A novel approach to apply security against Cross Site Scripting (XSS) attack by writing the custom rule with ModSecurity Web Application Firewall

Laxman Khokhar[1], Snehal Sathwara[2]

[1]*Department of Computer Engineering, Marwadi University, Gujarat, India*
[2]*Department of Computer Engineering, Marwadi University, Gujarat, India*

---***---

Abstract -Web applications are being more and more targeted for an attack, so effective security of Web applications must be a primary concern. There are most famous web attacks such SQL injection, Cross Site Scripting, Cross Site Request Forgery specified in OWASP (Open Web Application Security Project) Top 10 Web Application Risk. The Cross Site Scripting vulnerability also famous as XSS attack allows attackers to execute malicious codes in the victim's browser that can hijack session, damage the website, or redirect the user to malicious links. To defend web applications from attacks, we can deploy Web Application Firewall (WAF). The Web Application Firewall can filter and monitor the HTTP request and keep web application safe from web attacks. This firewall can work efficiently when it is configured with appropriate rules. Modsecurity allowing to write custom rules to mitigate the vulnerabilities. Hence we can patched the cross site scripting vulnerability via Web Application Firewall.

Key Words:Application Security,*XSS vulnerability, Exploitation, Web Application Firewall, Modsecurity, Reverse Proxy.*

1. INTRODUCTION

Web applications play a vital part in an organization's branding process. With their support, establishing a proper channel of contact between potential clients and the business organization is easier. Therefore web application security is of essential concern. The Cross Site Scripting vulnerability also famous as XSS attack which allows attackers to execute malicious codes in the victim's browser that can hijack session, damage the website, or redirect the user to malicious links. XSS vulnerabilities exist when an application sends untrusted data to a web browser without correct validation or escape. Web applications are vulnerable if it do not ensure that all user-filled input is correctly escaped, or Web applications do not verify that it is safe through input validation, before displaying that user input in the output page. [1]

An attacker can exploit the cross site scripting vulnerability by sending the malicious script to an unsuspicious victim. The victim's browser do not identify the untrusted script as it came from a trusted source. Hence the victim's browser will execute the script and the malicious script can control cookies, session tokens, or other confidential information maintained by the browser and used with that web application. Those scripts can also modify the content of web page. [5]

There are mechanism to apply the security against Cross Site Scripting vulnerabilities by implementing the Web Application Firewall (WAF) such as Modsecurity. We can deploy the Web Application Firewall (WAF) to safeguard web applications. It protect the web applications from various types of attacks such as cross-site-scripting, SQL injection, and cross-site request forgery etc.

The web application firewall (WAF) which will monitor and filter the HTTP(S) request and keep web application safe from web attacks. This firewall will work effectively when it is configured with proper rules. By writing the strong custom rules, it ensures the Web Application Firewall (WAF) correctly detects attacks and also block them if any rule violated or takes appropriate action.[4]

2. OBJECTIVE

The main objective of this research paper is to provide protection against Cross Site Scripting (XSS) attack by implementing Web Application Firewall. The Modsecurity is an open sourceweb application firewall which can block the XSS attacks and keep website safe from such attacks.

3. LITRATURE REVIEW

Victor Clincy and Hossain Shahriar in Web Application Firewall: Network Security Models and Configuration [16] presented the web application firewall rule policy can be categorized in two ways such positive and negative. A positive rule-based web application firewall includes the rules for whitelisting that will only allow traffic in case of pattern matched with the rules and block any unmatched traffic. A negative rule-based web application firewall includes the rules for blacklisting that basically allow all traffic via web application firewall but only block the traffic if the black-listing rule is suspect any malicious pattern matched.Typically, many of the firewall uses either positive or negative rule based policies but they use both together in extreme scenarios. In a positive policy-based web application firewall, the WAF will initially examine the HTTP(S) traffic to make sure the content allowed.Implementing the whitelisting rule involves advance-level research and in-depth experience of web application, as well as ensuring to whitelist the correct and acceptable input values.Fine tuning of positive rule-based web application firewall will take extreme amounts of resources, time and Comprehensive System details.In a negative rule-based web application firewall, the HTTP(S) traffic will be checked, if any data / argument is suspected of being malicious then such HTTP(S) traffic will be blocked.Using this model, itis not be sufficient only because hackers frequently discover new ways of bypassing security policies.For example, blacklisting some input characters cannot be covered as much as hackers could use complicated characters during an attack to bypass validation of the input.

