for SQLi, XSS, and RCE threats), reporting of request parameters, and how your app's behavior would have been modified.

We are continually reviewing and updating our technology, and welcome input from the user community regarding improvements to our approach.

NEXT STEPS

Before you jump into the RASP arena with both feet, we recommend you take a long, hard look at what you need from an application security solution. Here are some questions to ask yourself – and any security consultants or vendors you're working with – to get you started:

Understand your application

- Where are your customers, geographically?
- Have you mapped out their ideal and possible routes through the application, so you know where logins should be coming from, or at what points users are interacting directly with your code?
- What data and parameters is the app expecting from users, and can they take shortcuts to

the end of the workflow?
- What do you know about the libraries, Gems, and other third-party packages your app is calling?
- Have you (and do you continuously) vet that third-party code and acceptable URLs?

Understand your resources
- What human resources are available to monitor and manage the security of your applications?
- Do you have a separate web security team, or are you expecting your developers to take care of security?
- Are you deploying your apps from in-house server resources or working in a hosted environment?
- Does the solution require any specialized skills for installation and ongoing maintenance?

Understand your needs
- What level of uptime or availability do your customer SLAs dictate?
- When the solution identifies a threat or attack, what do you want it to do?
- Do you need to know when and where the threat was attempting to get in to the app, and how quickly do you need to know?
- Do you need to protect your customers' data while you're fixing the vulnerability?
- What percentage of false positives are you able to tolerate?
- If you're ready to take a look at IMMUNIO and experience directly how it can add an extra layer of protection to your application security without getting in the way of your developers or your business, head on over to www.immun.io/free-trial.html and request your no-strings trial today.

ABOUT IMMUNIO

With offices in the US and Canada, IMMUNIO is dedicated to helping organizations secure web applications, protect their users, and prevent information leakage through code vulnerabilities. Learn more at www.immun.io.

Recommended Reading:

- Runtime Application Self-Protection: A Must-Have, Emerging Security Technology, Gartner, Inc, 2012
- Runtime Application Self-Protection: Technical Capabilities, Gartner, Inc, 2012 (reviewed in 2014)
- Stop Protecting Your Apps; It's Time for Apps to Protect Themselves. Gartner, Inc, 2014
- Hype Cycle for Application Security, Gartner, Inc, 2015
- OWASP free guide to secure web application development
- The State of Application Security: Closing the Gap, SANS Institute, 2015
- Pragmatic WAF Management: Giving Web Apps a Fighting Chance, Securosis, 2104
- Maximizing Value from your WAF, Securosis, 2016
- 2015 Data Breach Investigations Report, Verizon, 2015
- Website Security Statistics Report 2015, White Hat Security, 2015

Canada - Corporate Headquarters
3 Place Ville Marie, Suite 400, Montreal, Quebec, Canada H3B 2E3

United States - Portland
220 NW 8th Ave, Portland, OR 97209

United States - Boston
745 Atlantic Ave, Boston, MA 02111