Global Information Assurance Certification Paper

WEB APPLICATION FIREWALLS

Enterprise Techniques

GIAC (GCIA) Gold Certification

Author: Jason Pubal, jpubal@mastersprogram.sans.edu
Advisor: Barbara Filkins

Accepted: March 13, 2015

Abstract

For years, attackers have assailed network and system level vulnerabilities, fueling demand for products like firewalls and intrusion detection systems. As these products mature and IT security teams learn to better handle network security, the information security industry is seeing a visible increase in attacks moving up the stack to target application-level vulnerabilities.

Web application firewalls protect web applications from attackers. They sit in front of web applications, monitor activity, and block malicious traffic. This paper describes how a web application firewall works and covers different deployment options. It goes into drivers for organizations to adopt web application firewalls, such as reducing the effort required to remediate vulnerabilities in production web applications, speeding up risk reduction, protecting web applications that may not be able to be fixed such as commercial or legacy applications, and the need to meet compliance requirements such as PCI DSS. It then takes a deep dive into how to reduce the window of vulnerability exposure through virtual patching, and how a web application firewall fits into an overall security monitoring and incident response strategy.

A lab environment was created to experiment with and demonstrate shielding in web application vulnerability management via virtual patching and security monitoring using ModSecurity, an open source web application firewall. Using the lab exercise results as an example, this paper goes into depth about using a web application firewall as a means for virtual patching and as a sensor in a network security monitoring infrastructure.

Introduction

For years, attackers have assailed networks and exploited system level vulnerabilities, fueling demand for products like firewalls and intrusion detection systems. As these products mature and IT security teams learn to better handle network security, the security industry is seeing a visible increase in attacks moving up the stack to target application-level vulnerabilities. These attacks are aimed at exploiting weaknesses in the web application software itself, not those at the network or server level. Traditional defenses such as network firewalls and intrusion detection systems are unable to offer comprehensive protection.

Internet facing web applications make up a large part of the attack surface, and are where attackers are focusing their attention. According to Verizon's Data Breach Investigations Report (2014), 35% of breaches were caused by web application attacks, making it the most prevalent attack pattern.

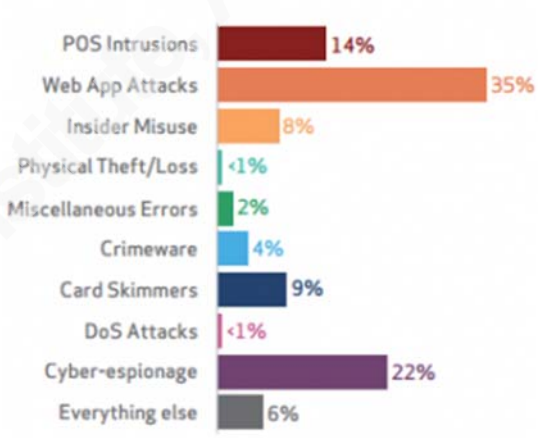

Figure 1. DBIR Frequency of Incidents. From "2014 Verizon Data Breach Investigations Report," 2014.

Web application firewalls (WAFs) protect web applications from attackers. They sit in front of web applications and block malicious traffic. But, web application firewalls can be used for more than just blocking generic malicious traffic – they can be used in a web application vulnerability program to drastically reduce remediation time without having to change application source code. As part of a security monitoring infrastructure,

Jason Pubal jpubal@mastersprogram.sans.edu

WAFs can also increase visibility into application traffic far beyond what is possible with a firewall or intrusion detection system.

Enterprises looking to manage the risks of web applications will benefit from these techniques. A lab environment was created to experiment with and demonstrate virtual patching and security monitoring using ModSecurity, an open source web application firewall. Using the lab exercise results as an example, this paper gives an in-depth view of using a web application firewall as a means for virtual patching and as a sensor in a security monitoring infrastructure.

1.1. Web Application Firewalls

WAFs are designed to protect web applications. WAFs are a shielding safeguard intended to defend applications accessed via the hypertext transfer protocol (HTTP). They are capable of preventing attacks that network firewalls or intrusion prevention systems cannot. WAFs sit in front of a web application, monitor application activity, and alert on or block traffic that is malicious or that does not comply with specific rules. The intention is to catch application level attacks, such as SQL injection and cross-site scripting, along with attempts to manipulate web application behavior.

This shielding technology does not require modification of the application source code. WAFs can reduce risk without actually fixing the underlying vulnerability. In cases where it might take a long time or it is infeasible to fix the vulnerability in code, a WAF is useful in protecting against attacks targeting the vulnerability.

1.2. Negative and Positive Security Models

WAFs compare requests to generic attack signatures and application specific policies for the web application being protected, and alert on or block violations. A web application firewall can follow a positive or negative security model to develop security policies for an application.

A positive security model defines what is allowed and rejects everything else. For example, when performing input validation, use of the positive security model would only specify the allowed characters as opposed to trying to filter out all of the potentially malicious input. This is referred to as whitelisting. Only allowing what is known as good

is generally considered to be the more secure approach. A positive security model can be achieved using strict content-validation polices. According to the Web Application Firewall Evaluation Criteria, it "can be more efficient (fewer rules to process per transaction) and more secure, but it requires very good understanding of the applications being protected. The positive security model can also be more difficult to maintain if the web application changes frequently" (2006, p. 11).

A negative security model defines what is disallowed and implicitly allows everything else. This is referred to as blacklisting. The negative security model is achieved by compiling a list of attack signatures, comparing web traffic against those signatures, and blocking traffic that matches. Blocking only what is known as bad is considered the more functional approach from a business perspective. According to Murphy and Salchow, "negative security policies do not take into account how the application works, they only notice what accesses the application and if that access violates any negative security patterns" (2007, p. 2).

Because the negative security model does not need to know anything about the application, it offers out-of-the-box protection. However, the negative security model does not provide protection against unknown attacks. Although considered more secure, the positive security model takes time and effort to setup, requires deep knowledge of the application, and has to be maintained as the application changes. To help with that learning curve, some WAFs have the ability to learn the application. They watch traffic to the application for a period of time to determine normal application behavior and inputs, creating a map of URLs and parameters. All WAFs use either a positive or negative security model, or some use a combination of both.

1.3. Deployment Options

Web application firewalls have a number of architectures and operating modes. These vary in ease of deployment and resulting WAF functionality.

- **Reverse Proxy** – In reverse proxy mode, the WAF sits inline and has an IP address. Incoming connections to the application are sent to the WAF, which makes a separate request to the web server. Encrypted connections are terminated at layer 7 letting the WAF decrypt and analyze all of the

web traffic. This allows for a more feature rich WAF deployment as it gives the WAF complete control over the traffic enabling it to rewrite content and inject security mechanisms per policy. The architecture is shown in Figure 2. The WAF is the device with the shield, and is sitting in-line between the firewall and the web server.

