REVIEW ON TYPES OF CROSS-SITE-SCRIPTING ATTACKS OVER INTERNET

Yogapriya, R[1] and Subramani, A (Dr)[2]

[1]Research Scholar, [2]Assistant Professor
Dept. of Computer Science,
M.V.Muthiah Govt. Arts College for Women,
Dindigul, Tamilnadu, India - 624005.

Abstract

Cross-site scripting attack is one of the most popular types of threat in web based applications. Recently, the web applications are becoming one of the standard platforms for representing data and their service over the Internet. Since the web applications are progressively utilized for security and services. Cross-site scripting (XSS) occurred by injecting the malicious scripts into web application, and it can lead to significant violations at the website. XSS attacks are the malicious scripts, which are embedded by attackers into the source code of webpage to be executed at client side by browsers. This paper presents vulnerabilities of XSS attack found in the modern web applications.

Keywords: Cross-site scripting; XSS, Attacks; Threats; Vulnerability; Web Application; Internet

I. Introduction

Attackers are constantly working with techniques to obtain sensitive data through the web applications. Applications that are vulnerable to malicious users can break the security and protection mechanism of the system by gaining access to personal information or taking control over system resources. It has been around since the 1990s and cross-site scripting flaws at some point have affected all most major websites like Google, Yahoo and Facebook [1]. XSS is amongst the most widespread of web application vulnerabilities and occurs when a web application makes use of invalidated or un-encoded user input within the output it generates. An attacker does not attack the victim directly, instead of that an attacker would exploit vulnerability within a website that the victim would visit essentially using the vulnerable website as a vehicle to deliver a malicious script to the victim's browser. While XSS has taken advantage of VBScript, ActiveX and Flash (although now considered legacy or even obsolete), unquestionably, the most widely abused is JavaScript primarily it is essential to most browsing experiences.

The purpose of this attack is to get access the personal information and system resource, which may cause damage to assets of individuals and the organizations, which has its existence over the web with exposure of being attacked. Depending upon various factors the level of risk varies among the reported vulnerabilities, Open Web Application Security Project (OWASP) has ranked XSS second most dangerous vulnerability among top ten vulnerabilities. The first attack of XSS was reported in early 90's. Currently XSS holds a

share of 43% among all the reported vulnerabilities [2]. Browser interprets and displays HTML Pages, Java scripts, AJAX and other content hosted on web server. The content, that hosted by web server may be malicious with the intention to target users. The common ways of how the attackers target users through browser as shown in Figure 1. Cookie and Session stealing, browser hijacking, buffer overflow, drive by download and a variety of other ways through which sensitive information maintained by the browser is stolen or access to resource is denied.

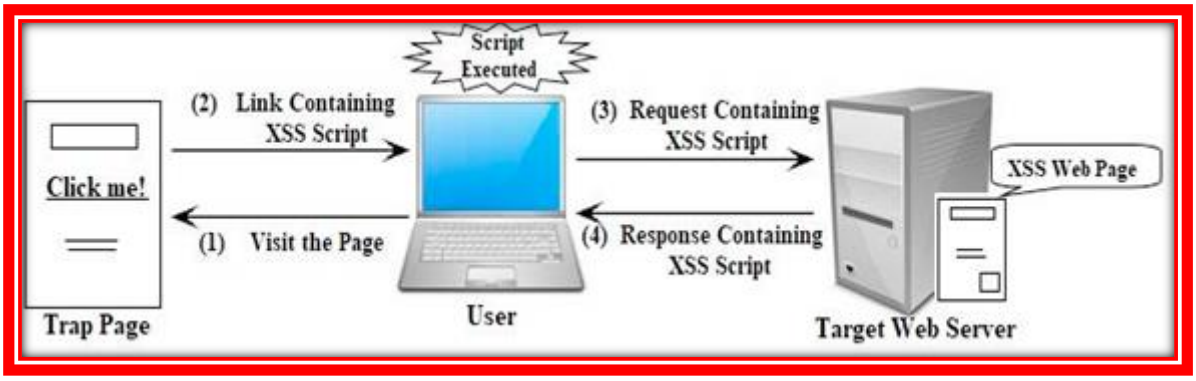

Figure 1: XSS Attacks

II. Literature Review

An application, that is exposed to input validation vulnerabilities, if an attacker finds that the application makes untested assumptions about the type, duration, format, or scope of input data. When inputs are not properly sanitized, attackers are ready to introduce maliciously crafted inputs, which might alter program performances or allow unauthorized access to resources. Improper input validation may invite a range of attacks, like buffer overflow attacks, SQL injection attacks, cross-site scripting, and other code injection attacks [20].