Teddy Mantoro in Log Visualization of Intrusion and Prevention Reverse Proxy Server against Web Attacks [17] proposed WAF based architecture, in which Modsecurity is running with reverse proxy mode. The paper's goal is to define the attack by SQL Injection and provide the appropriate protection from the attack by SQL injection. They have also defined in the paper the log analysis of the traffic handled via Modsecurity. They found in log analysis that few rules are violated against SQL injection attacks, and Modsecurity blocked these HTTP requests.

Trapti Jain and Nakul Jain in Framework for Web Application Vulnerability Discovery and Mitigation by Customizing Rules through ModSecurity [4] presented the new approach for the vulnerability scanning and then mitigating too. They developed the Web Application Scanner which runs the different tools in parallel using multithreading. They have used the tools such as Whatweb, Nikto, Dirb, and Nmap for vulnerability assessment. After that to mitigate the discovered vulnerabilities, they have deployed the web application firewall such as Modsecurity. The Modsecurity is the open source web application firewall which work at application layer. By writing the custom rules web application firewall protects against different vulnerabilities.

4. PROPOSED FRAMEWORK

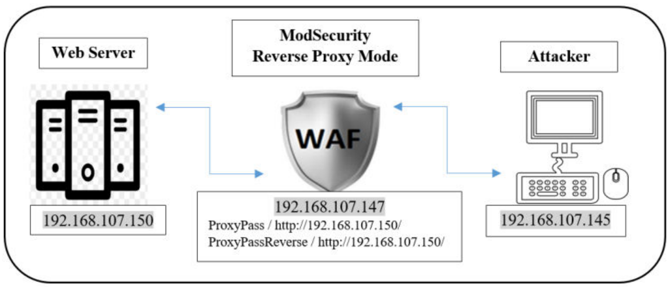

Figure-1: WAF based Architecture

We have a Deliberately Vulnerable Web Application (DVWA) hosted on the web server that serves as our target website and has vulnerable code that can be exploited using XSS attacks [10].

We have configured ModSecurity with Reverse Proxy mode over the firewall machine, so if an attacker tries to access the web server, all HTTP traffic is first routed to the firewall and then redirected to the web server[7][9].

We installed Kali Linux as the operating system on the attacker's machine, and used Burp Suite to preform attacks [11]. The attack vectors are summarized in Table-2.

5. IMPLEMENTATION OF RULE AND ITS WORKING

The Modsecurity can works effectively when it configured with appropriate rules. So here we have created the rule against cross site scripting vulnerability.

- **XSS Rule:**

SecRule REQUEST_URI "<[\w\W]+" "phase:2,log,deny,id:'1995',status:406,t:none,t:htmlEntityDecode,t:removeNulls,t:urlDecodeUni,t:compressWhiteSpace,t:lowercase"

Every Modsecurity rule start with SecRule and contains the variables, operator and actions.
The VARIABLES define the location to examine an HTTP transaction. In above rule we have used REQUEST_URI as variable.

The OPERATOR defines a regular expression, keyword or pattern to monitor in the variable(s).

RegEx means Regular Expression, it is a sequence of characters that allows you to create patterns match with different text. We have use the regular expression here to cover the multiple xss attack payloads in custom rule.

In custom rule, we have used the regular expression such as mentioned in table-1, which cover the multiple xss attack vectors.

Sr. No.	Symbol /Character	Description
1	\w	Match word characters(a-z, A-Z, 0-9)
2	\W	Match Special character, Whitespace
3	+	Occurs one or more times

Table-1: Regex Description

The ACTIONS define what will be done if the rule matches. Below are the action which have been used in rule.

I. phase:2: There are main 4 phases including 1. Request Headers, 2. Request Body, 3. Response Headers, 4. Response Body. Hence we have created the rule which focus on phase 2 such as **Request Body.**
II. log: Logged the HTTP request data.

III. deny: Deny or blocked the HTTP request to transfer to the origin server.
IV. id:'1995': The unique Id which must have allocated to the rule
V. status:406: The status code 406 shown as Apache header status.
VI. t:none: Specifies that no action is require to transform the value of variable used in the rule before matching
VII. t:removeNulls: removing the null
VIII. t:compressWhiteSpace: compress the whitespace
IX. t:lowercase: Converts REQUEST to lowercase

