International Journal of Trend in Scientific Research and Development (IJTSRD)
International Open Access Journal
ISSN No: 2456 - 6470 | www.ijtsrd.com | Volume - 2 | Issue – 1

# Secure Web Applications Against Cross Site Scripting (XSS): A Review

**Vamsi Mohan V**
Department of Computer Science,
School of Engineering and Technology,
Raffles University, Neemrana, India

**Dr. Sandeep Malik**
Department of Computer Science,
School of Engineering and Technology,
Raffles University, Neemrana, India

## ABSTRACT

Cross Site Scripting (XSS) attacks are most common vulnerability issues in the digital era for the Web applications. These attacks occur, when an attacker uses a web application to send malicious code in the form of client side script. These scripts exploit the vulnerabilities in the code and resulting in a serious consequence like theft of cookies, passwords and any confidential user data. In extreme cases, the user may have lost his/her control on the browser. In this paper, we explained detection, and prevention of Cross Site Scripting (XSS) vulnerability attacks through a systematic review process.

***Keywords:*** *Cross Site Scripting, XSS, web security, web proxy, Injection vulnerability, scripting languages security*

## INTRODUCTION

It is common to use client side validations and front-end scripting using JavaScript, VBScript and so on in web applications. Excessive use of these client side scripts increasing the possibilities of serious security vulnerabilities. The most severe threat among the software vulnerability attacks is Cross Site Scripting (XSS). Many of the recent reports on Web Application security reveals Cross Site Scripting (XSS) is one of the common and severe attack. OWASP 2017 has released Top 10 application security risks. In that report, Cross-Site Scripting (XSS) is considered as 3rd position in the vulnerable attacks. Cross Site Scripting Attacks are quite easy to attack and difficult to detect and prevent.

## REVIEW FROM LITERATURE

Suman Saha (2009) described in his publication on Cross-Site Scripting, Web application expands its usages to provide more variety of services and it has become one of the most essential communication channels between service providers and the common users. To augment the user experience, many web applications are using client side scripting languages such as JavaScript, VBScript, and so on. Excessive usage of front-end scripting languages increases the chances of serious security vulnerabilities in web applications, such as cross-site scripting (XSS).

In his survey on Cross Site Scripting, Suman Saha depicted that all the techniques those have been used to detect XSS and arranged wide analyses to evaluate performances of those vulnerability detection methodologies.

S.Shalini, S.Usha (2011) described that in the recent days, Cross Site Scripting (XSS) Attacks become more popular security issue in the modern web applications. These Attacks make use of vulnerabilities in the application, resulting in serious consequences, such as theft of confidential information, cookies, passwords and other user credentials.

S.Shalini, S.Usha mentioned usually, Cross Site Scripting attacks occur when user accessing information in intermediate trusted sites. Front-end scripts act as a web proxy and protect against information leakage from the user environment. Cross Site Scripting (XSS) Attacks are easy to run and execute, but difficult to detect and prevent. In addition

to that, most of the client-side scripts degrade the performance of the application resulting in a poor web surfing experience.

As per Shashank Gupta and Laliten Sharma (2012) Cross Site Scripting attacks on web applications are growing rapidly due to new front-end scripting technologies and frameworks. Cross-Site Scripting (XSS) vulnerabilities are being exploited by the attackers to steal web browser's resources such as cookies, passwords, and other credentials by injecting the malicious JavaScript code on the victim's web applications.

P. Umasankari, E. Uma, & A. Kannan (2013) stated recent reports about web applications reveals that cross-site scripting (XSS) is one of the most common and severe web security defects. It is a type of code injection vulnerability that enables attackers to send venomous scripts to the web clients. It occurs when the web application references the user input in its HTML pages without properly validating the web pages.

## DETECTION OF XSS VULNERABILITIES

Suman Saha, described three distinct types of XSS attacks: non-persistent, persistent, and DOM-based. He explained that non-persistent cross-site scripting vulnerability is the most common type. The attack code is not persistently stored, but, instead, it is immediately reflected to the user.