SQL injection attack is the insertion of SQL query through the input data from the client to the application. A successful SQL injection exploit can read and modify sensitive data from the database, implement administrative privileges on the database, and in some cases issue commands to the operating system [21]. SQL injection attacks are a type of injection attack, in which SQL commands are injected into the data-plane input in order to affect the execution of predefined SQL commands [22].

Session management that enables a web application to keep track of user inputs and maintain application status. In web application development, the session management is accomplished through the co-operation between the client and server. Since session ID is the only proof of the client's identity, its confidentiality, integrity, and authenticity need to be ensured to avoid session hijacking [23]. Vulnerabilities, which are specific to session management are great threats to any web application and are also among the most challenging ones to find and fix. The Sessions are targets for attackers because they can be used to gain access to a system without having to authenticate.

In [24], researchers has defined two various methodologies for recognition of cross site vulnerability and deterrence of cross site attack depends on conversion of web operations. Initial stage translates the web functional program structures is completed which currently refined examination methodologies be accessible for that words. It suitably implement the function structure by counting scrutinizes depends on input and output reliabilities obtained by preliminary stage. Utilization of vulnerabilities is restrained by controlling dynamically.

In [25], researchers have surveyed the XSS attack and found that most recent attack on existing websites is DOM based. The attack can harm the millions of people in few seconds as it exploits the vulnerabilities through submission method HTTP GET and HTTP POST. To prevent from this type of attack, a methodology of two way detector and filter is developed which identifies any suspicious URL submitted or stored in database of website and report to filter which is programmed to sanitize the data.

Table 1: State-of-Art Techniques in XSS

Techniques	Deployment	Advantages	Limitations
Noxes a client-side solution for mitigating Cross-Site Scripting attacks [11].	Web-browser	It support such a mitigate practice of XSS attack that considerably diminishes the amount of connection alert prompts and providing the defense against XSS attacks.	Also this tool suffers from low reliability and prohibits the inclusion of benign HTML.
SecuBat a web vulnerability scanner [12].	Web-server	The main goal of this scanner is to determine and exploit application-level vulnerabilities in a large number of real time web sites without human intervention.	The authors have tested this vulnerability scanner on more than 25,000 live web pages. But no ground truth is presented for these web sites.
Session Safe implementing XSS immune session handling [13].	Web-server	A server-side translucent tool does not require any modifications in the source code of web applications and shield against XSS attacks.	Such methods need several, highly structured server domains that may be awkward to handle. Also, they can offer only restricted security such as prohibiting access to the sensitive resources of a web

			application like cookies.
XSSDS a server-side detection of Cross-Site Scripting attacks [14].	Web-server	It has fine capability to discover XSS attacks by measuring the deviation between the HTTP web request and it is associated HTTP response.	The method of discovering persistent XSS attack suffers from few false positives and requires a more advanced training phase for the collection of more scripts.
XSS-GUARD precise dynamic prevention of Cross-Site Scripting attacks [15].	Web-server	Its main strength is that it can evade Illicit script data from being a part of The HTTP response webpage.	While this technique attempt to sanitize unsafe output, it still influence web browser parsers to infer unsafe HTML data and are vulnerable to threats that utilizes browser parse quirks.
BLUEPRINT robust prevention of Cross-Site Scripting attacks for existing browsers [16].	Web-server / web-browser	Blue print demonstrates the method to guarantee the safe construction of the intended HTML parse tree on the web browser .This approach provides security against malicious script injections and facilitates the support for script less for script-less HTML content.	Unfortunately, this technique requires modification both at the client as well as server side. Also the authors had not proposed any idea of handling the unsafe HTML content at the server-side.
SWAP mitigating XSS attacks using a reverse proxy [17].	Web-proxy / web server	It has a fine capability of detecting the deviation between benign and injected JavaScript code.	Many categories of XSS attacks cannot be detected by this technique.
Injecting comments to detect JavaScript code injection attacks [18].	Web-server	Their strength relies on discovering the XSS attacks by inserting the comment statements consisting of random	It has been observed that their proposed technique discovers a part of code injection attack.

