SANS Institute
Information Security Reading Room

Reducing Organizational Risk Through Virtual Patching

Joseph Faust

Reducing Organizational Risk Through Virtual Patching

GIAC (GSEC) Gold Certification

Author: Joseph Faust, CISSP, GSEC, josephfaust.dc@gmail.com
Adviser: Rick Wanner

Accepted: June 19th 2010

Today's information cycles continue to evolve faster, with application vulnerabilities and discovery cycles continuing to increase in frequency, while IT resources and budgets do not scale accordingly with this trend. When considering how organizations approach patching their software applications, traditional patch management struggles to keep pace with identified vulnerabilities due to a variety of complexities that lie within the application, organizational culture and management itself. The sheer volume of new attack techniques to defend against in application software can be staggering, with new vulnerabilities publicly distributed each day and new sophisticated attack vectors published every week. By incorporating a "Virtual Patching" strategy, an organization can greatly improve efforts to reduce their organizational risk through quick remediation of vulnerabilities in web software. As the web interface has become the ubiquitous interface to software, this paper will provide an overview of virtual patching web applications with a focus on the open source project ModSecurity. Through the adoption of Virtual Patching as another tool in the information security arsenal, this article will illustrate how organizations can decrease the risk from software vulnerabilities and provide overall better defenses across their technology environments.

1. Introduction

Software patching for IT Departments across the organizational landscape has always been an integral part of maintaining functional, usable and stable software. Historically the traditional patch cycle has been focused on fixing or resolving issues which affect functionality. In recent years, with the advancement of more sophisticated and targeted threats which are occurring in quicker cycles, this focus is dramatically changing. (Risk Assessment – Cisco, n.d.; Executive Office of The United States, 2005) . Corporations and Government now have a greater understanding of potential losses and expenses incurred by not maintaining application security and are moving towards an increased focus on patching and security (Epstein, Grow & Tschang, 2008). With organizations' reputations, consumer confidence and corporate secrets at risk, corporations and government are recognizing the need to shift and address vulnerabilities at a much faster pace than they historically have done so (Chan, 2004). Over roughly the last ten years, the length of time between the documentation of a given vulnerability in a piece of software and the development of an actual exploit that can take advantage of the weakness in the application, has decreased tremendously. According to Andrew Jaquith, senior analyst at Yankee Group, the average time between vulnerability discovery and the release of exploit code is less than one week. (Business Wire, 2006). It has also been identified that "99% of intrusions result from exploitation of known vulnerabilities or configuration errors where countermeasures were available" (Network Integration Services, 2010) . Clearly these statistics alone can prove daunting for many businesses trying to keep pace and maintain proper defenses against the bad guys.

For many businesses and organizations, the goal to maintain high quality, secure software exists, but realities impede the progress in being able to quickly address a vulnerability before it can be taken advantage of in a malicious manner. Several factors

Joseph Faust, josephfaust.dc@gmail.com

can complicate the ability to provide a rapid response time in producing a patch, or to mitigate a given vulnerability once it has been identified. First is application expertise. The lifespan of given software systems traditionally tends to outlive the anticipated duration which it was originally slated for. While application expertise was around during the software project's development, in many instances the talent is long gone and the application lives on without an active expert in the given technology or software. With a potential critical and damaging vulnerability around the corner, the lead time between when an issue is identified and when a patch is in place can be severely delayed due to the absence of such expertise. Finding appropriate talent who is versed in the proper programming language, understands the application and its dependencies and is able to mitigate software vulnerabilities with a fix, can prove to be a time consuming task.

Second are legacy applications, or third party vendor applications, where a business's commitment to significantly fund and maintain them is no longer an option, or the source code remains intellectually property of a third party vendor. Patch response time will vary greatly given the circumstances, but in many cases the business which relies on the software may not be able to engage the vendor for a fix. This may be due to the high costs to implement a fix, lack of knowledgeable security expertise staff with the vendor, or the complications derived from political and legal relationships with the vendor, such as SLA agreements, etc.

A third factor which can complicate the process of putting together an appropriate patch to secure an application against a recently identified vulnerability is the tendency to for an organization to fear change in a given system. Many times an application will reach a level of stability, a point where main stream bugs have been discovered and patched, but yet still many bugs and software flaws exist and are yet to be discovered and exploited. This is true in particularly in the case of legacy software which remains

functional for the needs of the business, but to which little attention has been paid because the software "still just works". Making a change to the software in a complex system can introduce risk, as maintaining the general availability and up time of online systems is crucial for corporations and businesses, and a change could impact the availability of the software application and translate into monetary loses or availability outages.

