ISSN 2347 - 3983
Volume 11. No.1, January 2023
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter051112023.pdf
https://doi.org/10.30534/ijeter/2023/051112023

An Analysis of the Prevention and Detection of Cross Site Scripting Attack

Priyanshi Panwar[1], Himani Mishra[2], Ritambhara Patidar[3]
[1]Research Student, S.G.I.T.S., Indore, India, priyanshisjr@gmail.com
[2]Assistance Professor, S.G.S.I.T.S., Indore, India, himanimishra.hm21@gmail.com
[3]Assistance Professor, S.G.S.I.T.S., Indore, India, ritam.patidar@gmail.com

Received Date: December 9, 2022 Accepted Date: December 20, 2022 Published Date : January 07, 2023

ABSTRACT

As technology progresses, web based applications are becoming more vulnerable and threats to it's security is increasing day by day. One of the ways by which an attacker can attack any web application is cross site *scripting attack. The* loop holes in a web based application can be exploited by a hacker in ways like, session-hijacking, cookie-stealing, malicious redirection etc. In this survey paper we focuses on the current XSS attack detection techniques and their limitations.

Key words: Cross-Site Scripting attack, prevention, detection, Web based Application, XSS.

1. INTRODUCTION

Web based application are advancing with the advancements in technology. But it also increases the probability of being exploited by the hacker. With the recent surveys and studies it is came to notice, that one of the ways to attack is XSS used by a hacker.

Cross site script attack uses injection method. The hackers takes the advantage of underlying inadequacy of present in web based application by injecting hostile script code through web pages input box, basically java script's code snippets along with input boxes are embedded and then sent at client side and when client reload or revisit the web page it's cookies details can be seen as well as its session token and other sensitive stored information can be seen.

Because web programmes do not effective monitoring and filtering system for the user input, an attacker inserts perilous scripts into reliable web sites. When a malicious script is run, a hacker take hold of the required information. Specifically, the personal user data. Researchers created a number of approaches that can successfully stop the hostile script to get executed and performed because of its practical significance. Additionally, Google Chrome included a client-side XSS filter using this method [1].

2. CSS ATTACK

Genrally, the attacker uses many methods to take advantage of vulnerable web based application. a scenario where an application that allows hacker to send hostile script and gather some type of data from the victim this is what known as cross site scripting attack (xss attack). it allows a hacker to insert malicious programming (such as javascript, vbscript, activex, html, or flash) into a weak dynamic website and then run that script on his computer to collect data. the usage of xss runs the risk of compromising confidential data, stealing or manipulating cookies, creating requests that could be construed as coming from a legitimate user, or executing malicious code on the end-user computers. the information is typically presented as a hyperlink with dangerous content that is disseminated online through all available channels. In this part, we examine the three primary xss attack types: persistent xss attacks, non-persistent xss attacks and dom-based attacks.

2.1 Stored XSS or Persistant or Type-II XSS Attack:

In persistent XSS, an hacker will insert hostile script into a website permanently, which lead to saving the script on the servers as HTML text in places like databases, comment fields, forum postings, etc., and will subsequently be displayed to the victim. When the victim(to be) visits a website that contains XSS-II attack code, the attack code runs on the victims's browser, sending the victims's sensitive data from his side to the hacker's side. The cached XSS attack is another name for the persistent XSS attack. When compared to "REFLECTED XSS," this form of XSS causes more severe damage. Figure 1 shows XSS attack flow.