		generated tokens and characteristics of being JavaScript code.	
Noncespaces using randomization to enforce information flow tracking and thwart Cross-Site Scripting attacks [19].	Web-browser / web-server	Evade all the troubles and obscurity occurs with sanitization	Does not provide any defensive mechanism regarding inserted JavaScript code downloaded from remote web site.

III. XSS Vulnerability

Moreover the attackers, target end users by encoding the URL and hiding the parameters also use URL's. The types of XSS are, Persistent XSS, Non-persistent XSS and Dom based XSS.

- The Persistent XSS attack is the very powerful attack that can be spread to millions of people at the same time. A malicious script is injected into web application and it is permanently stored on the server. When a user requests to provide information from server, then the injected script send an error message reflected by server.
- Reflected XSS attacks, also known as non-persistent attacks, is the common type of XSS attacks. The attacked code is not persistently stored; instead, it is immediately reflected back to the user. It is also known as reflected XSS attack. In this, the injected code is sent back to the user victim off the server, such as in an error message, search result, or any other response that includes the input which sent to the server as part of the request.

- This attack is typically delivered via emails, social networking sites and malicious links on the website. Then the script is activated through a link, which sends a request to a website with a vulnerability that enables execution of malicious scripts. The vulnerability is typically a result of incoming requests not being sufficiently sanitized, which allows for the manipulation of a web application's functions and the activation of malicious scripts. The link is embedded inside the text that provokes the user to clicking on it, which initiates the XSS request to an exploited website, reflecting the attack back to the user.

- DOM (Document Object Model) is a client side injection. Entire code is originated from the server that means it is developer's responsibility to make a safe web application. A DOM-based XSS attack is triggered on the client side. All XSS attacks

are executed at the browser. DOM allows dynamic scripts, including JavaScript, to reference the document's components. For example, a session cookie or a form field [4].

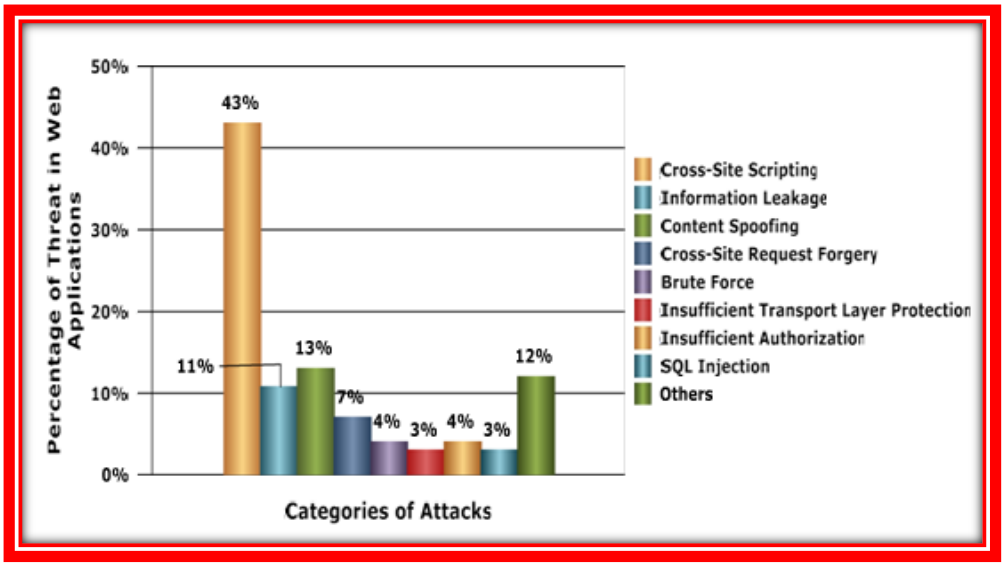

Figure 2: Attack Types

IV. XSS Approach

Despite number of techniques for mitigating XSS have been proposed at either client side or server side, it remains a threat to users. Thus an efficient approach to mitigate XSS is demanded. Kidra at al proposes Noxes, which they claim the first client side web proxy for mitigating XSS that relays all the HTTP requests from browser and serves as an application level firewall. Noxes supports XSS to mitigation mode that significantly reduces the number of connection alert prompts while at the same time that providing the protection against XSS attacks, where the attackers may target sensitive information such as cookies and session IDs. The main imitation of Noxes it demands user customized configuration and user interaction during any suspicious even.