In his publication Suman Saha, wrote that non-persistent cross-site scripting vulnerabilities can be exploited, for example, by sending to the victim an email with a special crafted link pointing to the search form and containing a malicious JavaScript code. By tricking the victim into clicking this link, the search form is submitted with the JavaScript code as a query string and the attack script is immediately sent back to the victim, as part of the web page with the result.

He explained Persistent type stores malicious code persistently in a resource (in a database, file system, or other location) managed by the server and later displayed to users without being encoded using HTML entities.

He mentioned regarding DOM-based cross-site scripting attacks are performed by modifying the DOM "environment" in the client side instead of sending any malicious code to server. So, the server doesn't get any scope to verify the payload.

According to S.SHALINI, S.USHA, Cross-site scripting or XSS is a web security vulnerability where the attacker injects malicious client side script into the web page. When user visits the web page, the script automatically downloads and run by the web browser. Due to application developers not having awareness or knowledge of security vulnerabilities, XSS become most popular attack. It results poorly developed code riddles with security flaws. JavaScript provide full access to HTML pages using Document Object Model (DOM). Hence, the script can modify the current document exists in arbitrarily. Even it is possible to delete the document and create a new document to send false message to the users.

Shashank Gupta &Lalitsen Sharma narrated, Cross-Site Scripting (XSS) attack is a common vulnerability which is being exploited in web applications through the injection of HTML tags and malicious Java Scripts. A weak input validation on the web application causes the stealing of cookies from the victim's web browser. Attacker hijack the victim's session by stealing the important cookies from the victim's browser.

As Shashank Gupta &Lalitsen Sharma wrote, generally for static detection of XSS, source code analysis will be performed. However, for dynamic testing of XSS, known attacks are executed against the web applications. Researchers have proposed various detection techniques to discover the XSS attacks. Various tools are available to detect the XSS vulnerabilities. To detect XSS vulnerable code in PHP code can be performed by Pixy tool. Many prototype tools have been developed. based on the Pixy tool in the industry.

In their journal, P. Umasankari et al. said, an attacker may inject the malicious scripts via script inputs in the web application's HTML pages. When a client visits the tapped web page, the client's browser not being aware of the presence of malicious scripts shall execute all scripts sent by application resulting in a successful XSS attack. XSS attacks may be the reason for severe security violations.

## PREVENTING XSS VULNERABILITIES

In their study, S.SHALINI, S.USHA, they stated a malicious Web site can employ JavaScript to make the changes to the local system and copy or delete the files.

Shashank Gupta and Laliten Sharma, stated that the existing techniques like filtering of tags and special characters, maintaining a list of vulnerable sites etc. cannot eliminate the XSS vulnerabilities completely.

They both detailed in their publication, XSS attacks are primarily classified into two types. i.e., Persistent and Non-Persistent Attacks.

In case of Persistent also known as stored attacks, the attacker posts the malicious code on the vulnerable web application's repository, to get executed by the victim's browser and attacks it.

In second case, non-persistent attack or reflected attack, the non-persistently stored malicious code on web server immediately displayed by the vulnerable web application back to victim's browser. Through this malicious code gets executed and victim must compromise its browser's resources such as cookies and passwords.

They depicted in their study, there are some types of platforms like Web Goat from OWASP, Acunetix to test or exploit the vulnerabilities of XSS attacks.

Shashank Gupta and Laliten Sharma detailed on preventing XSS vulnerabilities. Cross-site Scripting (XSS) is a top most vulnerability in the web applications, which demands an efficient approach on the server side and client side to protect the users of the web applications. To protect the XSS vulnerabilities, firewalls with security gateway are recommended to have between client and server to check the security pitfalls.

They also discussed about the infrastructure policies in their publication and explained about one of the relevant policy BEEP (Browser-Enforced Embedded Policies), which changes the browser behavior and restricts to execute malicious scripts. Security policies dictate the type of requests to send to BEEP-enabled-browser. Researchers developed another tool called WebSSARI (Web Security via Static Analysis and Runtime Inspection), performs type-based static analysis to identify potentially vulnerable code modules and protect them with runtime guards.

An interesting thing about the client side scripts protection, Shashank Gupta and Laliten Sharma explained, that the researchers have developed the Noxes, which acts as a personal firewall to allow or block connection to websites based on certain predefined rules. Users can white list or blacklist the web sites. When the browser sends a HTTP request to an unknown website, Noxes immediately alerts the client, who chooses to permit or deny the connection, and remembers the client's action for future use.

