Detection and Mitigation of Cross-site Scripting using NIDS

Nainita Thakkar[1], Prof. Rajvirsinh Vaghela[2]

[1] *Research Scholar, IT systems and Network Security, GTU PG School, Ahmedabad, Gujarat, India*
[2] *Assistant Professor, Computer Science & Engineering, NSIT, Jetalpur, Gujarat, India*

ABSTRACT

Web application is the application program that keeps on remote server, is accessed globally and delivered over the web through a browser interface. Web Application Security is always issue usually considered because of 60% of attacks are targeting the web application platform. Cross site Scripting may be a vulnerability that found in web application. Cross-site scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into trusted websites. Cross site Scripting happens once associate attacker uses an online application to send malicious code, typically within the type of browser aspect script, to a singular user. One of the most important attacks on technology is XSS attack that is on the third number in Top 10 list of Web Application Security Project (OWASP) in 2017.There are such a lot of ways to prevent this type of attacks. Network Intrusion Detection is one resolution to prevent Cross-site Scripting attacks. This will work for payload recognition and detection of cross-site scripting. To recognize and detect the XSS attack regular expression pattern matching and preprocessing method is used. Thus to improve the result of false positives arises and to prevent it at network level this technique is suggested here. The proposed system is the dual layer security with the use of content security policy as well as Network Intrusion Detection System. The successful implementation of the proposed system will detect and mitigate XSS more precisely.

Keyword: - *Network Intrusion Detection, Cross-site Scripting, Regular Expression, Web Application Security*

1. INTRODUCTION

With the evolution of internet, collected information shared via on-line networking and growing business adoption of web as one approach of doing business and providing service, websites are sometimes attacked directly. Hackers either hunt for to explore to break the corporate network or the end-users accessing on-line internet site by following them to dive-by downloading. As an outcome, business is concentrate on to protection of the online applications themselves additionally to the protection of the online applications themselves additionally to the protection of the hidden network and operative systems. Greater number of online attacks happens via XSS and Structured Query Language injection attacks that sometimes effect from imperfect coding practices and negligence to disinfecting insert to and return from the online application. Attackers will probably use various methods through your application try to damage your business or organization. Every of those methods present a risk which will, or may not, be serious enough to pay attention [7].

1.1 What is Cross-site Scripting?

Cross-site Scripting (XSS) happens whenever associate application take un-trusted knowledge and address it to the browser without verification. This permits hackers to perform malicious scripts within the target browser which might ending in end user session hijacking, spoil web application or send the end user to infected sites. XSS attack is a kind of injection, during which harmful scripts are inserted into trustworthy websites. Cross-site scripting attacks happen once an offender utilizes an online application to send infected code, usually within variety of a browser script, to a unique user. Weakness that permit the attacks to successfully done square measure quite common and happen anyplace an online application utilizes insert from end user inside the outcome develops without verifying and encrypting. Offender will point cross-site scripting to commit infected script to an unaware user. The top user’s browser has no technique to understand that script shouldn’t be trustworthy, and can accomplish the script. As a result of it wonders the script coming from a trustworthy supply, the infected script will hijack any cookies, session tokens, or alternative fragile data maintained by the browser and used there with website.

1.2 Types of XSS:

1) Stored XSS: In this type of XSS, the attacker will insert a malicious script into database. The end-user will send an entirely innocent request to the server. The server will send that malicious data to the end-user in terms of response and bad thing happens.
2) Reflected XSS: In this type of XSS, the attacker will send infected data through browser and it will reflected back in terms of response that will affect the website.
3) DOM-based XSS: The hacker will send infected URL to end-user. After clicking on that link the website is affected[16].

1.3 Content Security Policy:

Content Security Policy is a type of policy that restricts which scripts can be run or loaded on a web page. It is a computer standard introduced to prevent cross-site scripting, click jacking and other code injection attacks resulting from execution of malicious content in the trusted web page context. CSP provides a standard method for website owners to declare approved origins of content that browsers should be allowed to load on that websites covered types are JavaScript, CSS, HTML frames, web workers, fonts, images, embeddable objects such as Java Applets, ActiveX, audio and video files, and other HTML features. CSP can be delivered within HTML code using HTML META tag, although in this case its effectiveness will be limited. Website can declare multiple CSP headers. Each header will be processed separately by the browser. It grants web developers additional control over what location a client browser is permitted to load resources from/other sites are allowed to interact with developer’s site. A correctly configured policy affords you and your user’s additional security against certain attacks. It is used to prevent and mitigate attacks that involve code injection, such as XSS attacks.