Voget et al proposed a technique, which is a combination of static and dynamic analysis for mitigating XSS which aim to identify the information leakage using tainting of input data in the browser. The problem with this technique is that it does not mitigate the damage caused by other types of XSS attacks such as port scanning, web page defacement and browser resource consumption [7][8][9]. In order of an XSS attack to take place the vulnerable website needs to directly include the user input in its pages. Then an attacker can insert a string that will be used within the web page and treated as code by the victim's browser. The following server-side pseudo-code is used to display the most recent comment on a web page.

```
print"<html>"
    print"<h1>Most recent comment</h1>"
```

```
    printdatabase.latestComment
print"</html>"
```

This script given above is simply print out the latest comment from a database and printing the contents out in an HTML page, that assuming the comment printed out, it consists of only text. The above page is vulnerable to XSS, because an attacker could submit a comment which contains a malicious payload such as <script>doSomethingEvil();</script>.

Users who are visiting the web page will get served the following HTML page.

```
<html>
    <h1>Most recent comment</h1>
    <script>doSomethingEvil();</script>
</html>
```

When the page loads in the victim's browser, the attacker's malicious script will execute, most often without the user realizing or being able to prevent such an attack.

Client Side Prevention

The main disadvantage is that it requires client actions whenever a connection violates the filter rules. This approach addresses all types of XSS attacks. It only detects the send user information to a third-party server, not any other exploit such as those involving Web content manipulation. Noxes, acts as a private firewall, it allows or blocks connections to websites on the basis of filter rules, which are basically user-specified URL white lists and blacklists. The browser sends an HTTP request to an anonymous website then Noxes immediately alerts the client, who chooses to allow or deny the connection, and remembers the client‟s action for prospect use. The Client-side prevention provides a personal protection layer for clients, so that they need not depend on the security of Web applications [4][5][6].

Server Side Prevention

Users of internet specify the prerequisites of sensitive functions i.e. it contain HTML outputs and post conditions of sanitization functions. During the runtime, instrumented guards ensure for conformance of these user-specified conditions. The Web SSARI (Web Security via Static Analysis and Runtime Inspection) tool, which executes the type based static analysis to identify potentially weak code sections and implement them with runtime guards. The Other approaches use dynamic taint-tracking mechanisms to monitor the stream of input data at runtime. They ensure that the inputs are syntactically restricted (only treated as literal values) and do not hold unsafe content defined in user-specified security policies. Some of the server side prevention mechanisms require the collaboration of browsers. An example is BEEP (Browser-Enforced Embedded Policies), a mechanism that modifies the browser, so that it cannot execute unlawful scripts. Security policies that dictate what data the server sends to BEEP-enabled-browsers [5][6].

Runtime Attack Prevention

In general, these methods set up a proxy between the client and the server to capture incoming or outgoing HTTP traffic. Then the proxy checks the HTTP data for illegal scripts or verifies the resulting URL connections against safety policies. XSS defenses focus on preventing the real time attacks using intrusion detection systems or runtime monitors, which can be deployed on either the server side or client side [4][5].

V. Role of Worst Attacker with JavaScript

The consequences of an attacker can do with the ability to execute JavaScript on a web that may not immediately stand out, especially since browsers run JavaScript in a very tightly controlled environment and that JavaScript has limited access to the user's operating system and the user's files. However, when considering that JavaScript has access to the following, it's easier to understand how creative attackers can get with JavaScript.