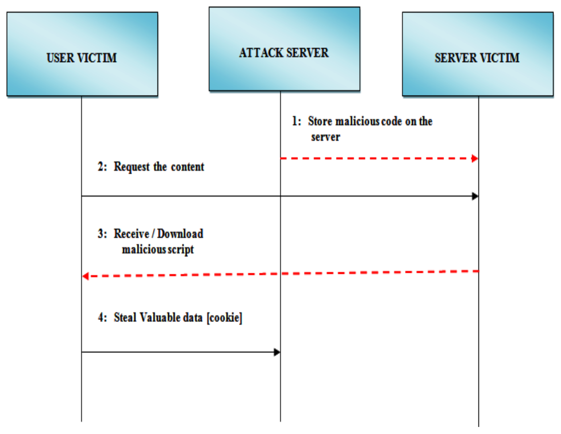
USER VICTIM
ATTACK SERVER
SERVER VICTIM
1: Store malicious code on the server
2: Request the content
3: Receive / Download malicious script
4: Steal Valuable data [cookie]

Figure 1: Flow of Persistent XSS Attack

2.2 Reflected or Non-Persistant or Type-I XSS Attack:

The most frequent kind of XSS attacks is non-persistent cross-site scripting attack. The user is quickly reflected with the attack code rather than it being constantly kept. An alternative name for it is a reverse XSS attack. The attack is transmitted through e-mail or through websites. When a victim clicks on the untrusted link, the information is transferred to the server. The susceptible web application displays both the requested web page and the information that was supplied to it through the link when a victim clicks on the link. The Figure 2 shows flow.

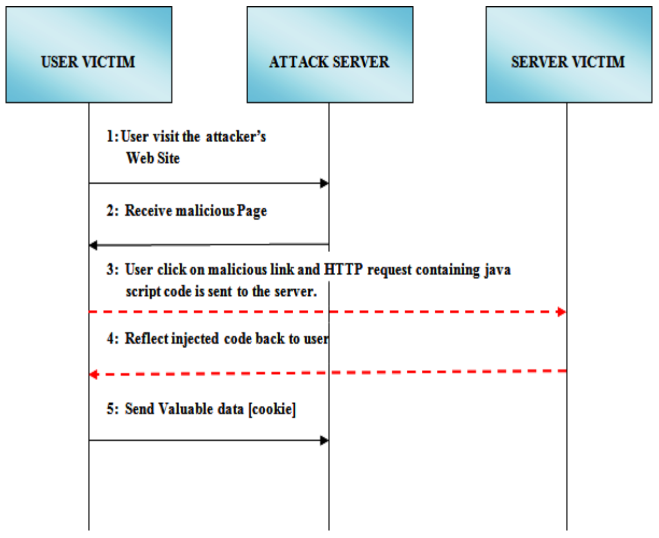
USER VICTIM
ATTACK SERVER
SERVER VICTIM
1: User visit the attacker's Web Site
2: Receive malicious Page
3: User click on malicious link and HTTP request containing java script code is sent to the server.
4: Reflect injected code back to user
5: Send Valuable data [cookie]

Figure 2: Flow of Non-Persistent XSS Attack

2.3 DOM (Document Object Model) - Based XSS or Type-0 Attack:

Instead of delivering harmful code to the server, DOM-based cross-site scripting attacks alter the DOM "environment" on the user end. Document object model, sometimes known as DOM, is a platform- and language-neutral interface. The HTML or XML document can be changed by an attacker's script or programme since DOM permits scripting to do so. Consequently, DOM-based XSS makes use of DOM's weakness to carry out the XSS. Figure 3 shows the flow of DOM based attack.

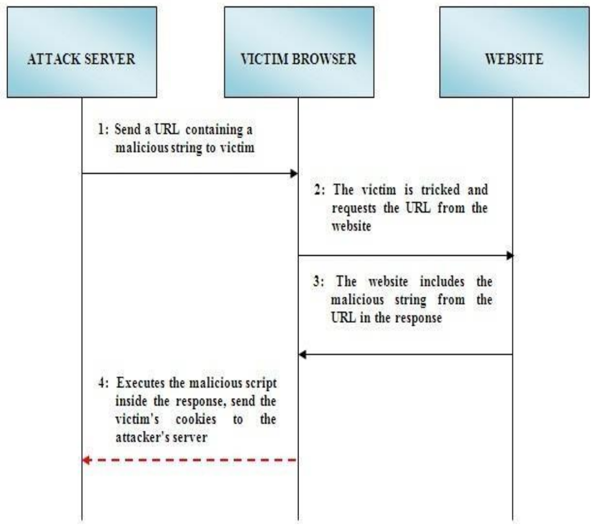
ATTACK SERVER
VICTIM BROWSER
WEBSITE
1: Send a URL containing a malicious string to victim
2: The victim is tricked and requests the URL from the website
3: The website includes the malicious string from the URL in the response
4: Executes the malicious script inside the response, send the victim's cookies to the attacker's server