The decision to make changes to long standing legacy code can be analogous to decisions or thought process a doctor might go through when weighing the benefits of options for patients. "Why perform open heart surgery on a patient who is 80 years old and in otherwise good health for a potential heart problem that may or may not prove fatal?" In software, the idea that we might create more problems, introduce instability or possibly introduce new bugs during the operation is a real consideration. Many times given the overwhelming complexities or cost/benefit feasibility to develop a patch, a do-nothing assumption of the risk approach is taken, in light of no other reasonable options. "Wouldn't it be great if we could attempt to fix the patient and lower the risk of heart failure, without opening him up with surgery?" These factors which can significantly delay or impede a patch to be produced in a timely manner can be greatly reduced through the use of a Virtual Patch.

2. Closing The Gap with Virtual Patching

2.1 A Virtual Patch Defined

In situations where traditional patches are not feasible, a virtual patch can be utilized to reduce the likelihood of a successful attack. A virtual patch can be defined as the ability to do "vulnerability mitigation in a separate layer, where you get to fix

Joseph Faust, josephfaust.dc@gmail.com

problems in applications without having to touch the applications themselves" (Ristic, 2010), or as "a policy for an intermediary device (WAF) that is able to identify/block attempts to exploit a specific Website vulnerability (Barnett, 2009). A virtual patch deals with the process or method of fixing problems by altering or eliminating vulnerabilities by controlling the inputs and outputs to and from the applications (Shinn, 2008).

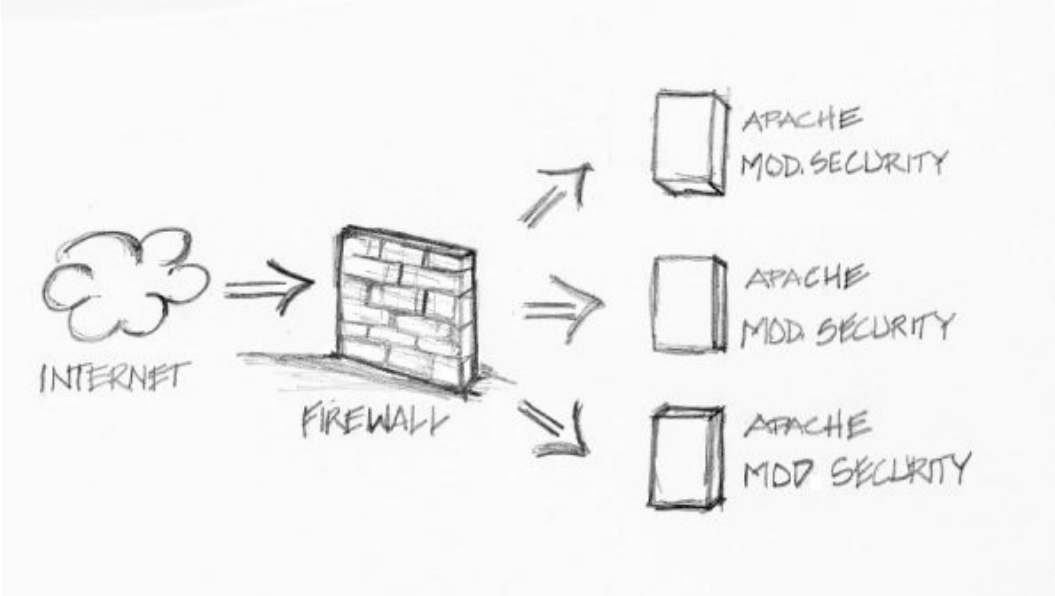
A virtual patch can be created to mitigate a given vulnerability through packet manipulation & proxies via brokering the protocols to the application in question. The virtual patch itself is created through the rule language of the packet manipulator. For the purposes of this paper, I will focus on Virtual Patching with Mod Security, a popular and extremely versatile Open Source Web Application Firewall (WAF) maintained by Breach Security used to create and apply custom virtual patches. The range and capabilities in Mod Security are far reaching and in this paper we will focus specifically on Virtual Patching approaches with Mod Security.

2.2 Architecture

The architecture of the environment for which packet manipulation can occur is key and a cornerstone to the virtual patch success. Using Mod Security, several network arrangements exists to allow packet manipulation for a virtual patch to be implemented.

2.2.1 Embedded

The first arrangement is to run Mod Security embedded as an Apache Module on the server which you aim to defend and apply virtual patches to a given application. In this case Mod Security runs embedded with the applications which it is defending, as an Apache Module. This is the simpler of the two methods, as no other changes are needed to enable defense and virtual patch capabilities for your web applications. See Figure 1-1