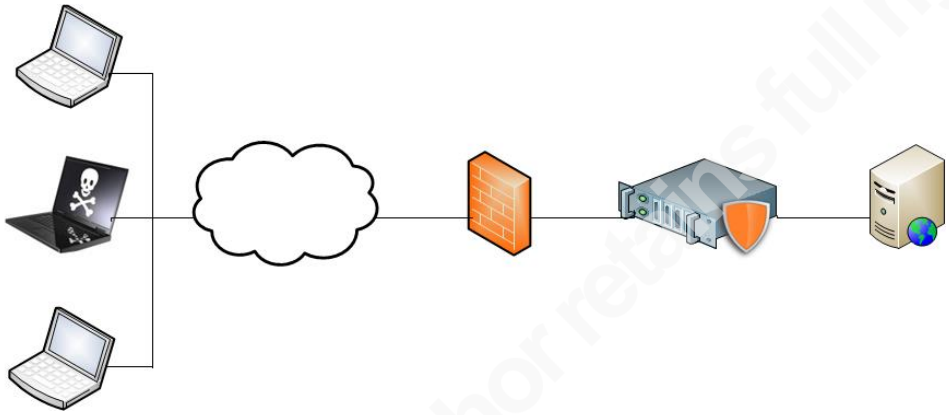

Figure 2.Reverse Proxy or Layer 2 Bridge WAF.

- **Layer 2 Bridge** – In layer 2 bridge mode, the WAF sits in-line and acts as a layer 2 switch. The WAF performs passive SSL decryption, and is able to block traffic by simply dropping the offending packets. This allows for higher performance than reverse proxy with less significant network changes. However, not terminating and replaying the traffic limits some WAF functionality because it cannot rewrite elements per policy. In addition, the WAF will not be able to decrypt traffic using Diffie-Hellman cipher suites in a forward secrecy implementation. Architecturally this looks like reverse proxy, and is also shown in Figure 2.
- **Out-of-Band** – In out-of-band mode, the WAF is not in-line. It gets a copy of the traffic via a monitoring port on a network device. It can passively decrypt SSL traffic, and has the same Diffie-Hellman limitation that bridge mode suffers. The WAF's ability to block traffic is further limited – it can only send TCP-reset packets to interrupt traffic. This mode has the least amount of impact on the network and application, and it allows the WAF to be configured to only alert on malicious traffic

eliminating the danger of blocking false-positive detections or failing closed and causing an application outage. The architecture is shown in Figure 3. The WAF is sitting off of a switch, and is getting a copy of traffic going to the web server.

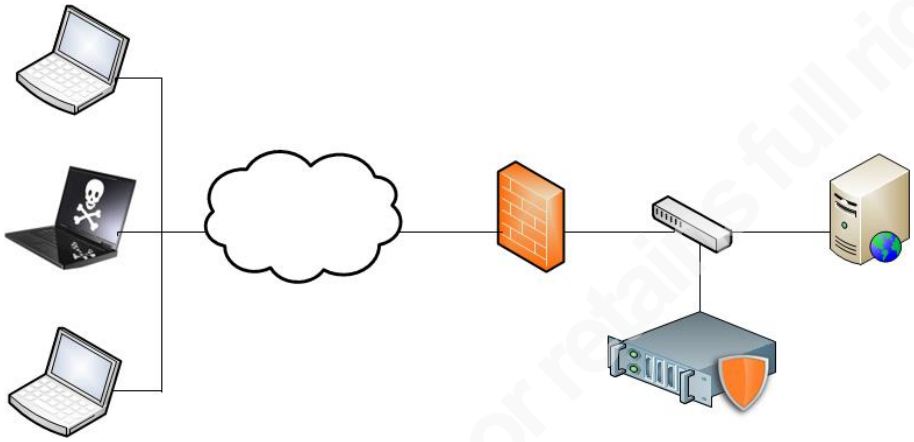

Figure 3.Out-of-Band WAF.

- **Server Resident** – A server resident or embedded WAF is software installed on the host running the web server. This could be implemented as an independent application, or a web server plug-in. Server resident WAFs are not as functional as their network appliance counterparts, but remove the additional network point of failure. They put extra load on the server, so it is important to take a good look at server resource utilization prior to implementation. The architecture is shown in Figure 4. The WAF, represented by the shield, is functioning as software on the web server.

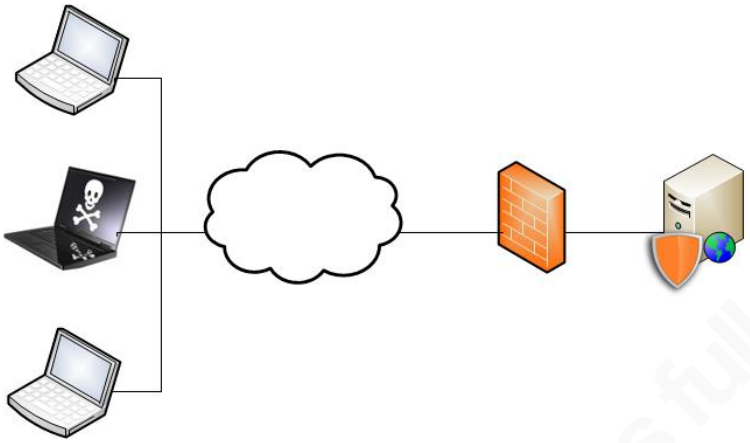

Figure 4. Server Resident WAF.

- **Internet Hosted / Cloud** – A newer but increasingly popular option is using a cloud provider to implement a WAF solution. Gartner predicts that by "2020, more than 50% of public web applications protected by a WAF will use WAFs delivered as a cloud service or Internet-hosted virtual appliance – up from less than 10% today" (D'Hoinne, Hils, Young, and Feiman, 2014, p.1). This works like the reverse proxy option – public DNS is configured to point to the cloud provider, which then creates another connection to the web property. Since the implementation details are not under corporate control, one needs to review any compliance requirements and make sure they are met by the cloud provider. Like other cloud services, review the service level agreements and make sure they are acceptable. The architecture is shown in Figure 5. The WAF, represented by the shield, is external to the corporate environment and is software as a service (SaaS) in the cloud.

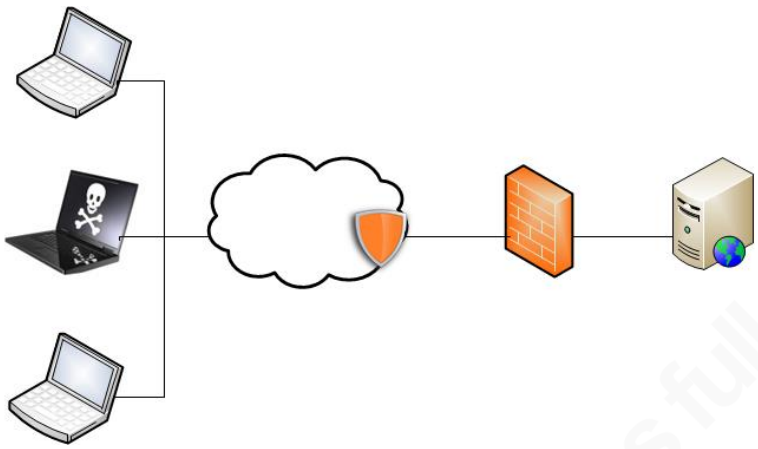

Figure 5. Cloud WAF.

1.4. WAF Drivers

1.4.1. Protecting Production Applications

The best way to secure an application is to develop it using a mature, secure software development lifecycle (SDLC). The earlier in the SDLC a security issue is addressed, the cheaper it is to remediate. Conversely, the later in the SDLC a security issue is found, the more expensive it is to fix – the worst case scenario being a security vulnerability that is found once an application is live in production. "The main benefit of a WAF is the subsequent protection of completed, production applications on the application level with a reasonable amount of effort and without having to change the application itself" (Dermann, M., Dziadzka, M.,Hemkemeier, B., Hoffmann, A., Meisel, A.Matthias Rohr, M., et al, 2008, p. 10). That is, a WAF can reduce the time and cost of remediating the risk of vulnerabilities in live, production-level web applications.