Figure 3: Flow of DOM-based XSS Attack

3. XSS HAZARDS

Once an attacker knows the loophole in the website which can be victimized, the hacker can launch multiple attacks which includes session hijacking, infiltration, data stealing and propagation of worms[2]. Following section summarizes the harms that a cross site attack can bring:

1. Cookie hijacking: Cookie is basically stores the information about client or web site and is stored at user's computer and it can be accessed through server[3]. The cookie hijacking is done by executing a script in a web page which will lead to hijacking of user's browser, this will send current session detail to hacker web site.
2. Session Hijacking: It will allow the hacker to take control of user's session and then he can perform illegal actions like force posting, illegal transfers etc.
3. Build Get and Post requests: It cannot directly hijack the cookie for usage if the cookie has Http Only property set than it can combine user and client information, but XSS is capable of creating a get, post request in JavaScript to perform the attacks.
4. Defacing Website: "This is the worst vehicle".
5. Inserting hostile content: adding hostile control to a page.

6. XSS Phishing attack: For instance, an attacker can alter a web page's DOM tree structure and content by combining XSS with phishing. The site's login window is simulated by the JavaScript tree code. XSS transmits the user name and password to the server for attack after the user inputs them.
7. XSS worm[4]: For instance, by combining XSS with phishing, an attacker can change the DOM structure and information present in web page. The JavaScript code simulates site's login window. After the user enters them, XSS sends the user name and password to the server for attack.

4. DETECTION AND PREVENTION OF XSS ATTACKS

Detection of XSS attack is significant an aspect to prevent the attack. There are several techniques that has been used for the spotting of the XSS attack. The attack happens when a hostile script is executed at user's end from a web browser. More than 60% of websites are still susceptible to XSS attacks despite the widespread adoption of standardized code development practices in many apps [5].