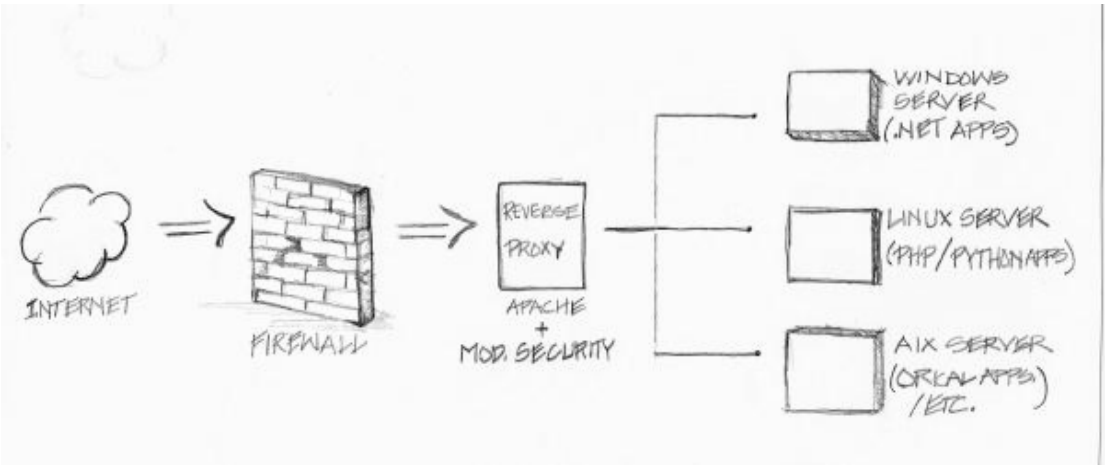


(Figure 1-A) Standalone Embedded ModSecurity Environment

2.2.2 Reverse Proxy

The second architecture to run Mod Security to enable defense and virtual patching is to configure Mod Security on a reverse proxy server running Apache. In this case Mod Security can defend and apply virtual patches to numerous applications, on numerous servers which it is tasked to oversee.

Joseph Faust, josephfaust.dc@gmail.com



(Figure 1-B) Reverse Proxy with ModSecurity

With this model many advantages exist as a security separation layer exists and the defense is truly operating system and programming language agnostic. Mod Security can be configured to protect various web applications running on a various operating systems, whether they are Linux, Windows, Unix, MacOS etc. – it matters not.

An added benefit to this model is that administration efforts can occur with greater ease, as you're dealing with one technology stack to administer instead of splitting administration expertise across several O/S environments. A key and *fundamental* assumption to both architecture models is that your applications and Mod Security are tightly coupled. Mod Security is tasked to maintain the integrity of the application, and proper application request routing must be maintained with this defense approach.

2.2.3 Architecture Advantages & Disadvantages

Deciding which is the right and the best architecture approach for your organization will truly depend on your business needs. Running Mod Security as an

Apache module on each individual server can be quickly stood up and ready to be brought online without requiring a change to the network , thus reducing the number of overall team members involved to implement. Also with this architecture, no new hardware is really needed to implement an embedded ModSecurity install. One of the downsides to running ModSecurity embedded is that while the Mod Security application footprint is minimal, it will still share the computing resources of the server hosting the web application. Another drawback to the embedded architecture is the potential administration troubles, that can quickly get out of hand in managing multiple servers with rule sets and configuration, as more web servers and web applications are introduced to the company's application lineup. This can be aided through scripted deployment and auto updating, but it does involve a bit more complexity to maintain a growing number of servers which may increase exponentially as the organization grows over the years.

Running Mod Security on a reverse proxy server does have several business advantages as the expertise to maintain and update one instance of Mod Security with virtual patches and configuration can be leveraged against many web application and servers which need to be defended and virtually patched. The organization can respond to taking other vulnerable applications “under the wing” fairly quickly, which may not have originally been identified as candidates needing crucial patches. The flexibility of this architecture approach shines since the applications that might need critical patches can quickly be arranged to be covered by Mod Security for protection and virtual patching, even though the applications may be written in a variety of languages and running on different operating systems. This architecture also works well for closed source proprietary web based vendor software, which your organization may run, but which the vendor may not still be in business or adequately capable of patching the software.

A few drawbacks to the reverse proxy server implementation include introducing

a single point of failure if the reverse proxy fails or bottlenecks traffic. With the advances in virtualization and on demand resource balancing this may become less of an issue and drawback to this approach. With each architecture there are advantages and disadvantages. Ultimately, the decision boils down to whether your organization chooses to implement centralized (reverse proxy) virtual patching administration or a decentralized (embedded) virtual patching administration approach.

2.3 The Patch

2.3.1 The Vulnerable Web Application

With either architecture in place, the environment is set to deploy a virtual patch in response to a reported vulnerability. To get an understanding of how a Virtual Patch can be applied to mitigate a real application vulnerability, we'll turn to a case where the code handling user logins of a company commerce portal is flawed, leaving the website susceptible to a SQL injection attack. This vulnerability could be used by an attacker to gain unauthorized access to information or escalate his privileges in the system, gaining administrator access. Once the attacker has gained administrative privileges, the potential end result of the successful exploitation of this web application vulnerability could translate into significant dollar losses for the business as information stored in the application is up for grabs for those with mal intent. With this security compromise, the potential damage to the company's reputation and the loss of consumer confidence is great, as customer or financial data could be stolen and sold on the black market or company trade secrets lost.