1.4.2. Protecting Legacy and COTS Applications

Generally, companies have access to the source code of applications that were developed in house and are live and in production. While it is more expensive, it is possible to remediate the security vulnerability by fixing the source code. If the application is commercial off the shelf (COTS) or a legacy application, an organization's ability to address issues in code could be minimal to nonexistent. For legacy applications, the organization may no longer have developers for the application or even have developers who know how to write code in the language the application was originally

developed. For a COTS application, the organization can disclose the problem to the third party, but are at the mercy of the vendor's priorities and timelines. "With decreasing access to the web application – and depending on its importance and complexity – the benefits stemming from the use of a WAF grow rapidly. Using a WAF often results in the least additional work for the required security level" (Dermann, M., Dziadzka, M.,Hemkemeier, B., Hoffmann, A., Meisel, A.Matthias Rohr, M., et al, 2008, p. 17).

1.4.3. Vulnerability Management

Shielding is an integral part of the vulnerability management process. According to Nicolett, "shielding should be used to protect vulnerable assets until mitigation is complete" (2005, p. 5). When a vulnerability is discovered in an application, a WAF signature can be created to block attacks against it. Whether the application was developed in house or is a COTS application, this protects it while the vulnerable code is fixed or an organization waits for a vendor to provide an update. This is referred to as a virtual patch.

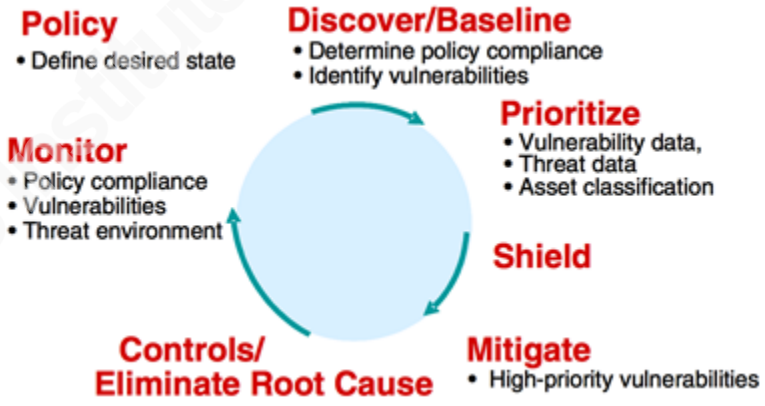

Figure 6. Vulnerability Management Process. From "How to develop an effective vulnerability management process" by Nicolett, M, 2005. Gartner.

Some web application firewalls help automate virtual patching by integrating with dynamic application security testing (DAST) tools. Some WAFs can import DAST results, and automatically generate signatures that protect against vulnerabilities the DAST tool found. According to Feiman, "accuracy of the WAF increases when DAST

passes to it detected security vulnerabilities and attack patters, so WAF can terminate sessions that match malicious patterns" (2013, p. 3).

1.4.4. Compliance

Compliance is a big driver for the adoption of WAFs. The Payment Card Industry Data Security Standard (PCI DSS) is an information security standard for organizations that handle cardholder information for debit and credit cards. Defined by the PCI Security Standards Council (SSC), the standard was created to increase controls around cardholder data to reduce credit card fraud caused by the exposure of that data. Organizations that accept, store, processe, or transmit credit card data must comply with the PCI DSS. PCI DSS version 1.2 added web application firewalls as an alternative to web application vulnerability assessments in 2008. Now in version 3, PCI DSS requirement 6.6 reads (2013, p. 59):

> *For public-facing web applications, address new threats and vulnerabilities on an ongoing basis and ensure these applications are protected against known attacks by either of the following methods:*
>
> - *Reviewing public-facing web applications via manual or automated application vulnerability security assessment tools or methods, at least annually and after any changes*
> - *Installing an automated technical solution that detects and prevents web-based attacks (for example, a web-application firewall) in front of public-facing web applications, to continually check all traffic*

Failing to comply with PCI DSS can lead to fines of up to $500,000 levied by banks and credit card institutions. If PCI DSS compliance applies to a company that does not have the resources to conduct an application vulnerability security assessment after every change, deploying a WAF could be necessary to comply with the standard. According to D'Hoinne and Hils, "The PCI requirement has given additional momentum to the WAF market, helping it expand beyond niche use cases, especially in financial and banking organizations" (2014, p. 3).

1.4.5. Intrusion Detection and Incident Response

WAFs can serve as sensors in a security monitoring infrastructure. Sensor collect data and alerts on potentially malicious traffic for analysis in the detection phase of intrusion detection and incident response, as shown in Figure 7. The alerts and logs from a WAF can be sent to a security incident and event management (SIEM) system for correlation with other sensors, enhancing monitoring capabilities and expanding it into web traffic. Over other kinds of sensors, a WAF gives the best visibility into web application traffic and potential attacks against an organization's websites. If an Internet presence represents a substantial risk to an organization, a WAF might be the most important security monitoring sensor the organization can deploy.

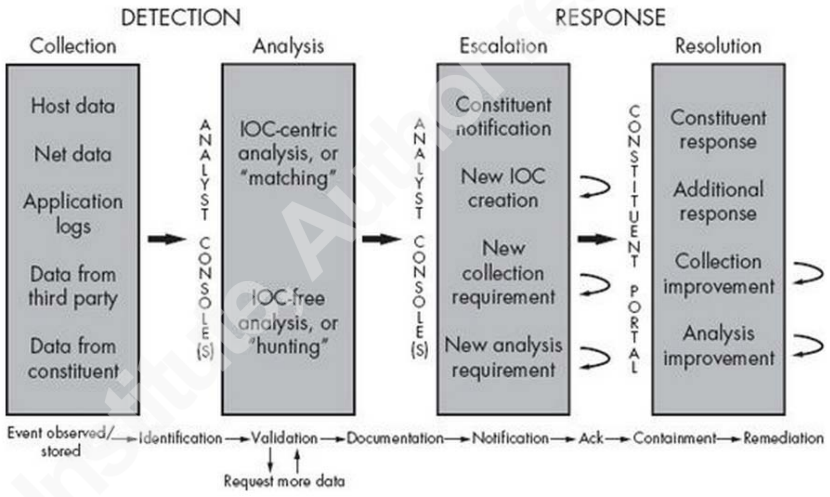

Figure 7. Intrusion Detection and Incident Response Process. From "The Practice of Network Security Monitoring" by Richard Bejtlich, 2013, No Starch Press.

1.5. Vendors

While outside the scope of this paper, choosing a WAF will rely on the environment and application requirements. Gartner has a Magic Quadrant for Web Application Firewalls that highlights the commercial space as shown in Figure 8.

Figure 8. WAF Magic Quadrant. From "Magic Quadrant for Web Application Firewalls" by Jeremy D'Hoinne, Adam Hils, Greg Young, & Joseph Feiman, 2014, Gartner.

2. Web Application Firewall Enterprise Techniques

A lab environment was put together to experiment with and demonstrate virtual patching and security monitoring using ModSecurity, an open source web application firewall.