- Malicious JavaScript has access to all the same objects the rest of the web page has, including access to cookies. Cookies are often used to store session tokens, if an attacker can obtain a user's session cookie, they can impersonate that user.
- JavaScript can read and make arbitrary modifications to the browser's DOM (within the page that JavaScript is running).
- JavaScript can use XMLHttpRequest to send HTTP requests with arbitrary content to arbitrary destinations.
- JavaScript in modern browsers can leverage HTML5 APIs such as accessing a user's geolocation, webcam, microphone and even the specific files from the user's file system. While most of the APIs require the user opt-in, XSS in conjunction with some clever social engineering can bring an attacker a long way.

In the above, the combination with social engineering, allow attackers to pull off advanced attacks including cookie theft, keylogging, phishing and identity theft. Critically, the XSS vulnerabilities provide the perfect ground for attackers to escalate attacks to more serious ones [1][3].

A cookie, also known as a web cookie, browser cookie, and HTTP cookie, is a text string stored by a user's web browser. A cookie consists of the bits of information, which may be encrypted for information privacy and data security purposes. The cookie is sent as a HTTP header by the web server to web browser and then sent back unchanged by the browser. Each time it accesses that server. A cookie can be used for the authentication, session tracking (state maintenance), storing site preferences, shopping cart contents, the identifier for a server-based session, or anything else that can be accomplished through storing textual data. A Cookie is not executable, because they are not executed and they cannot replicate themselves and are not viruses. However, due to the browser mechanism, to set and read cookies, they can be used as a Spyware. Anti-spyware products may warn users about some

cookies because cookies can be used to track the people. Many web applications rely on session cookies for authentication between individual HTTP requests because client-side scripts generally have access to these cookies, simple XSS exploits can steal these cookies. For instance Cookie Grabber.

If the application doesn't validate the input data, the attacker can easily steal a cookie from an authenticated user. All the attacker has to do is to place the following code in any posted input (i.e. message boards, private messages, user profiles)

```
<SCRIPT type="text/javascript">
    varadr = '../evil.php?cakemonster=' + escape(document.cookie);
</SCRIPT>
```

The above code will pass an escaped content of the cookie (according to RFC content must be escaped before sending it via HTTP protocol with GET method) to the evil.php script in "cakemonster" variable. The attacker then checks the results of his evil.php script (a cookie grabber script will usually write the cookie to a file) and use it [1].

VI. Conclusion

In this paper, we have focused on a specific case of attack against web applications. We have seen how the existence of cross-site scripting (XSS for short) vulnerabilities on a web application can involve a great risk for both the application itself and its users. We have also surveyed existing approaches for the prevention of XSS attacks on vulnerable applications, discussing their benefits and drawbacks. Whether it dealing with persistent or non-persistent XSS attacks, there are currently very interesting solutions which provide the interesting approaches to solve the problem. But these solutions present some failures, some do not provide enough security and can be easily bypassed, others are so complex that become impractical in real situation.

References

[1] Learn The Difference Between Injection And Cross-Site Scripting Attacks!, retrieved from https://www.securitycommunity.tcs.com/infosecsoapbox/articles/2018/02/13/learn-difference-between-injection-and-cross-site-scripting-attacks

[2] Cross-site Scripting (XSS), retrieved from, https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

[3] Cross-site Scripting (XSS) , retrieved from, https://www.acunetix.com/websitesecurity/cross-site-scripting/

[4] Shivani Singh1 and Bhawna Kumari2, "Preventing cross-site scripting attacks on the client side" Scholars Research Library Archives of Applied Science Research, 2015, 7 (2):9-14.

[5] Nayeem Khan, Johari Abdullah, Adnan Shahid Khan, "Towards Vulnerability Prevention Model for Web Browser using Interceptor Approach", CITA), 2015 9th International Conference on IT in Asia, At Kuching, Sarawak, Malaysia.

[6] AbdallaWasefMarashdih and ZarulFitriZaaba, "Cross Site Scripting: Detection Approaches in Web Application", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 7, No. 10, 2016, p.no.155.

[7] Michelle E Ruse, SamikBasu, "Detecting Cross-Site Scripting Vulnerability using Concolic Testing", IEEE, 2013 10th International Conference on Information Technology: New Generations.