- **How WAF Rule work in attacking environment:**

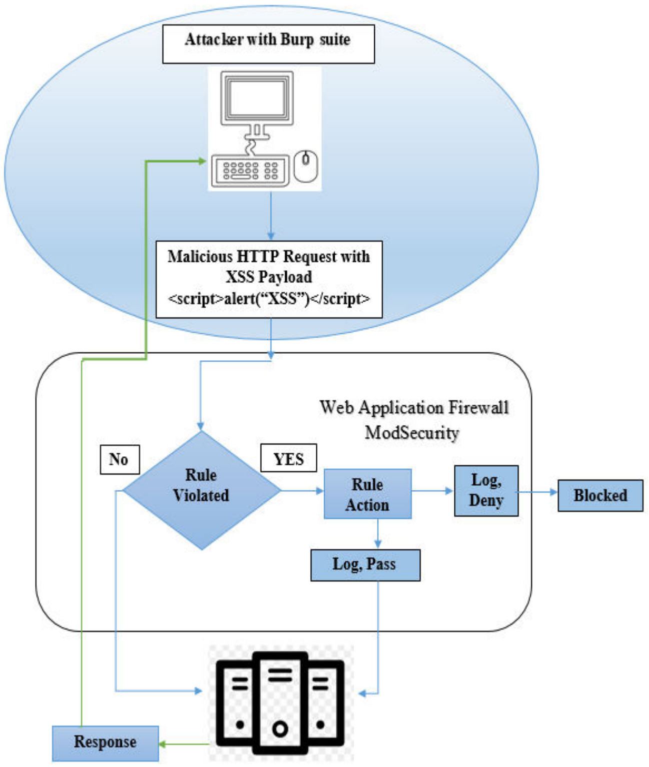
Attacker with Burp suite
Malicious HTTP Request with XSS Payload <script>alert("XSS")</script>
Web Application Firewall ModSecurity
No
YES
Rule Violated
Rule Action
Log, Deny
Blocked
Log, Pass
Response

Figure-2: WAF Rule Work

As shown in Figure-2, When an attacker trying to send malicious request to origin web server then the WAF will be monitoring the HTTP(S) requests and check for suspicious behavior. If WAF found any suspicious behavior or pattern matches against rule then WAF will take the appropriate action and blocked that request according to the rule action. And if WAF detects the legitimate HTTP(S) requests, then those HTTP(S) requests are always passed to the Origin Server.

- List of XSS Attack Vectors :

Sr. No.	Attack Vector
A1.	Laxman
A2.	Harry
A3.	<script> alert("XSS Attack")</script>
A4.	<body onload="alert(11111)">
A5.	<svg onload=prompt("Hacked")>
A6.	<sCrIpT>alert(11111)</sCrIpT>
A7.	<b onclick=alert(1)>click me!
A8.	<img src=x onerror=alert("XSS")>

Table-2: List of Attack Vectors

6. RESULTS AND DISCUSSION

The Log analysis is done in below two scenarios:

1. Before applied Rule
2. After applied Rule

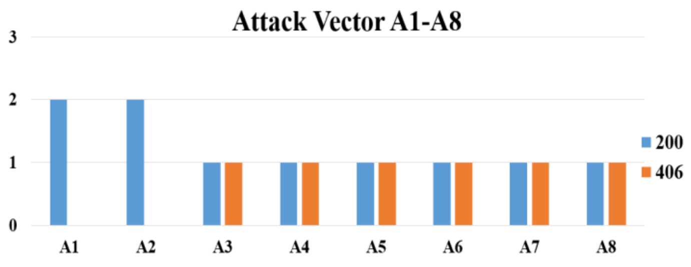

Figure-3: Log Analysis by Attack Vector and Rule

We have taken payloads/attack vectors mentioned in Table-1.In first scenario, where the rule is not applied to the Web Application Firewall, we found that all the A1 to A8 requests were successfully forwarded to the Origin Server which shows 200 status code and XSS attack was performed.

In second scenario, where the rule was applied to the Web Application Firewall, we observed that only the A1 and A2 requests had been forwarded to the Origin Server as it is legitimate traffic and rest requests A3 to A8 are being blocked due to WAF suspected it as XSS attack and shows 406 status code.

7. CONCLUSION

The Cross Site Scripting vulnerabilities are mainly caused by unsuitable filtering of user inputs, such as the incorrect order in which the encoding method is applied. This paper explains the exploitation of Cross Site Scripting vulnerability over DVWA website. Also observed that the Cross Site Scripting vulnerability can be patched via Web Application Firewall such as Modsecurity. The Modsecurity can monitor and filter HTTP requests and blocked the malicious HTTP requests. The Modsecurity allows to write the custom rule which will block such malicious HTTP request and keep website safe from XSS attack. Therefor we have created the custom rule for the Cross Site Scripting vulnerability. We have analyzed that the logs and concluded that the rule is working as expected with zero false positive.