2.1. Lab Environment

To experiment with and demonstrate web application firewall use cases, three virtual machines were built in a lab environment as shown in Figure 9. On a server running an Ubuntu Linux distribution, there is an Apache web server with a server resident instance of the open source WAF, ModSecurity. This server is running a web application called Damn Vulnerable Web App (DVWA). On a second server running a specialized distribution of Linux built for network security monitoring, Security Onion, a

ModSecurity centralized log management tool called AuditConsole is installed. On a third virtual machine running Windows is an environment for web application security testing using two tools: a DAST tool called Burp Suite and a vulnerability aggregation tool called ThreadFix. These components are further described below.

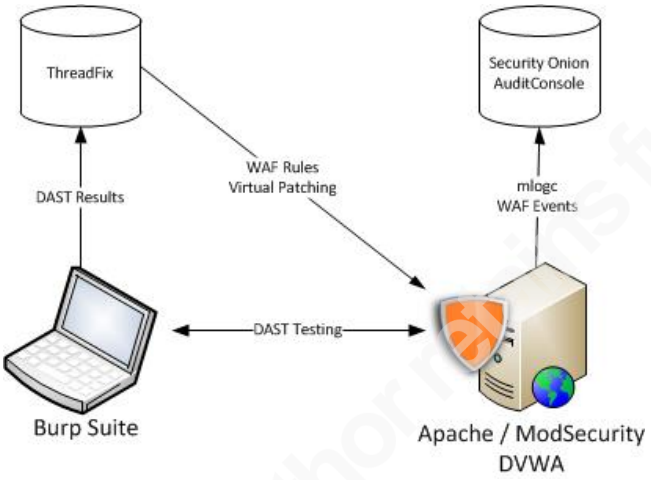

Figure 9. WAF Lab Architecture.

2.1.1. ModSecurity

ModSecurity is an open source, free web application firewall. According to Ristic, it "is a toolkit for real-time web application monitoring, logging, and access control" (2012, p. 4). Apache is an open-source HTTP server, and its functionality can be extended using modules. ModSecurity was implemented as an Apache module.

ModSecurity supports two deployment options: server resident and reverse proxy. Because ModSecurity is an Apache Module, it can be added to any compatible instance of a running Apache web server. For organizations that have many web servers, this scales as the web infrastructure grows. However, ModSecurity shares resources with the web server which could make an embedded implementation taxing on a web server already at full utilization. To setup a reverse proxy WAF, configure a dedicated Apache reverse proxy and add the ModSecurity module to it. In this configuration, there are more dedicated resources for the WAF. However, reverse proxy mode also adds a new point of failure to the overall infrastructure. For the lab, an embedded instance of ModSecurity was used.

2.1.2. ModSecurity Core Rule Set

The Open Web Application Security Project (OWASP) has a project that maintains a set of generic attack detection rules called the ModSecurity Core Rule Set (CRS). These rules aim to provide a base level of protection for all web applications. The CRS was installed with ModSecurity, and enabled to show how ModSecurity might function as a security monitoring sensor.

2.1.3. Damn Vulnerable Web App

The web application that ModSecurity is protecting in the lab is Damn Vulnerable Web App (DVWA). It is an open source application written in PHP with a MySQL database backend. DVWA has several security flaws purposefully written into a functional web application. It was created as a teaching aid for security professionals and software developers to learn how to discover and remediate vulnerabilities.

2.1.4. AuditConsole

With deployments of more than a single instance of ModSecurity, centralized logging is essential for scalable security monitoring. One application that can be used for central logging of ModSecurity events is JWall's AuditConsole. It receives events from multiple instances of ModSecurity, logs them to a database, indexes them, and has a web-application front end for security analyst review of the events.

AuditConsole can receive events via file-uploads of ModSecurity audit log files. It can also leverage mlogc, a tool that is distributed with ModSecurity, to send logs to the AuditConsole receiver using an HTTP connection. The events can either be stored in the embedded SQL database, or an external database such as MySQL. Users define rules that incoming events are evaluated against, such as creating notifications, deleting events, or executing scripts. Events can be tagged by users, allowing them to be noted as false-positives, or something to be followed up on at a later time.

Security Onion is a Linux distribution used for network security monitoring (NSM). AuditConsole was installed on top of Security Onion as a proof-of-concept. Once ModSecurity logs are centralized on Security Onion, one could utilize other tools built into Security Onion for analysis of those events or to correlate the events with logs from other NSM sensors.

Jason Pubal jpubal@mastersprogram.sans.edu

2.1.5. Burp Suite

Burp Suite is a platform for performing security testing of web applications. It is a favorite among application penetration testers. It has an interception proxy that allows inspection and modification of traffic between the browser and target application, and several built in tools for manipulating traffic for security testing. The professional version also has a dynamic application security testing scanner.

Burp Suite can be used to find vulnerabilities in a web application. A WAF, such as ModSecurity, can then be used to virtually patch those security flaws.

2.1.6. ThreadFix

ThreadFix is a web application vulnerability management tool that aggregates results from various web application security testing tools over time. It consolidates and de-duplicates findings, while providing various dashboards and reports of the data. It also integrates with various defect tracking tools to push web application vulnerability data into the format and processes that developers are comfortable using.

ThreadFix can also assist in virtual patching by generating WAF rules based on the vulnerabilities it stores. ThreadFix creates ModSecurity rules to block malicious traffic capable of exploiting vulnerabilities found in Burp Suite's DAST scan results.

2.2. Virtual Patching

In an ideal world, the secure software develop lifecycle (SDLC) would produce web applications free of security vulnerabilities. In the real-world, there is need for a web application vulnerability management process to assess applications, find security flaws, and help drive them to remediation. Once identified, vulnerabilities in custom applications take time to fix. Essentially, we are taking development time away from creating new functionality to devote to fixing a security issue. This is a costly endeavor at best. At worst, the web application is a legacy application and the company does not have development resources devoted to it. To fill this gap, organizations could use a WAF to help remediate web application vulnerabilities without the need to change the application's source code.

Virtual Patching is "a security policy enforcement layer that prevents the exploitation of a known vulnerability" (Barnett, 2013. p. 69). Given a custom rule that addresses a specific vulnerability, the WAF can analyze transactions and intercept attacks in transit so the malicious traffic targeting the vulnerability never reaches the application. Exploitation attempts would be unsuccessful without having to modify the source code of the application

The lab environment is running Damn Vulnerable Web App, which has several web application vulnerabilities. Using the reflected cross-site scripting example in DVWA, here is how virtual patching might work within a web application vulnerability management program.

2.2.1. Cross-Site Scripting

Cross-site scripting (XSS) attacks are a type of injection attack where an attacker sends a malicious script, typically browser side JavaScript, through a web application to another end user. The user's browser then executes that script, and the malicious code carries out its nefarious purpose. XSS can be used to "hijack user sessions, deface web sites, insert hostile content, redirect users, and hijack the user's browser using malware" (Williams and Wichers, 2013, p 9).

The XSS example in DVWA is implemented using a form asking for the user's name. After typing in "Jason" and clicking submit, the application says "Hello Jason" as shown in Figure 10.