Another client side approach is proposed by Researchers, which aims to detect the information leakage using tainting of input data in the browser. A mechanism for detecting malicious java script is proposed, in which the browser embedded script auditing component, and IDS that processes the audit logs and compare them to signatures of known malicious behavior or attacks.

Shashank Gupta and Laliten Sharma (2012) narrated to overcome the vulnerable attacks, many client side solutions invented. However, most of them degrade the performance of client's system resulting in poor web surfing experience. The necessity to install updates or additional components on each user's web browser or workstation also degrade the performance of client side solutions.

For mitigating the XSS attacks, P. Umasankari et al. proposed several solutions in their paper. Defensive coding practices, input validation and XSS testing techniques, vulnerability detection techniques are mostly attack prevention techniques. However, these methods, if it is performed manually, are prone to human errors and hard to enforce in existing web applications. Therefore, automation of this task would be beneficial.

In their paper, they proposed an automated approach that statically removes the XSSVs from the program source code. The proposed approach consists of two methods: XSSV Detection and XSSV Removal. XSSV detection method identifies the potential XSSV s in the program source code using static analysis and pattern matching techniques. XSSV removal method identifies the HTML context of each user input referenced in the potential XSSV.

It then secures the potential XSSVs by applying the appropriate escaping methods using escaping library provided by ESAPI. Results show that the approach was effective in securing all the XSSVs found in the subjects by using encoding facilities. Based on the way, the XSS threat is mitigated;

## CONCLUSION

Through Cross-Site Scripting (XSS), the software vulnerabilities are increasing. To prevent such vulnerabilities, it is suggested all developers to follow robust coding standards and follow security guidelines, while coding the applications. It prevents entering unnecessary and dangerous threats from the Internet through browser. Implementing the best practice of

defensive coding is the best solution to stop the software vulnerabilities. These issues motivate the need for a solution to the Cross-Site Scripting attacks (XSS).

**REFERENCES:**


1. G.Wassermann, D.Yu, A.Chander, D.Dhurjati,H.Inamura,Z.Su, **Dynamic Test input generation for the web applications, in: Proceedings of the International Conference on software Testing and Analysis (ISSTA'10),2010**, pp. 249-260.
2. H.Liu, H.B.K.Tan, **Testing input validation in web applications through automated model recovery**. IEEE Journal of System Software 81. (2008). PP- 222-233.
3. J. Garcia-Alfaro and G. Navarro-Arribas, "**Prevention of Cross-Site Scripting Attacks on Current Web Applications,**" Available: http://hacks-galore.org/guille/pubs/is-otm-07.pdf
4. J.H. Hayes, A.J.Offutt ,**Input validation Analysis and Testing, Empirical Software Engineering** 11,(4) 2009. PP- 493-522.
5. P. Umasankari, E. Uma, & A. Kannan (2013), **Dynamic Removal of Cross Site Scripting Vulnerabilities in Web Application**. International Journal of Advanced Computational Engineering and Networking, ISSN: 2320-2106, Volume- 1, Issue- 4, June-2013.
6. Shashank Gupta &Lalitsen Sharma, (2012), **Exploitation of Cross-Site Scripting (XSS) Vulnerability on Real World Web Applications and its Defense**. International Journal of Computer Applications (0975 – 8887), Volume 60– No.14, December 2012.
7. S.Shalini, S.Usha (2011), **Prevention Of Cross-Site Scripting Attacks (XSS) On Web Applications In The Client Side**. IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 1, July 2011, ISSN (Online): 1694-0814.
8. Suman Saha (2009), **Consideration Points: Detecting Cross-Site Scripting**. International Journal of Computer Science and Information Security (IJCSIS), Vol. 4, No. 1 & 2, 2009.
9. OWASP, November 2009, **OWASP Top Ten Project http://www.owasp.org** (Accessed January 2011).
10. Zhushou Tang, Haojin Zhu, Zhenfu Cao, Shuai Zhao, L-WMxD: **Lexical Based Webmail XSS Discoverer, IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)**, 2011, pp. 976-981.