ACKNOWLEDGEMENT

The authors express their gratitude to IndusfacePvt MSS Team. Ltd. to always assist and guide the work of this research.

REFERENCES

[1]. Liu, M., Zhang, B., Chen, W., & Zhang, X. (2019). A Survey of Exploitation and Detection Methods of XSS Vulnerabilities. IEEE Access, 7, 182004-182016.
[2]. OWASP, "Cross-site Scripting (XSS)", 2018. [Online] Available: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS) [Accessed 30-Dec-2019].
[3]. Acunetix, "Types of XSS: Stored XSS, Reflected XSS and DOM-based XSS", 2019. [Online] Available: https://www.acunetix.com/websitesecurity/xss/ [Accessed 30-Dec-2019].
[4]. Jain, T., & Jain, N. (2019, March). Framework for Web Application Vulnerability Discovery and Mitigation by Customizing Rules through ModSecurity. In 2019 6th International Conference on Signal Processing and Integrated Networks (SPIN) (pp. 643-648). IEEE.
[5]. Veracode, "CROSS-SITE SCRIPTING (XSS) TUTORIAL: LEARN ABOUT XSS VULNERABILITIES, INJECTIONS AND HOW TO PREVENT ATTACKS", 2019. [Online] Available: https://www.veracode.com/security/xss [Accessed: 30-Dec-2019].
[6]. Cloudflare, "What is a WAF? | Web Application Firewall explained", 2019. [Online] Available: https://www.cloudflare.com/learning/ddos/glossary/web-application-firewall-waf/ [Accessed: 30-Dec-2019].
[7]. Jesin A, "How To Set Up ModSecurity with Apache on Ubuntu 14.04 and Debian 8", 2015. [Online] Available: https://www.digitalocean.com/community/tutorials/how-to-set-up-modsecurity-with-apache-on-ubuntu-14-04-and-debian-8 [Accessed: 30-Dec-2019].
[8]. Justin Ellingwood, "How To Install Linux, Apache, MySQL, PHP (LAMP) stack on Ubuntu 14.04", 2014. [Online] Available: https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-14-04 [Accessed: 30-Dec-2019].
[9]. Mateusz Papiernik, "How To Use Apache as a Reverse Proxy with mod_proxy on Ubuntu 16.04", 2017. [Online] Available: https://www.digitalocean.com/community/tutorials/how-to-use-apache-as-a-reverse-proxy-with-mod_proxy-on-ubuntu-16-04 [Accessed: 30-Dec-2019].
[10]. Hacking Tools, "Damn Vulnerable Web Application (DVWA)", 2019. [Online] Available: http://www.hackingtools.in/free-download-damn-vulnerable-web-application-dvwa/ [Accessed: 30-Dec-2019].
[11]. PortSwigger Ltd., "Burp Suite", 2019. [Online] Available: https://www.portswigger.net/burp [Accessed 30-Dec-2019].
[12]. Singh, J. J., Samuel, H., &Zavarsky, P. (2018, April). Impact of Paranoia Levels on the Effectiveness of the ModSecurity Web Application Firewall. In 2018 1st International Conference on Data Intelligence and Security (ICDIS) (pp. 141-144). IEEE.
[13]. OWASP, "OWASPTop10-2010-PressRelease", 2010. [Online] Available: https://www.owasp.org/index.php/OWASPTop10-2010-PressRelease [Accessed 30-Dec-2019].
[14]. OWASP, "Category:OWASP Top Ten Project", 2013. [Online] Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#OWASP_Top_10_for_2013 [Accessed 30-Dec-2019].
[15]. OWASP, "Top 10-2017 Top 10", 2017. [Online] Available: https://www.owasp.org/index.php/Top_10-2017_Top_10 [Accessed 30-Dec-2019].
[16]. Clincy, V., & Shahriar, H. (2018, July). Web Application Firewall: Network Security Models and Configuration. In 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC) (Vol. 1, pp. 835-836). IEEE.
[17]. Mantoro, T. (2013, September). Log visualization of intrusion and prevention reverse proxy server against Web attacks. In 2013 International Conference on Informatics and Creative Multimedia (pp. 325-329). IEEE.
[18]. Github, " PayloadsAllTheThings / XSS Injection/ ", 2015. [Online] Available: https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/XSS%20Injection[Accessed 30-Dec-2019].
[19].Kemp, "How To Write A WAF Rule - Modsecurity Rule Writing", 2020. [Online] Available: https://support.kemptechnologies.com/hc/en-us/articles/209635223-How-to-write-a-WAF-rule-Modsecurity-Rule-Writing [Accessed: 30-March-2020].