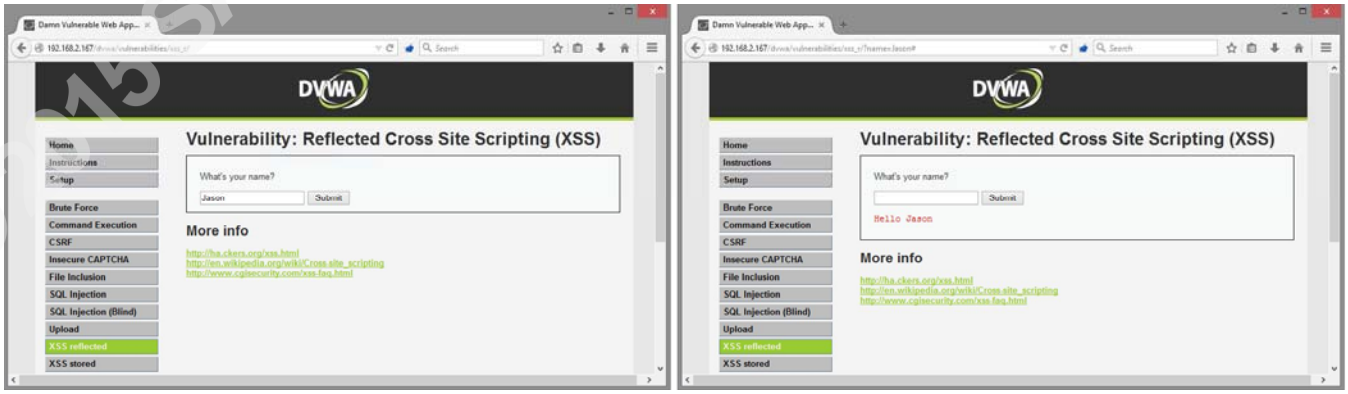

Figure 10. DVWA XSS Example Functionality.

Instead of typing in a name, one could provide JavaScript – for example, a simple script that creates a pop-up box that says "XSS":

<SCRIPT>alert<"XSS">;</SCRIPT>

DVWA will insert this script into the HTML where it would have put "Jason", and the browser will execute it, as shown in Figure 11.

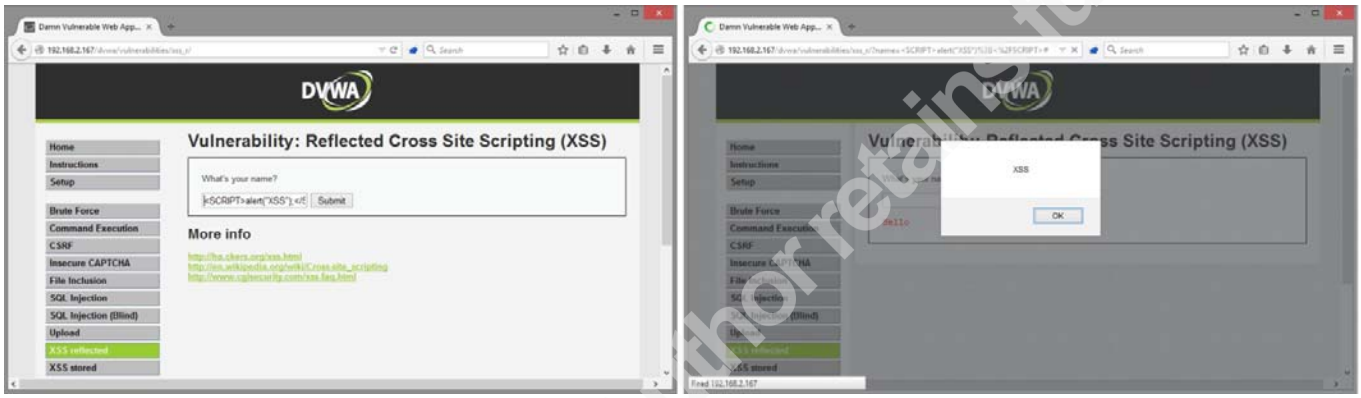

Figure 11. DVWA XSS Exploit.

2.2.2. Dynamic Application Security Testing

As part of security testing within the secure SDLC and as general vulnerability management, dynamic application security testing should be conducted against web applications. DAST technologies are designed to detect conditions indicative of a security vulnerability in an application in its running state. Web application vulnerability scanners are tools that scan web applications to look for security vulnerabilities such as cross-site scripting. They communicate with an application through the web front-end in order to identify potential security vulnerabilities and architectural weaknesses. First they spider the application, starting at the application's front page and recursively following every link to find each page and input. Then they fuzz those inputs looking for responses that indicate security issues.

In the lab environment, Burp Suite Pro was used to scan the XSS example in DVWA, and the XSS vulnerability was a finding in the results as shown in Figure 12.

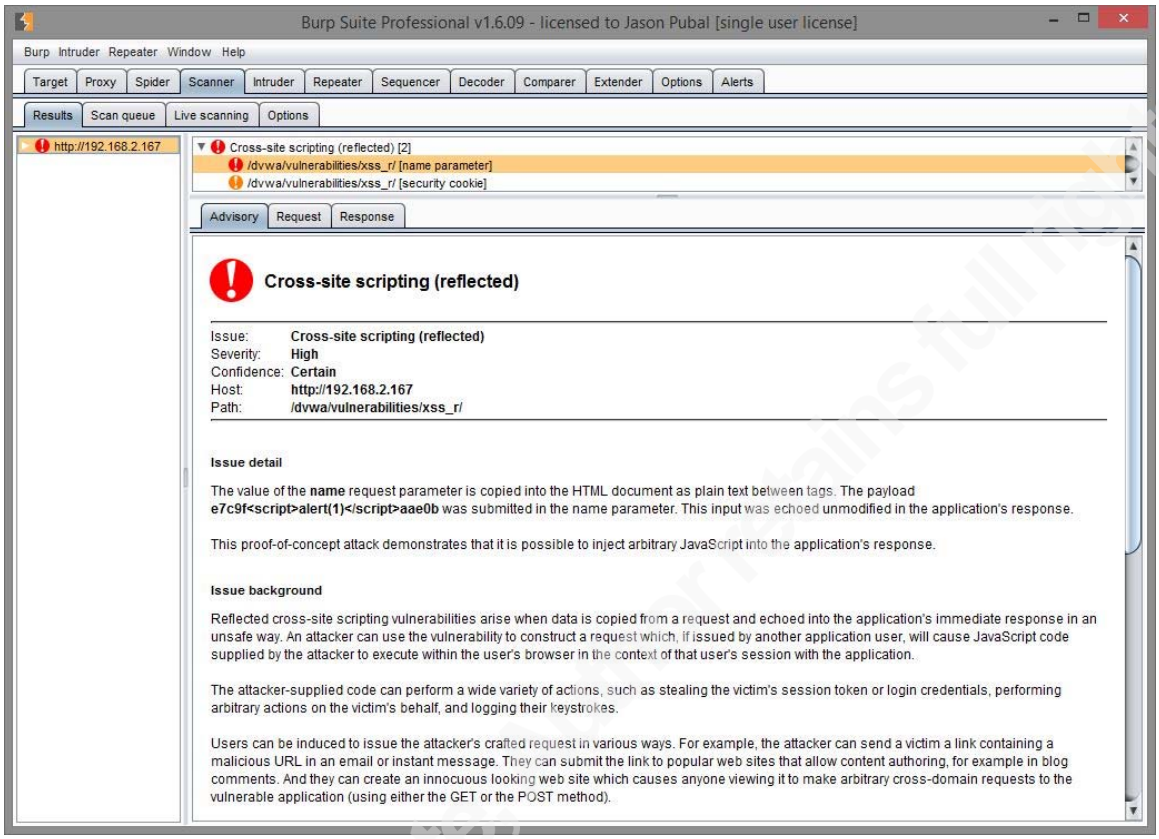

Figure 12. Burp Suite XSS Finding.