Static and dynamic detection techniques are two categories of XSS attack detection techniques. Static detection involves examining the program's source code to search for any potential XSS vulnerabilities. Simulating user operations is the method of performing the dynamic testing. It either manually sends injection points or utilizes tools (such as fiddler or burp suite [6, 7], etc.) and analyses the server's response to determine whether an XSS attack has occurred. Many diverse automated tools exists, that are able to perform XSS detection. XSSER[8] is one such tool which is capable for automatic injection testing of XSS attack[8]. Xelenium[9] is also a tool which performs security testing developed by OWSAP. It uses JAVA Swing which helps to find security loophole. Feature matching [10], which runs match checking on input data, is the primary defence against XSS assaults. The apparent problem with this approach is that, hacker can avoid it by adding characters or complete coding. Today, researchers looked at XSS attack detection and protection methods using guidelines from various sources. The S2XS2 tool was created by Shahriar and Zulkernine [10] to identify XSS on the server's end using an automated framework based on injection and policy generation principles. Script code location for content production in the border marker server. For JSP projects, they created a tool to spontaneously construct rules and set boundaries. To examine this approach, they employ four JSP programmes. The findings indicate that this technique can identify the majority of known XSS threats. In 2016, Jinkun Pan and Xiaoguang Mao [11] introduced the DomXssMicro microbenchmark. DomXssMicro is created with six components, it is constructed using templates that were taken from typical vulnerabilities. There are 175 test cases in DomXssMicro, each of which aims to evaluate a different aspect of Dom-based XSS. Various Dom-based XSS detection technologies were also assessed, that might offer direction and a look into a choice and development of detection tool. Kohail[12] used XML and XSD to develop a tool to detect and prevent server side XSS attacks. In this technique they are forcing web application life cycle which will lead to stop unstrusted user input by changing the code structure. But this technique need lot of information from the server to perform well and it will reduce the performance drastically. Gupta [13] , a technique develop by Gupta[12] for automatic detection of vulnerabilities by scanning JavaSript code at different site location, it has three steps firstly he find the site of injection on the server with the help of script locator, then in second step he injects hostile XSS attack vectors with the help of BlogIT. and lastly uses XSS attack list scanning attacks. It's has a disability of not detecting a DOM based XSS. Gandi [14] what he did was, he generated a html script from the server and randomized XML namespace prefix. Next the html tags were annoted with random tags, but for this a policy has to be sent to the browser. This technique is not good to use because browser has to be modified as well as http header is introduced. Hao Chen[15] is more focused on client end and for this he has developed an analyzer that will analyze the flow of execution on AJAX based web application. A client-side analysis JavaScript code is generated to create the FSA, which is then embedded in the browser and agent to keep track of all client application activity. XSS attacks will occur throughout the installation phase if the prebuilt procedure doesn't comply with the FSA. This technology does not call for altering an Ajax application's source code. As a result, XSS attacks may be transparently prevented on any site. Prithvi Bisht[16] work to protect the server side from XSS attack by developing XSS-Guard which is a new framework. To prevent illegal input on the server-side output of the script's content, wish to identify any harmful may be recognised by any input filtering system, and XSS-GUARD uses research on dynamic web applications to ask for any HTML in order to build the script set that will run. J.Sun[17] stops the browser from leaking information by using the stain model and information flow analysis, by rewriting code track framework. For this firstly we have to abstract the semantic of JavaScript and convert the code into Syntax tree for intermediate representation of JavaScript. According to experimental findings, JSTFlow can both ensure the security of sensitive data and find XSS attacks. B.Mewara[18] develops a method which is to filter the reflection based on coding which will help to detect hostile web applications. This method increased detection rate accurately, and it also applies defense mechanism known as XSS-Me to prevent hostile script execution. This solution can easily handle complex attacks. Gupta[19] developed the defense meachanism against XSS for the cloud environment application. CSSXC is a strong defense mechanism against XSS. It basically finds all the injection point that are present in the cloud. The programme makes advantage of an XSS repository's blacklist of JavaScript vectors.

5. CONCLUSION

An emerging approach for application software businesses is the utilization of the web paradigm. It enables the creation of widespread apps that thousands of users with only basic web clients may theoretically access. Furthermore, the development of new tools that are no longer constrained by certain operating systems is made possible by the presence of new technologies for the advancement of online features. When it comes to the online paradigm, the methods currently used to secure traditional apps are often insufficient, and end users are frequently left in charge of safeguarding significant components of any web based service. The web applications which are being developed should use support security tools inorder to make sure a deployment[20] free from malicious code and attack. Cross-site scripting (XSS) vulnerabilities in a web application have been shown to pose a serious danger to both the programme and its users. This paper reviewed current strategies for preventing XSS attacks on susceptible apps. There are now some highly challenging solutions that provides techniques to handle the issue. But some of these methods fall short; some don't offer enough protection. In order to create safe, stable and secure web services, more cogent algorithms and methods need to be developed to provide a better detection and prevention mechanism.