2.3.2 The Injection Attack

The attack occurs by exploiting improper validation of the login input parameters by causing a change in the original application design's authentication logic. Instead of executing the original programmer's logic, the attacker is able to prematurely end the

logic of the program and create his own, effectively bypassing any login security to gain complete access to all information.

The attack string below shows the attacker's initial reconnaissance SQL injection probe, which provides information back to the attacker that the application is vulnerable to injection and data alteration. This exposure to sql injection can lead to the destruction, modification or theft of the organizations' application data.

The Attack Request:

GET /app.asp?Login='%20or %201=convert(int,(select%20@@version %2b'/'%2b@servername2b'/'%2bdb_name() %2b'/'%2bsystem_user))--sp_password

Application Response Returned to the Attacker:

HTTP/1.1 500 Internal Server Error
Content-Length: 598
Content – Type: Text/ html
Cache-control: private
Set Cookie:ASPSESSIONIDGQQGQGCS=JHMBOBKCBINEHLPKJHOPABBE; path=/
Connection: close
<p>Microsoft OLE DB Provider for ODBC Drivers
Error '80040e07'
[Microsoft] [ODBC SQL Server Driver] [SQL Server] Syntax error converting the nvarchar value 'Microsoft SQL Server 2000 - 8.00.2039 (Intel X86) .May 3 2005 Copyright © 1988-2003 Microsoft Corporation .Standard Edition on Windows NT 5.2 (Build 3790: Service Pack 1) /EXAMPLE_SQL/OPT/OPT2 ' to a column of data type int.

Joseph Faust, josephfaust.dc@gmail.com

Figure 2-A Adapted from "ACM PD Lecture: Intrusion Detection" (Shinn, 2007)

2.3.3 The Virtual Patch Defense

Upon security review or notification of the vulnerability, an information security practitioner utilizing Mod Security or a network appliance that supports virtual patching, can craft a virtual patch used to re-mediate the issue and have the vulnerability avenue shutdown, potentially within minutes.

The Defending Virtual Patch :

```
SecRule ARGS:Login "!^([a-zA-Z]
{8,16})$“ \
"phase:2,capture,log,deny,status:404
```

Figure 2-B Adapted from "ACM PD Lecture: Intrusion Detection" (Shinn, 2007)

In this case, this is accomplished through crafting a virtual patch in Mod Security to mitigate the vulnerability and quickly shutdown the avenue for the SQL injection in being able to successfully bypass authentication.

2.3.4 Mod Security Rule Engine

Virtual Patching in Mod Security is accomplished through a rule-based engine,

which inspects traffic at the protocol level and acts accordingly to the given rule set. Specifically crafted application defense and mitigation rules are loaded into the Mod Security software to respond to situations when web application attack conditions are met, with Mod Security having the ability to perform deep inspection of the http protocol and to take defensive action accordingly. Rules are honored in Mod Security based on the rule set regiment which follows:

SecRule VARIABLES OPERATOR [TRANSFORMATION_FUNCTIONS, ACTIONS]

In the preceding example, the ModSecurity engine inspects all http protocol traffic destined to the vulnerable application and enforces a custom application rule set for compliance, in order to even be able to communicate with the login page. In this case, Mod Security inspects the Login parameter for compliance to the crafted patch, allowing only valid character requests and length, and denying user activity that does not conform to the virtual patch defense rules – therefore preventing successful exploitation of the weakness in the login page. Upon failing the protocol inspection rules, the attacker's activity is captured with a volume of stateless http details (which can later be reviewed to forensically understand the attack), and the attacker's request to communicate with the website is denied.

2.3.5 Vulnerable Web Application – Brute Force Attacks

In another example, a company's web application's login page is susceptible to brute force log in attacks, where an attacker has time and computational automation on his side to successfully achieve a brute force attack, in effect trying all possible password key combinations until success. A Virtual Patch in Mod Security once again is able to be called into action for defense to mitigate the threat of a brute force attack— all the while,

not modifying an ounce of source code in the vulnerable web application.

2.3.7 Virtual Patch – Brute Force Mitigation

The following figure outlines a potential defense rule that can be put in place in Mod Security to help limit the web application's exposure to unfettered, brute force cracking of a valid username and password combinations in the login.php page of the website.

A Virtual Patch To Prevent Brute Force Attacks:

```
<Location /login.php>
# Enforce an existing IP address block

SecRule IP:bf_block "@eq 1" "phase:2,block,\
msg:'IP address blocked because of suspected brute force attack'"
# Check for authentication failure

SecRule RESPONSE_HEADERS:Location ^/login.php \
"phase:5,chain,t:none,nolog,pass, \
msg:'Multiple authentication failures from IP address',\
setvar:IP.bf_counter=+1"
SecRule IP:bf_counter "@gt 25" "t:none,\
setvar:IP.bf_block,\
setvar:!IP.bf_counter,\
expirevar:IP.block=3600"
</Location>
```

(Ristic, 2010)

Joseph Faust, josephfaust.dc@gmail.com

In this rule, authentication is checked for login failures exceeding a predefined threshold (25), and a block of the attacker's IP address is put in place for one hour to prevent subsequent tries to login in, mitigating this application threat.