[8] Shashank Gupta, B. B. Gupta, "XSS-SAFE: A Server-Side Approach to Detect and Mitigate Cross-Site Scripting (XSS) Attacks in JavaScript Code", Research Article - Computer Engineering and Computer Science, 6 October 2015.

[9] Shashank Gupta, Lalitsen Sharma, " 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.

[10] R.Yogapriya, A. Subramani, "A Survey on Vulnerabilities, Attacks and Issues in MANET, WSN and VANET",IJCSE International Journal of Computer Sciences and Engineering,Vol.-6, Issue-11, Nov 2018.

[11] Engin Kirda, Christopher Kruegel, Giovanni Vigna, Nenad Jovanovic Noxes: a client-side solution for mitigating cross-site scripting attacks, SAC 06 Proceedings of the 2006 ACM symposium on Applied computing. Pages 330-337.

[12] Stefan Kals, Engin Kirda, Christopher Kruegel and Nenad Jovanovic, SecuBat: A Web Vulnerability Scanner , WWW 06 Proceedings of the 15th international conference on World Wide Web, Pages 247-256.

[13] Martin Johns, SessionSafe: Implementing XSS Immune Session Handling, ESORICS 06 Proceedings of the 11th European conference on Research in Computer Security, Pages 444-460.

[14] Martin Johns, Björn Engelmann and Joachim Posegga, XSSDS: Server-Side Detection of Cross-Site Scripting Attacks, 24th Annual Computer Security Applications Conference, ACSAC 2008, Anaheim, California, USA, 8-12 December 2008, Pages 335-340.

[15] Prithvi Pal Singh Bisht and V. N. Venkatakrishnan, XSS-GUARD: Precise Dynamic Prevention of Cross-Site Scripting Attacks, Lecture Notes in Computer Science book series (LNCS, volume 5137), pp 23-43.

[16] Mike Ter Louw and V. N. Venkatakrishnan, Blueprint: Robust Prevention of Cross-site Scripting Attacks for Existing Browsers SP 09 Proceedings of the 2009 30th IEEE Symposium on Security and Privacy, Pages 331-346.

[17] Peter Wurzinger, Christian Platzer, Christian Ludl and Engin Kirda, SWAP: Mitigating XSS attacks using a reverse proxy, ICSE 09, 31st International Conference on Software Engineering, IEEE Computer Society, May 16-24, 2009, Vancouver, Canada.

[18] Hossain Shahriar and Mohammad Zulkernine, Injecting Comments to Detect JavaScript Code Injection Attacks, COMPSACW 11 Proceedings of the 2011 IEEE 35th Annual Computer Software and Applications Conference Workshops, Pages 104-109.

[19] Matthew Van Gundy and Hao Chen, Noncespaces: Using Randomization to Enforce Information Flow Tracking and Thwart Cross-Site Scripting Attacks, Proceedings of the Network and Distributed System Security Symposium, NDSS 2009, San Diego, California, USA, 8th February - 11th February 2009.

[20] D. Vandana, Y. Himanshu, and J. Anurag, "A survey on web application vulnerabilities," International Journal of Computer Applications, vol. 108, no. 1, pp. 25–31, 2014.

[21] S. Priti, T. Kirthika, S. Pooja, and S. Bushra, "Detection of SQL injection and XSS vulnerability in web application," International Journal of Engineering and Applied Sciences (IJEAS), vol. 2, no. 3, 2015.

[22] S. Hossain and H. Hisham, "Fuzzy rule-based vulnerability assessment framework for web applications," International Journal of Secure Software Engineering, vol. 7, no. 2, pp. 145–160, 2016.

[23] B. Animesh and M. Debasish, "Genetic algorithm based hybrid fuzzy system for assessing morningness," Advances in Fuzzy Systems, vol. 2014, Article ID 732831, 9 pages, 2014.

[24] M.E Ruse, S. Basu, "Detecting Cross-Site Scripting Vulnerability Using Concolic Testing", Information Technology: New Generations, Tenth International Conference IEEE, USA, pp 633-638, 2013.

[25] Singh, A. and Sthappan,S. , A survey on XSS web-attack and Defence Mechanisms, International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), Volume 4, Issue 3, 2014. ISSN: 2277 128X.