2.2.3. Vulnerability Aggregation

In an enterprise environment with several applications being developed or many applications being tested within a vulnerability management program, more than one application testing tool could be testing a plethora of web applications. Aggregation and correlation of all of those findings helps the management of a program with a large scope. In the lab environment, an open source vulnerability aggregation tool called ThreadFix was used to import the Burp Suite Pro DAST scan results as shown in Figure 13. ThreadFix was also used because it has the ability to generate web application firewall rules based on DAST results.

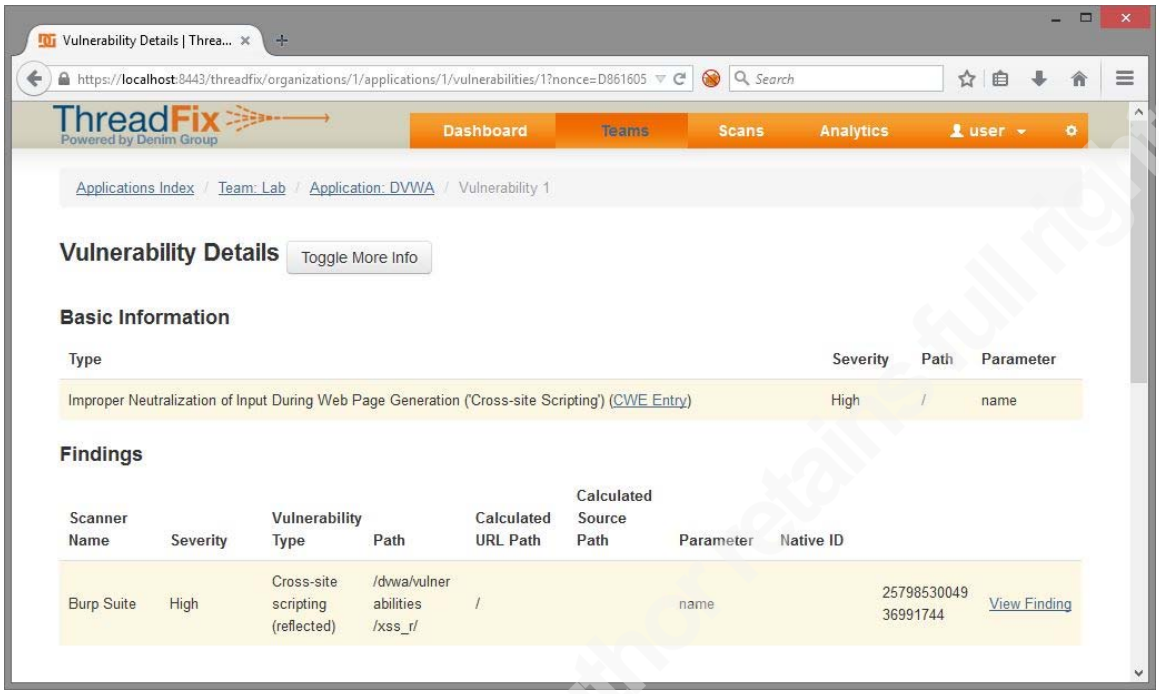

Figure 13. ThreadFix XSS Finding.

2.2.4. Web Application Firewall Rules

The last step is to have ThreadFix generate WAF rules to protect against the vulnerability, and deploy them to ModSecurity. ThreadFix's new WAF rule is shown in Figure 14.

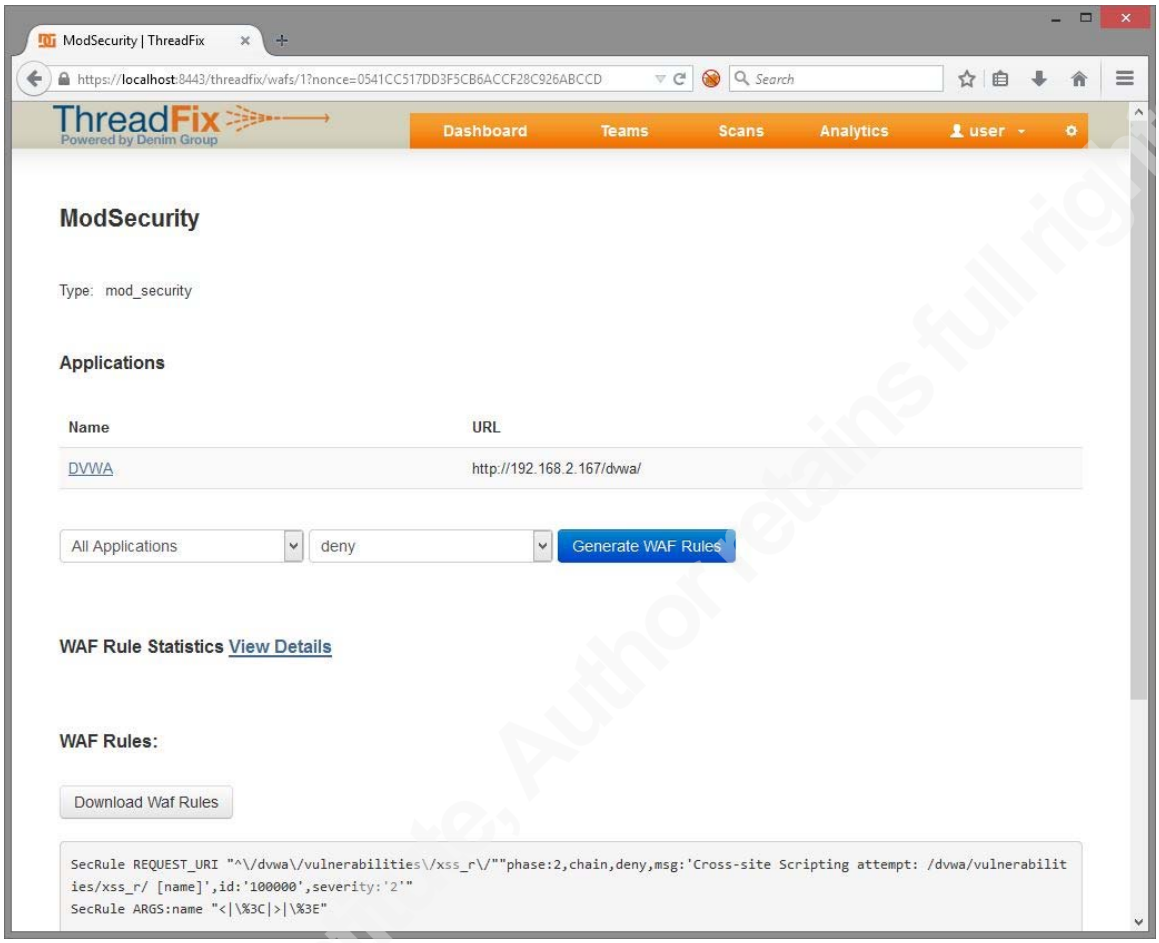

Figure 14. ThreadFix WAF Rules.