The Virtual Patch can also be expanded upon and made more granular to meet additional requirements. Here, the virtual patch can be implemented quickly and with little cost in comparison to a traditional patch, in the event that reworking the web application code is not immediately an option to hold defenses against this type of attack.

2.3.8 Vulnerable Web Application – Session Attacks

A third scenario where virtual patching proves useful is for a web application that may have been in use in a production environment for some time and is found to be susceptible to a session attack due to the way cookies are being set. While today it is a known and good coding practice to set cookie values with the HTTP-only flag, this application was not coded with that standard. Either that development guideline or browser capability did not exist at the time the application was developed, or the developer simply was not aware, or did not adhere to best practice. Nonetheless, a large potential amount of applications were put into production this way. Due to this coding approach the application is now exposed throughout with XSS vulnerabilities and the potential for an attacker to successfully steal a login session cookie is increased, in effect bypassing the designed login security of the web application. The vulnerable cookie exchange is shown below.

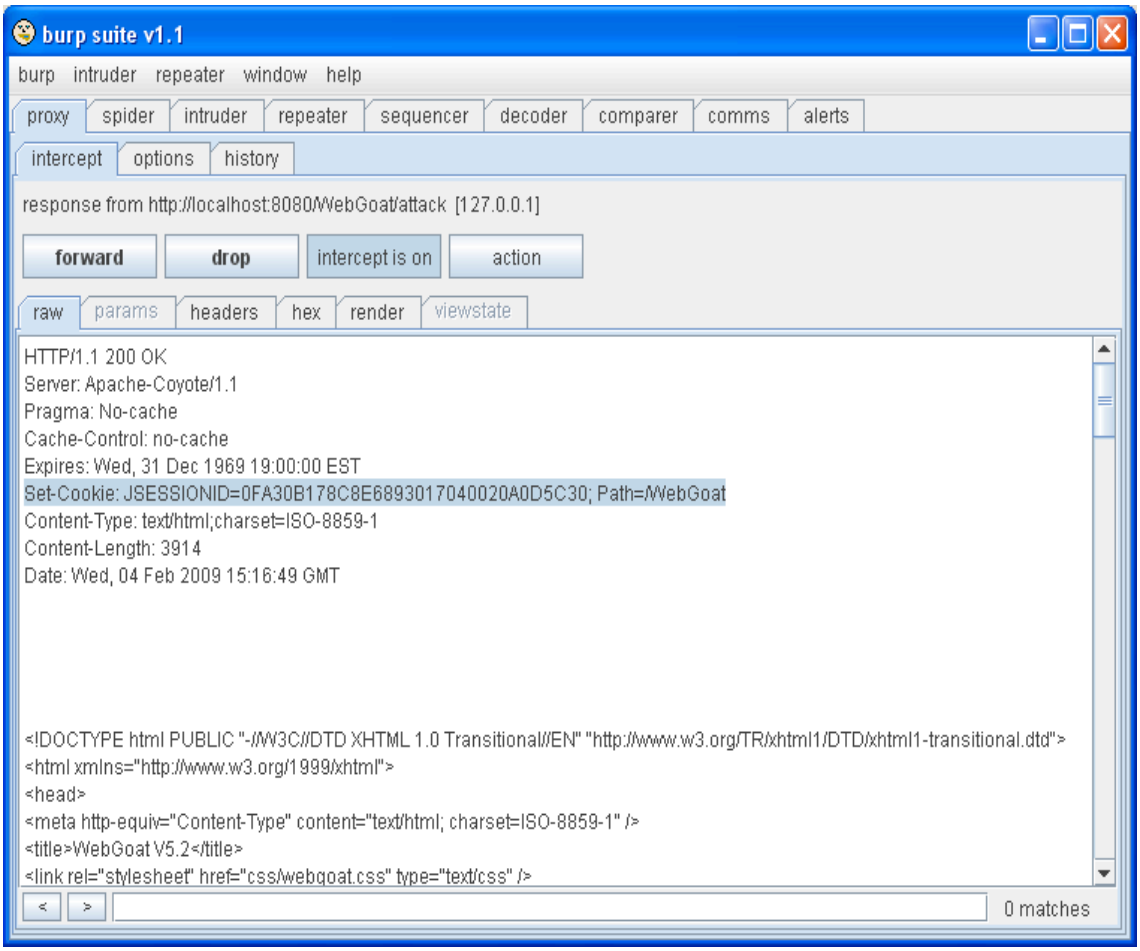


Figure 2-B Cookie Exchange Susceptible to XSS Attack

(Barnett, 2009)

The attackers can move quickly to exploit this vulnerability, and time is of the essence. While ideally, a skilled programmer in the application language is quickly able to review and rework the source code against this vulnerability, the reality of it happening in such a timely fashion due to organization constraints may prove otherwise. The significant risk faced that this vulnerability introduces for the business may not be acceptable and waiting for the delivery of a traditional patch could prove to be too late to be useful, compromising the application and data for an active and persistent attacker.