2. Related Work

In the paper [1], the authors presented various web application attacks, prevent that attacks and decrease vulnerability of web application. They also discussed that XSS is type of most popular web-attacks. There are different research works done on preventing XSS attacks. In future scope, they discussed to develop a new approach that tracks untrusted data during runtime and randomizes web application source code on both browser and server to prevent XSS attacks. In the paper[2], the author describes how XSS done, impact of XSS ,types of XSS, check whether the site is unprotected to XSS or not different tool to examine cross-site scripting vulnerability and obstructive measures against XSS. In the paper [3], they used CSP which is a strong client-side security layer use in reducing & recognizing web attacks consists of cross-site scripting. By using CSP, web site admin, is vulnerable to error and might need modification in code. They take an approach is introduced to control the restrictions of site admin to use all advantage of CSP which give benefit to more resistant web sites from cross-site scripting. Algorithm is applied as plug-in. Plug-in can be installed on any web application with minimum efforts. Misconfiguration of CSP headers may start to dangerous security issues or purpose the web application to bug. In the paper [4], they discussed according to OWASP and accepted vulnerabilities and risks reported XSS as one of most dangerous vulnerabilities in web application. They proposed a context-sensitive approach based on static taint analysis and pattern matching techniques to identify and control the XSS vulnerabilities in source code of web applications. In the paper [5], the author pay attention on the browser identification mechanism integrated with HTML and CORS properties to identify cross-site scripting attacks with rule based filter by using browser extensions. They design a client-side protection by utilizing browser extension method in Firefox for implementation. The aim of identification system is to digitize the presenting and automatically filter cross-site scripting attacks. In this paper [6], the authors discussed about IDS as first line of protection against network security. Snort is one tool that is used as IDS. They presented how snort implements intrusion detection. The intrusion detection is first and important task in network security system. They presented a snort’s implementation and application from practical point of view. The flow of snort is composed into six parts: catching data package, analyzing code of data, pre-processing the package, parsing the rule detecting engine and logging.

3. Problem Statement

3.1 Problem Identification

Based on the conducted survey we recognized that Cross Site Scripting is the most critical attack performed by the attacker. From the past few years OWASP is declaring the Cross Site Scripting as the top most attack performed in the real time. To detect and mitigate XSS attack is the most common problem.

3.2 Problem Definition

To build a system that can detect and mitigate the XSS attack precisely.

3.3 Proposed Solution

To provide a complete solution for the problem, here we have proposed a system which can detect and mitigate the XSS accurately. The proposed system is dual layer system which will use the Content Security Policy and the Network based Intrusion Detection System to detect and XSS filter to mitigate the XSS attack.