A file called modsecurity_crs_15_customrules.conf containing the rule was created, and a symlink was placed in ModSecurity's activated_rules directory. This was the only rule enabled in ModSecurity, as it was the only one in the activated_rules directory as shown in Figure 15.

Figure 15. ModSecurity Activated Rules.

After the XSS rule was deployed to ModSecurity, a test was performed to see if was effective. The same proof-of-concept JavaScript exploit above in our example was attempted a second time. This time, ModSecurity blocks the attempt with an HTTP 403 Forbidden error as shown in Figure 16.

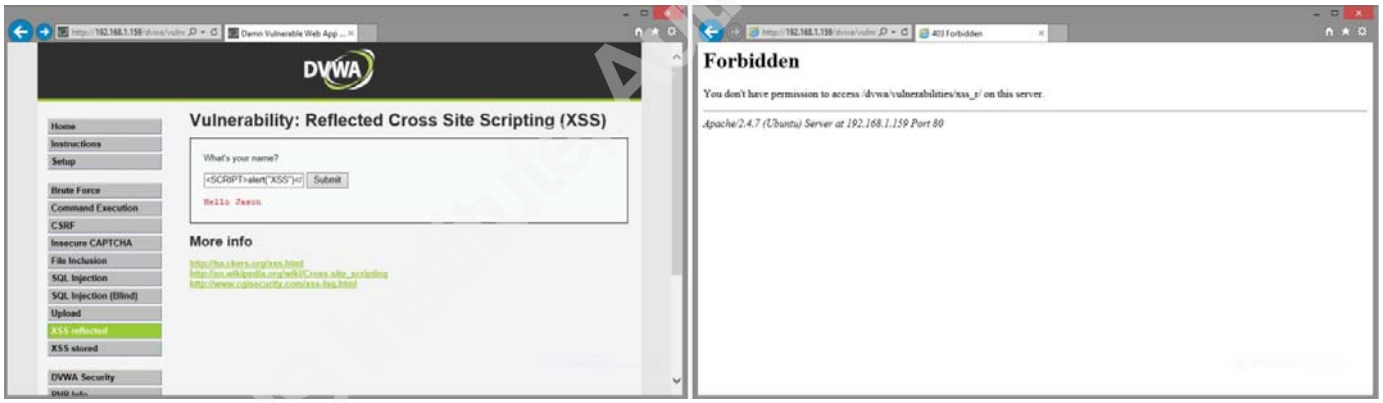

Figure 16. DVWA Blocked XSS Exploit.

Viewing the Apache logs shows that ModSecurity blocked the exploit attempt because it matched the pattern created in the new WAF rule as shown in Figure 17.

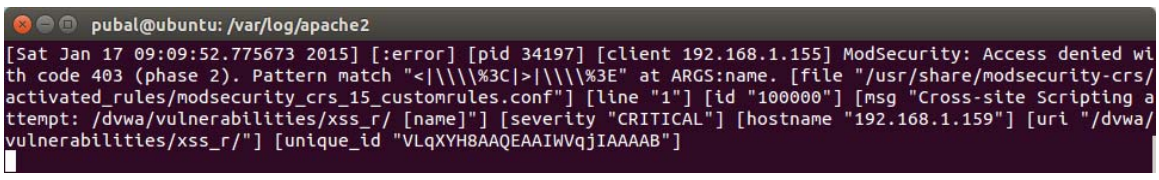

Figure 17. Apache ModSecuirty Logs.

Jason Pubal jpubal@mastersprogram.sans.edu

The main advantage of virtual patching is the speed of risk reduction. That is, the exposed attack surface is reduced far faster than by any other means. It is important to understand the level of protection virtual patching provides. Depending on the flaw, it may or may not be possible to completely remediate the vulnerability with a virtual patch. In some cases, the best that can be done is identifying when a malicious attacker attempts to exploit the vulnerability. Virtual patching and fixing the source code are not mutually exclusive, and virtual patching should only be viewed as temporary risk reduction until a source code fix is pushed to production.

2.3. Network Security Monitoring

Network security monitoring (NSM) is "the collection, analysis, and escalation of indications and warnings to detect and respond to intrusions" (Bejtlich, 2013, p. 3). NSM is a security paradigm that believes prevention eventually fails and security breaches are inevitable. The goal is to have enough visibility into the network to detect and appropriately react to a security incident. Crimes involving technology take time – after an initial compromise it can take days or weeks to find, gather, and exfiltrate what the criminal wants. Most sophisticated intruders seek to gain persistence in target networks. This gives defenders an opportunity to detect and respond to intruders before they can do damage to the enterprise.

According to Sanders and Smith, "the NSM cycle consists of three distinct phases: Collection, Detection, and Analysis" (2014, p. 9).

- **Collection**: The NSM cycle begins with collection. This occurs with sensors that are used to generate and collect data for NSM detection and analysis. Where the sensors are located defines what network visibility exists. Sanders and Smith recommend a risk based approach for deciding what data to collect, and where to deploy sensors. One such sensor could be a web application firewall. While quantifying risk to an organization, a mission critical, Internet facing web application quickly raises to the top making it obvious that web application monitoring is in scope. The alerting and logs that a web application firewall generates provide the best visibility into web traffic and web application attacks available.

- **Detection**: "Detection is the process by which collected data is examined and alerts are generated based upon observed events and data that are unexpected" (Sanders and Smith, 2014, p. 10). Detection can be as simple as a user calling the helpdesk to complain about an infected desktop. Usually, it is in the form of traffic that matches a signature on a sensor such as an intrusion detection system or web application firewall that then triggers an alert for investigation.
- **Analysis**: The NSM cycle ends with analysis. This is when a human reviews and interprets the data generated by the alert. During the course of the investigation, an analyst will often pull in data from other sources. This may involve packet analysis and network or host forensics. At this point, an event may be escalated to an incident and an incident response (IR) process would be initiated. Or, the result could be that the investigation was triggered by a false-positive alert and the sensor is tuned so that it does not alert on the same anomalous traffic in the future.

2.3.1. Web Application Firewall's Role in NSM

Web application firewalls are sensors in a network security monitoring infrastructure. In the risk based sensor placement strategy, consider what risk web applications pose to an organization. If an organization does not have a web presence or only has a small, static marketing website; web applications would rank low on a risk assessment. If an organization has a mission critical web application such as a large ecommerce site or is a business built on a mobile application, then a risk assessment would rank the risks involved rather highly. If web applications pose a high level of risk for an organization, a WAF will be the most critical sensor in an NSM deployment.

WAFs are sensors particularly suited for monitoring web application traffic. The ability to decrypt web traffic using SSL or TLS makes a big difference in traffic visibility as compared to other kinds of sensors such as firewalls or intrusion prevention systems (IPS). Gartner conducted a survey to see how organizations are handling encrypted traffic. According to D'Hoinne and Hils, the survey showed that "less than 20% of organizations with a firewall, IPS, or unified threat management (UTM) appliance

decrypt inbound or outbound SSL traffic. However, more than 90% of organizations with a public website and a WAF decrypt inbound web traffic" (2013, p. 2). That is, inbound web traffic that might pass an IPS uninspected because it is encrypted is much more likely to get decrypted and inspected by a WAF.

2.3.2. ModSecurity Centralized Logging via AuditConsole

As a proof-of-concept, the lab has an instance of Security Onion, an open source NSM Linux distribution. AuditConsole, a centralized log management tool for ModSecurity, is installed on top of Security Onion. ModSecurity is an NSM sensor collecting and sending logs and alert data to AuditConsole to enable detection and analysis by a security analyst.