Joseph Faust, josephfaust.dc@gmail.com

Again, armed with a well crafted Virtual Patch, this vulnerability can instantly be mitigated across a single server or dozens of servers once the Virtual Patch is in place and correctly altering the response cookie with the proper HTTPOnly instruction.

2.3.9 Virtual Patch – Session Attack Mitigation

The Virtual Patch crafted in response to in this case is able to detect outbound cookies which are being set and sent back to the browser client without the correct HTTPOnly attribute, and to change them on the fly to the correct coding standard, mitigating the chance for a successful XSS attack.

```
SecRule RESPONSE_HEADERS:/Set-Cookie2?/ "!(?i: \;? ?httponly;?)"
"chain,phase: 3,t:none,pass,nolog"
SecRule MATCHED_VAR "(?i:(j?sessionid|(php)? sessid|(asp|jserv|jw)?session[-
_]?(id)?|cf(id| token)|sid))" "t:none,setenv:http_cookie=% {matched_var}“
Header set Set-Cookie "%{http_cookie}e;
HTTPOnly" env=http_cookie
```

(Barnett, 2009)

The rapid response time and scalability of this patch makes Virtual Patching an attractive tool which can hold the defenses in absence of the immediate availability of the personnel who can get a software patch in place, or to serve as stop gap to buy time until funding constraints for a software level application fix can be secured. General resource availability constraints may exist, and more time may be needed to properly implement the correct application fix with the appropriate teams. With a virtual patch in place and

defending against this threat, a traditional software patch could be created and systematically flowed through a formal build process to release to production, without compromises or shortcuts in maintaining the build quality standard.

Given the choices that many organizations face, Virtual Patching provides another strong option for Government Organizations, Corporations and Non-Profits to scale up defense tactics given the volume of vulnerabilities being found in web applications and the and limited resources to get them fixed. In the three preceding examples, a security practitioner can identify the flaw or vulnerability and at the same time offer a solution that can be put in place to reduce the risk for the organization. The Information Security practitioner need not have specific in-depth knowledge of the application language or framework (PHP, PYTHON, JAVA, .NET, etc) , rather they handle all patches in a separate layer. With this separation layer abstraction, a dynamic and active response to counter a given threat can quickly be achieved.

3. ModSecurity & Virtual Patching Approaches

3.1 Patching Methodology

While the intention of this paper is to focus on the aspects of the custom Virtual Patch, one would be remiss not to mention community rules which are available in ModSec and maintained by a vibrant community of information security professionals. These rule sets, ‘out of the box' , apply many community defense rules which are more general in protection nature, but inherently make your web applications significantly more secure. Rule-sets can be configured to auto update community rules to stay current for defenses which are generic in nature or for an administrator to download on demand, review and deploy accordingly. Each approach will greatly depend on an organization's tolerance for risk.

Joseph Faust, josephfaust.dc@gmail.com

3.2 Negative Security Model

Rules in Mod Security can be crafted to take a permissive approach using a negative security model where the all traffic is allowed and only the offending attack method is hunted and mitigated. Using the positive security model, the exact opposite is performed – allowing only traffic & behavior of that which is specifically allowed and defined, denying anything which does not meet the set rules for validity. Negative security model rules are generally easier to create, require volumes of rules to provide effective coverage and can be quicker to implement but easier to defeat.

3.3 Positive Security Model

The Positive Security model provides a stronger defense and requires more time to analyze the target application and craft the appropriate virtual patch, and more time to and vet accordingly. The potential to temporarily break an application can increase greatly if the scope of the patch is identified incorrectly.

4. Virtual Patching Benefits

4.1 Technical

Through Virtual Patching techniques, Information Technology Departments can better be positioned to protect the health and availability of the systems which they maintain and get out in front of potential issues, as they are revealed to the information security community. Application security patch response times can be greatly be reduced with a virtual patch and can pair well with the incident response process for shutting down active attacks. When activity is detected and determined to be severe, a response team can act accordingly and take actions that defeat the attack, by quickly crafting a virtual patch for defense. With Virtual Patches, emergency application fixes can be implemented by the security practitioner who discovered them or is on top of the

releases documenting the vulnerability. The security practitioner best understands the vulnerability, and can assemble a virtual patch very quickly, instead of waiting to get the right developers with the right skill set and language capabilities who are authorized to make the appropriate application software fix.

4.2 Business

On the business side of Virtual Patching, the benefits of virtual patching are prevalent as well. With Virtual Patching, critical corporate and government applications and customer data can better be secured, as the virtual patch can quickly shutdown the avenue to exploit an application for profit, intelligence gathering, etc. Overall, the attacker's window to an application is shortened and the risk for the business is reduced along with that window.