4. Proposed System

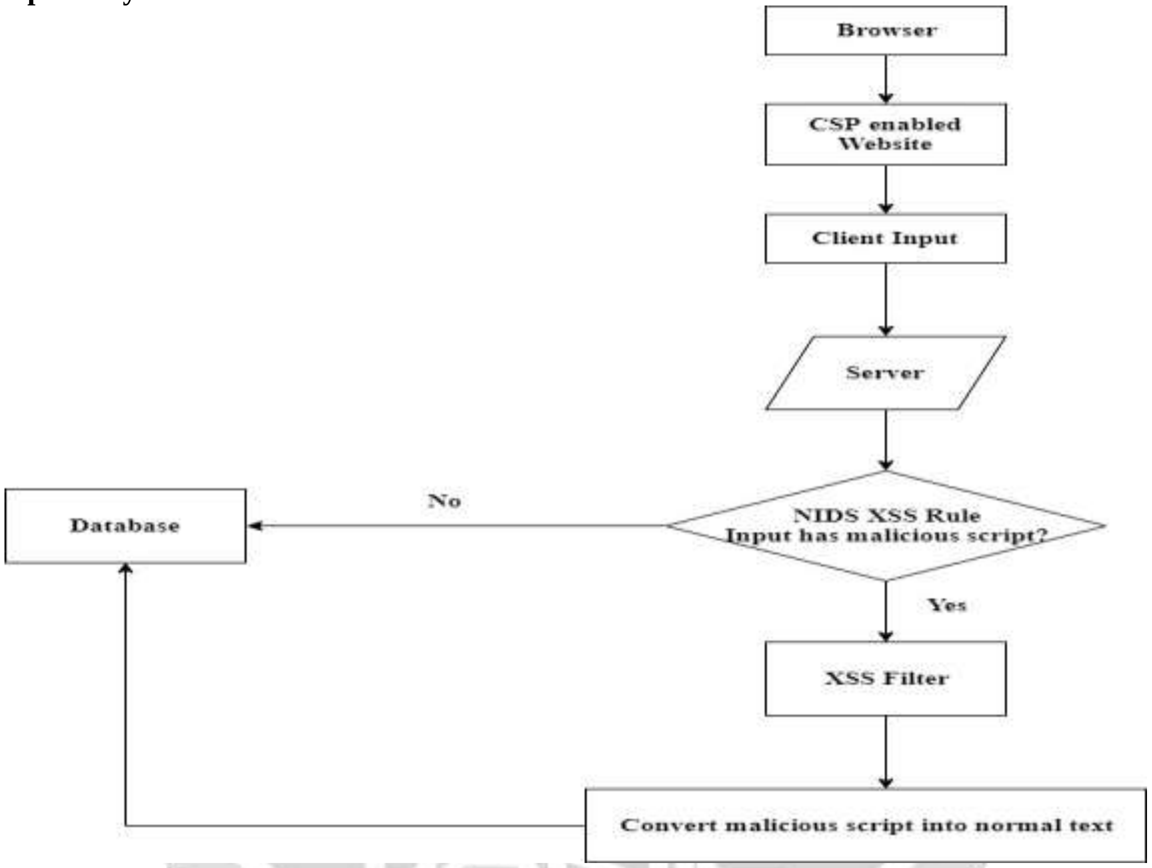

Browser
CSP enabled Website
Client Input
Server
NIDS XSS Rule Input has malicious script?
No
Database
Yes
XSS Filter
Convert malicious script into normal text

Fig-1: Flow Diagram

Above diagram describes the proposed system for detection and mitigation of XSS using NIDS.

Step 1: Start

Step 2: Browser will load the CSP enabled website

Step 3: Client will submit the Input

Step 4: Input submit to the server

Step 5: Server will check the input has malicious script? Using NIDS rules.

Step 6: If (Input has malicious script==true)

Generate log

XSS Filter converts malicious string into normal text

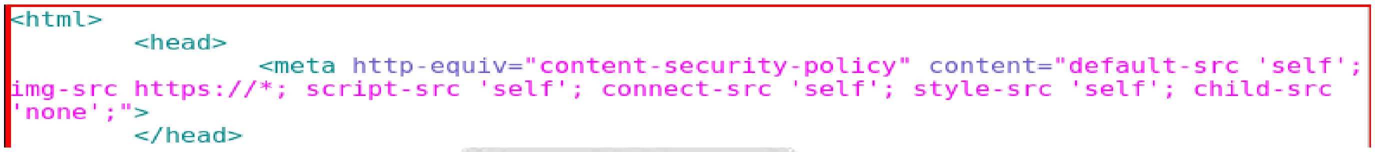
Stored normal text into database

Else Stored into database

Step 7: End

5. Experimental Work

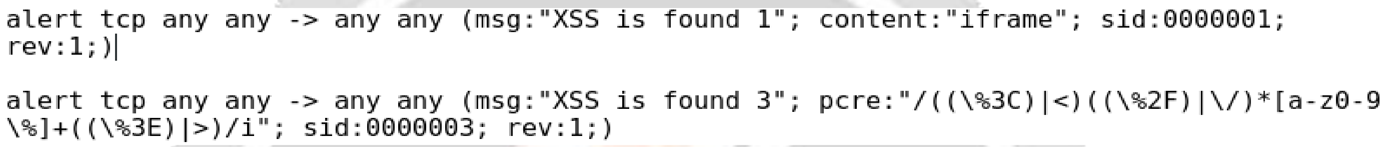
In the first step, CSP is added to the website.



```
<html>
        <head>
                <meta http-equiv="content-security-policy" content="default-src 'self';
img-src https://*; script-src 'self'; connect-src 'self'; style-src 'self'; child-src
'none';">
        </head>
```

Fig-2: CSP defined

Some snort rules are added to the file for detecting XSS.

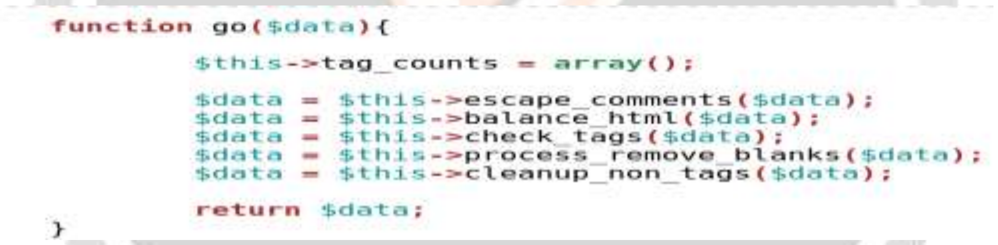


```
alert tcp any any -> any any (msg:"XSS is found 1"; content:"iframe"; sid:0000001;
rev:1;)

alert tcp any any -> any any (msg:"XSS is found 3"; pcre:"/((\%3C)|<)((\%2F)|\/)*[a-z0-9
\%]+((\%3E)|>)/i"; sid:0000003; rev:1;)
```