During the DAST scan from the virtual patching example, the OWASP Core Rule Set was enabled in ModSecurity and configured to alert on malicious traffic. These alerts were sent, via mlogc, to AuditConsole. Figure 18 shows a screenshot of the dashboard in AuditConsole right after the scan completed. If this were a scan done for reconnaissance by an attacker, the security analyst could perform analysis based on these alerts and take whatever actions were deemed appropriate.

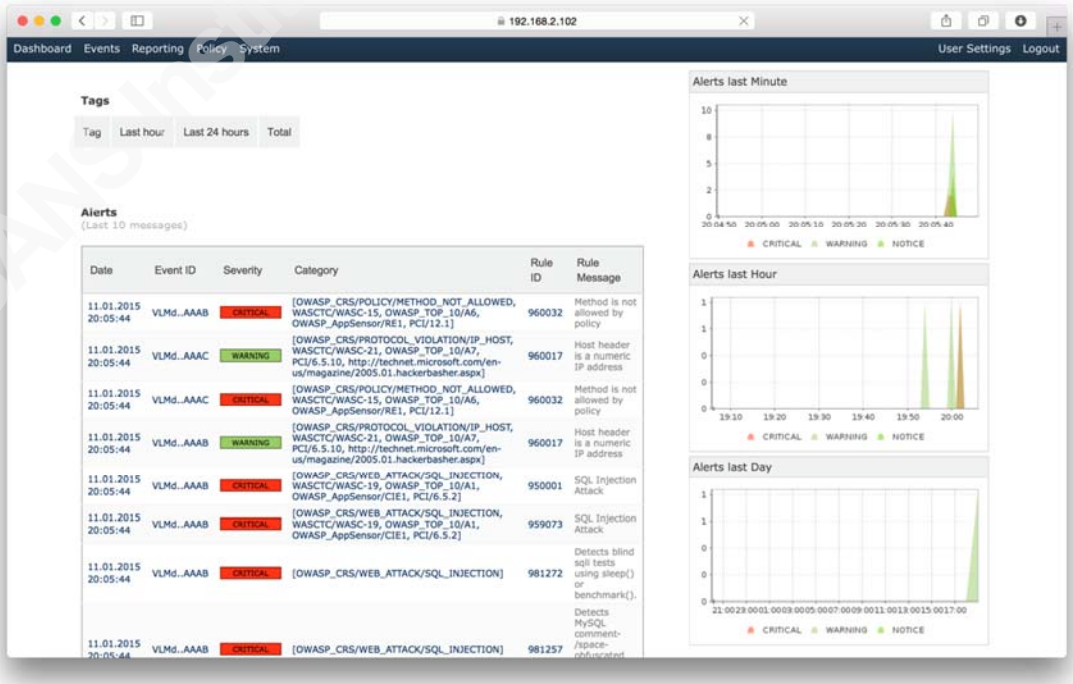

Figure 18. AuditConsole Dashboard.

Jason Pubal jpubal@mastersprogram.sans.edu

3. Conclusion

In the era of the Internet, web applications are critical to conducting business. Attackers know this, and are focused on exploiting vulnerabilities for their own gain. Web application firewalls shield web applications from these attacks.

When new web application vulnerabilities are found, virtual patching quickly remediates the risk without needing to change the application's source code. For production applications that will take time to fix, utilizing a web application firewall to deploy a virtual patch will require the least amount of effort for the desired security posture. For a legacy or COTS application, it could be the only mitigating control available short of taking the vulnerable functionality offline.

Web application firewalls have visibility into application traffic that no other security monitoring sensor is capable of analyzing. If web applications pose risk to an organization, web application firewalls are vital to your overall protection strategy.

As a final word of advice, consider who will be doing the application security monitoring. Application security has a different knowledgebase and skillset than general network security monitoring. Going beyond IP addresses and ports to look deep into Layer 7 HTTP traffic takes specialized skills. When adding application security monitoring to an analyst's responsibilities, consider giving that staff member the specialized training they need to be successful.

Jason Pubal jpubal@mastersprogram.sans.edu

4. References

Auger, R., Barnett, R., Cano, C., Chuvakin, A., Estrade, M., Ristic, I., et al. (2006, January).*Web Application Firewall Evaluation Criteria.* Retrieved November 20, 2014 from http://projects.webappsec.org/f/wasc-wafec-v1.0.pdf

Barnett, R. (2013). *The web application Defender's Cookbook: Batting Hackers and Protecting Users*. Indianapolis, Ind: John Wiley & Sons.

Bejtlich, R. (2013). *The practice of network security monitoring: Understanding incident detection and response*. San Francisco: No Starch Press.

Dermann, M., Dziadzka, M.,Hemkemeier, B., Hoffmann, A., Meisel, A.Matthias Rohr, M., et al. (2008, March). *Best Practices: Use of web application firewalls.* Retrieved November 20, 2014, from https://www.owasp.org/images/b/b0/Best_Practices_WAF_v105.en.pdf

D'Hoinne, J., Hils, A., Young, G., Feiman, J. (2014, June). *Magic quadrant for web application firewalls*. Retrieved November 25, 2014, from http://www.gartner.com/reprints/imperva?id=1-1VJQEFW&ct=140617&st=sg&elq=2df8cc4b7b814062bdda2fafd5c75ee3&elqCampaignId=

D'Hoinne, J., Hils, A. (2013, December). *Security leaders must address threats from risking SSL traffic*. Retrieved November 25, 2014 from https://www.gartner.com/doc/2635018/security-leaders-address-threats-rising

D'Hoinne, J., Hils, A. (2014, February). *Web application firewalls are worth the investment for enterprises*. Retrieved November 25, 2014 from https://www.imperva.com/lg/lgw.asp?pid=505

Jason Pubal jpubal@mastersprogram.sans.edu

Feiman, J. (2013, February). *Application security detection and protection must interact and share knowledge.* Retrieved November, 24 from https://www.gartner.com/doc/2350318/application-security-detection-protection-interact

Murphey, A., Salchow, K. (2007, October). *Applied Application Security – Positive & Negative Efficiency*. Retrieved November 20, 2014 from https://www.f5.com/pdf/white-papers/applied-app-security-wp.pdf

Nicolett, M. (2005, March). *How to develop an effective vulnerability management process*. Retrieved November 23, 2014, from http://www85.homepage.villanova.edu/timothy.ay/DIT2160/IdMgt/how_to_develop_.pdf

Payment Card Industry Security Standards Council. (2013, November). *Payment Card Industry (PCI) Data Security Standard.* Retrieved November 20, 2014, from https://www.pcisecuritystandards.org/documents/PCI_DSS_v3.pdf

Ristic, I. (2012). *Modsecurity handbook*. London: Feisty Duck.

Sanders, C., Smith, J., & Bianco, D. J. (2014). Applied network security monitoring: Collection, detection, and analysis. Waltham, MA: Syngress.

Verizon. (2014) *2014 Data breach investigations report*. Retrieved November 20, 2014, from http://www.verizonenterprise.com/DBIR/2014/reports/rp_Verizon-DBIR-2014_en_xg.pdf

Williams, J., & Wichers, D. (2013). *OWASP Top 10*. Retrieved January 20, 2015, from http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf

Jason Pubal jpubal@mastersprogram.sans.edu