For high availability systems which require downtime for a traditional patch to be put in place, virtual patching again provides solid benefits in this area. To implement an urgent software patch, taking the system off-line can translate in loss of revenue or availability of the system at inopportune times. The chance that the traditional software patch introduces new instability or causes of an unforeseen issue always exists. While a traditional emergency software patch can be rolled back in the event it fails, many times this involves multiple teams in the process and the patch back out and regression testing can be costly. With a critical vulnerability discovered, a virtual patch can be crafted, tested and deployed without taking the application off-line. The virtual patch can easily rolled back without the need to do deep regression testing, as the original code base is left intact and only the virtual patch removed in the event of an issue. This allows for adequate time and consideration for code changes to be put in place, without the extreme urgency and risk of imminent of exploitation while waiting for the traditional software patch.

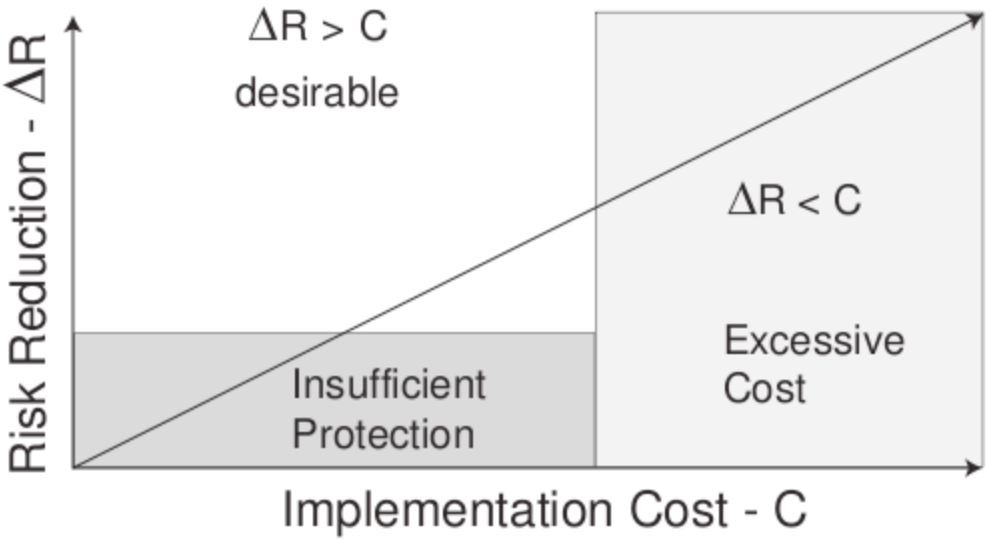
Joseph Faust, josephfaust.dc@gmail.com

4.3 Getting It Right & Striking A Patch Balance

Long gone are the days are the days where the effort for an attacker to find and exploit these vulnerabilities would take a considerable and sustained effort, as automation tools have eased this effort. Once a vulnerability is identified, an organization must balance their decisions on the approach to fix through traditional formal build software patches or through an on demand virtual patch. If there is a proof-of-concept exploit code publicly released, speed must be considered in mitigating the threat as soon as possible. Ideally, whenever a software patch can be created and applied to the direct system, it should be. Working through these considerations and application risk profiles, one can best determine the approach for sending specific patches for a traditional source patch approach and those which may need to first be covered with a Virtual Patch.

Joseph Faust, josephfaust.dc@gmail.com

The Virtual Patch can be applied first and later accompanied with a properly vetted traditional source code fix. With a Virtual Patch, patches can be created very quickly, with minimal cost by a skilled rule writer and rollback can occur within seconds or minutes, given the installation method of a virtual patch. This poses a lesser threat to breaking application functionality and allows for a far more aggressive iterative patch stance than that of a traditional software patching. With a virtual patch approach, time



can also be bought in having protection until a formal software level patch in the application is ready to go to production. When faced with the financial business question of to patch or not patch - and accept the risk, potential losses, virtual patching can help balance the equation when options are limited and the costs to traditionally patch prove too excessive.

The figure 2-B above shows optimal patching approaches with respect to business

Figure 2-B Threshold for Implementing a Countermeasure (2010, AKELA)

and organizational costs required to reduce risk. Virtual Patching in many cases can be light on implementation cost and heavy on gains in risk reduction.

5. Virtual Patching Strategy – Areas To Focus On

5.1 Assess & Prioritize

To be successful with your Virtual Patching approach, one should be prepared to spend adequate time in determining what priorities exist for your applications and what level of risk is acceptable. Scan and have your applications penetration tested for flaws, revisiting them on a continuous schedule, working through the identified vulnerability lists to determine real weaknesses in your applications that need to be patched.