REFERENCES

1. G. Dong, Y. Zhang, X. Wang, P. Wang and L. Liu, **Detecting cross site scripting vulnerabilities introduced by HTML5**, 2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE), Chon Buri, 2014, pp. 319-323.
2. R. Johari and P. Sharma, **A Survey on Web Application Vulnerabilities (SQLIA, XSS) Exploitation and Security Engine for SQL Injection**, 2012 International Conference on Communication Systems and Network Technologies, Rajkot, 2012, pp. 453-458
3. What are cookies. http://www.whatarecookies.com/
4. M. R. Faghani and U. T. Nguyen, **A Study of XSS Worm Propagation and Detection Mechanisms in Online Social Networks**, in IEEE Transactions on Information Forensics and Security, vol. 8, no. 11, pp. 1815-1826, Nov. 2013.
5. A. W. Marashdih and Z. F. Zaaba, **Detection and Removing Cross Site Scripting Vulnerability in PHP Web Application**, 2017 International Conference on Promising Electronic Technologies (ICPET), Deir El-Balah, 2017, pp. 26-31.
6. Ishikawa, Tomohisa, and Kouichi Sakurai. **Parameter manipulation attack prevention and detection by using web application deception proxy**, Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, ACM, Jan 2017.
7. Katy Anton, Jim Manico, and Jim Bird, **Top 10 proactive controls 2016**, OWASP, US, 2016.
8. XSSer: **Cross Site scripter**. [2018-04]. https://xsser.03c8.net/.
9. Xelenium info page. http://www.hackguide4u.com/2012/07/owasp-xelenium-xss-s canner.html.
10. H. Shahriar and M. Zulkernine, "S2XS2: **A Server Side Approach to Automatically Detect XSS Attacks**, 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, Sydney, NSW, 2011, pp. 7-14.
11. J. Pan and X. Mao, **DomXssMicro: A Micro Benchmark for Evaluating DOM-Based Cross-Site Scripting Detection**, 2016 IEEE Trustcom/BigDataSE/ISPA,Tianjin,2016, pp. 208-215.
12. Tawfiq S. Barhoom and Sarah N. Kohail, **A new server-side solution for detecting Cross Site Scripting attack**, International Journal of Computer Information Systems, Vol. 3, No. 2, 2011.
13. Shashank Gupta and B. B. Gupta, **Automated discovery of JavaScript code injection attacks in PHP web applications**, International Conference on Information Security & Privacy (ICISP), Nagpur, INDIA, 11-12 December 2015, Elsevier, Procedia Computer Science, vol. 78, pp.82 – 87, 2016.
14. M. Gundy and H. Chen, **Noncespaces: Using Randomization to Enforce Information Flow Tracking and Thwart Cross-site Scripting Attacks**, Proc. of NDSS, San Diego, Feb. 2009.
15. Q. Zhang, H. Chen and J. Sun, **An execution-flow based method for detecting Cross-site Scripting attacks**, The 2nd International Conference on Software Engineering and Data Mining, Chengdu, China, 2010, pp. 160-165.
16. Prithvi Bisht, V. N. Venkatakrishnan, **XSS-GUARD: Precise Dynamic Prevention of Cross-Site Scripting Attacks**, Department of Computer ScienceUniversity of Illinois, Chicago, 2008, pp. 23-43.
17. W. Xiao, J. Sun, H. Chen and X. Xu, **Preventing Client Side XSS with Rewrite Based Dynamic Information Flow**, 2014 Sixth International Symposium on Parallel Architectures, Algorithms and Programming, Beijing, 2014, pp. 238-243.
18. B. Mewara, S. Bairwa, J. Gajrani and V. Jain, **Enhanced browser defense for reflected Cross-Site Scripting**,

Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization, Noida, 2014, pp. 1-6.

19. Shashank Gupta and B. B. Gupta, **CSSXC: Context-Sensitive Sanitization Framework for web applications against XSS vulnerabilities in cloud environments**, Procedia Computer Science, No. 85, pp. 198-205, Elsevier, 2016.
20. Forrest, S., Hofmeyr, A., Somayaji, A., Longstaff, T.: **A sense of self for unix processes. In: IEEE Symposium on Security and Privacy**, pp. 120–129 (1996)
21. Livshits, B., Erlingsson, U.: **Using web application construction frameworks to protect against code injection attacks**, In: 2007 workshop on Programming languages and analysis for security, pp. 95–104 (2007