Fig-3: NIDS rule to detect XSS

XSS filter is applied to filter out the malicious script coming to the database.



```
function go($data){

        $this->tag_counts = array();

        $data = $this->escape_comments($data);
        $data = $this->balance_html($data);
        $data = $this->check_tags($data);
        $data = $this->process_remove_blanks($data);
        $data = $this->cleanup_non_tags($data);

        return $data;
}
```

Fig-4: XSS Filter

6. Result & Analysis

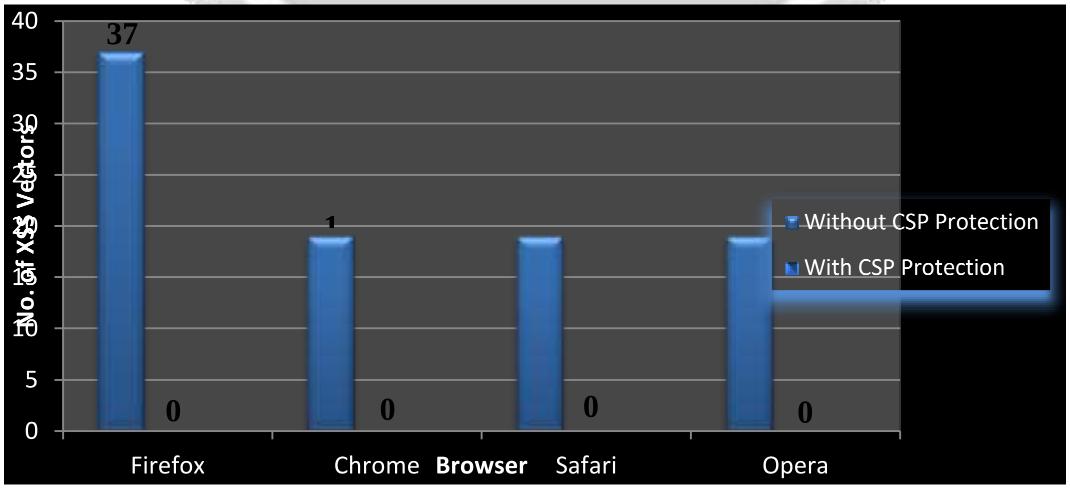
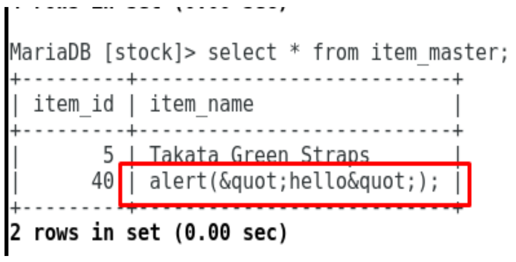


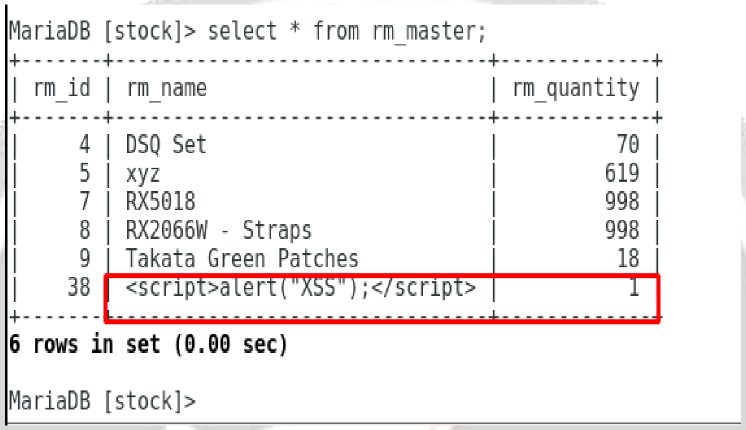
Fig-5: CSP Effect on Different Browsers

Without CSP protection, as many as 37 XSS vectors were successful (Firefox). Even the XSS auditor in Chrome, Safari, and Opera could not eliminate all XSS vectors. However, applying CSP protection eliminated all XSS vectors for each browser.



```
MariaDB [stock]> select * from item_master;
+---------+--------------------------+
| item_id | item_name                |
+---------+--------------------------+
|       5 | Takata Green Straps      |
|      40 | alert(&quot;hello&quot;); |
+---------+--------------------------+
2 rows in set (0.00 sec)
```