5.2 Test & Tune Virtual Patches

When creating rules, consider whether the Negative or Positive Model best fits the application in question. Spend enough time testing each Virtual Patch against the applications which you are defending, so that false positives can be minimal. Plan for False Positives, where valid traffic behavior is being denied and have a documented process for those in your organization to report issues and escalation process to resolve (Shinn, 2007). Too many false positives and the goodwill and cooperation of your internal business customers or clients will begin to fade. When deploying virtual patches in mass or with generic community rules for the first time, it is important to plan for a necessary period of tuning.

Joseph Faust, josephfaust.dc@gmail.com

In situations where False Positives can be severely detrimental, run your initial rules with actions to only log and not block the request. This way you can review the logs of the application interactions that would have been blocked, to determine if it was a valid behavior of the application which your rule did not account for. False Negatives, where attacks that should have blocked but are let through need to be considered as well, as "absence of evidence is not evidence of absence" (Dr. Carl Sagan).

5.3 Communicate

To be successful with a Virtual Patching strategy, it is also important to ensure communication with the application owner or organization about the ongoing defenses is being maintained. Periodically provide metrics to management via analyzing the logs for statistics, trends and explain and communicate these active threats and challenges which the organization faces. Make it part of a regular interval to report on malicious attack activity faced and the provided defenses implemented through virtual patching. The need to continuously adapt to today's increasingly sophisticated attacks make virtual patching an attractive option to stay current with defenses as attacker's push the envelope each day.

6. Conclusion

In today's dynamically changing environments, the traditional patch cycle for many organizations simply cannot scale and keep pace with the level and frequency of attacks and vulnerabilities being discovered and exploited today. Through the use of Virtual Patching organizations can help reduce web application vulnerabilities across the board and scale responses and coverage accordingly with appropriate defenses that can be put in place in within minutes or hours instead of a increasingly complicated traditional patch cycle that may consist of days, weeks or sadly in some cases months.

Joseph Faust, josephfaust.dc@gmail.com

Virtual Patching can help lower the application risk profile for an organization by reducing the overall number of vulnerabilities in a given system, and in turn, the potential for loss through strengthening the overall application and network defenses. By incorporating Virtual Patching as another tool in your information security arsenal, a more responsive organizational patch cycle can be provided and attack vector mitigation cycle set. This results in a comprehensive reduction of risk for the organization by better defending the applications and organization data, while keeping pace with quickly evolving threats and attack techniques.

Armed with a dedicated and professional information security staff and a "Defense In Depth" patching approach, organizations can greatly reduce risks through a well thought out Virtual Patching strategy.

Joseph Faust, josephfaust.dc@gmail.com

References

Risk Assessment – Cisco (2010, October 20) Retrieved from http://www.cisco.com/web/about/security/intelligence/vulnerability-risk-triage.html

Epstein, K., Grow, B., & Tschang, C. (2008, April 10). The new e-spionage threat. *Bloomberg Businessweek*, Retrieved from http://www.businessweek.com/magazine/content/08_16/b4080032218430.htm

Chan, J. (2004, January 31). *Essentials of patch management policy and practice*. Retrieved from http://www.patchmanagement.org/pmessentials.asp

Executive Office of the President of the United States, President's Information Technology Advisory Committee. (2005). *Cyber Security: A Crisis of Prioritization* Arlington, Virginia: National Coordination Office for Information Technology Research and Development. Retrieved from http://www.nitrd.gov/Pitac/reports/20050301_cybersecurity/cybersecurity.pdf

Slonim, Y. (2005, December 19). The software quality lifecycle. *Dr. Dobbs*, Retrieved from http://www.drdobbs.com/184407853

"Shrinking Time from Vulnerability to Exploit Top Challenge to Effective Patch Management. (Company overview)." Business Wire. Business Wire. 2006. Retrieved December 05, 2010 from HighBeam Research: http://www.highbeam.com/doc/1G1-143999683.html

Ristic, I. (2010). Modsecurity Handbook. London: Feisty Duck Ltd.

Barnett, R. (2009). WAF Virtual Patching Challenge – Black Hat Briefing. *Proceedings of the Black Hat 2009 DC*, www.blackhat.com/presentations/bh-dc-09/Barnett/BlackHat-DC-09-Barnett- WAF-Patching-Challenge- Whitepaper.pdf

Shinn, M. (2008, January 23). *Virtual patching*. Retrieved from http://www.prometheus-group.com/blogs/36-web-security/106-virtual-patching.html

Risk reduction and compliance through vulnerability management. (2010, November 1) Retrieved from http://www.netisinc.com/brochures/Risk_Reduction_and_Compliance_through_Vulnerability_Management_v7.pdf

Shinn, M. (2007). Advanced intrusion detection with ModSecurity. Proceedings of the

ACM DC PDC, Washington, DC.

AKELA INC. (2010) "What is risk and vulnerability assessment" retrieved from: http://www.akelainc.com/pdf_files/What%20is%20risk%20and%20vulnerability%20assessment.pdf

Joseph Faust, josephfaust.dc@gmail.com