Fig-6: XSS filter effect on database



```
MariaDB [stock]> select * from rm_master;
+-------+-----------------------------+-------------+
| rm_id | rm_name                     | rm_quantity |
+-------+-----------------------------+-------------+
|     4 | DSQ Set                     |          70 |
|     5 | xyz                         |         619 |
|     7 | RX5018                      |         998 |
|     8 | RX2066W - Straps            |         998 |
|     9 | Takata Green Patches        |          18 |
|    38 | <script>alert("XSS");</script> |        1 |
+-------+-----------------------------+-------------+
6 rows in set (0.00 sec)

MariaDB [stock]>
```

Fig-7: No XSS filter Effect on database

After applying XSS filter the effect on database is shown in figure.

7. CONCLUSIONS

The proposed system is using the CSP, NIDS and XSS filter that will make it more accurate and robust to detect and mitigate the Cross-site Scripting Attack. This system is high accurate to detect and mitigate the XSS attack. CSP will block the pop-ups on the database, retrieving cookies. After defining the rule for XSS detection, it is able to detect XSS. But it is entered into database as a script of XSS that affect the database. So, after that by applying the filter that can able to mitigate XSS when it is entered into database. This filter can refine all the entries and enter it as a normal text into database. It is a feasible solution to prevent XSS attack.

8. REFERENCES

[1]. M. Khari and P. Sangwan, "Web-application attacks: A survey", IEEE, 2016.

[2]. M. Dayal Ambedkar, N. Ambedkar and R. Raw, "A comprehensive inspection of cross site scripting attack", 2016 International Conference on Computing, Communication and Automation (ICCCA), 2016.

[3]. Samer Attallah Mhana, Jamilah Binti Din and Rodziah Binti Atan, "Automatic generation of Content Security Policy to mitigate cross site scripting", 2016 2nd International Conference on Science in Information Technology (ICSITech), 2016.

[4]. M. Gupta, M. Govil, G. Singh and P. Sharma, "XSSDM: Towards detection and mitigation of cross-site scripting vulnerabilities in web applications", 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2015.

[5]. C. Wang and Y. Zhou, "A New Cross-Site Scripting Detection Mechanism Integrated with HTML5 and CORS Properties by Using Browser Extensions", 2016 International Computer Symposium (ICS), 2016.

[6]. Z. Zhou, Chen Zhongwen, Zhou Tiecheng and Guan Xiaohui, "The study on network intrusion detection system of Snort", 2010 International Conference on Networking and Digital Society, 2010.

[7]. "Top 10-2017 Application Security Risks - OWASP", Owasp.org, 2018. [Online]. Available: https://www.owasp.org/index.php/Top_10_2017-Risk.

[8]. 2018. [Online]. Available: https://www.bespokesoftwaredevelopment.com/blog/what-is-web-application-software/.

[9]. 2018. [Online]. Available: http://www.synerzip.com/owasp-top-10-web-application-security-risks/.

[10]. "OWASP A2 - Broken Authentication and Session Management - GBHackers On Security", GBHackers On Security, 2018. [Online]. Available: https://gbhackers.com/broken-authentication-and-session-management/.

[11]. 2018. [Online]. Available: https://latesthackingnews.com/2017/07/09/web-applications-attacks-insecure-direct-object-reference/.

[12]. 2018. [Online]. Available: https://www.linkedin.com/pulse/20140913122506-4298133-sensitive-data-exposure- a-cxo- not-just-cso-nightmare.

[13]. "All You Need To Know About Cross-Site Request Forgery (CSRF)", Darknet, 2018. [Online]. Available: https://www.darknet.org.uk/2017/07/all-you-need-to-know-about-cross-site-request- forgery-csrf/.

[14]. 2018. [Online]. Available: http://computersecuritypgp.blogspot.in/2016/01/what-is-cross-site-scripting-or-xss.html.

[15]. "Cross-site scripting", En.wikipedia.org, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Cross-site_scripting.

[16]. R. Sillem, V. Sillem and R. Sillem, "Securing Web Applications, Part 3. Cross Site Scripting Attacks", Scott Logic, 2018. [Online]. Available: http://blog.scottlogic.com/2016/02/29/Cross-site-scripting.html.

[17]. Yusof, I. and Pathan, A. (2016). Mitigating Cross-Site Scripting Attacks with a Content Security Policy. Computer, 49(3), pp.56-63.