System Protection against Cross-Site Scripting (XSS) Attacks

Shubham Vashisth[1] Ishika Dhall[2]
[1,2]Department of Computer Science
[1,2]Amity University, Noida sector-125, Uttar Pradesh, India

Abstract— Cross-site scripting is chiefly a wide-ranging hacking technique used on websites, the principle behind it is that the attacker can hide scripts behind URL and webpages using <script> i.e. script tag, it's going to be hidden, that means, it won't be visible but still it will be executed allowing the attacker to perform many malicious tasks on web. This paper acquaints the readers with the concept and detailed knowledge about cross-site scripting, its working, preventions, security, and its evolution over the years. This research paper will also elaborate on the main concept of cross-site scripting which is necessary to know while using accessing various features of the web.

Keywords: Cross-Site Scripting; Cybersecurity; Vulnerability; Code-injection; Secured Systems

I. INTRODUCTION

Cross Site Scripting or (commonly known as XSS) is a code injection attack enabling the injection of malicious code into the webpage or website. Nowadays Cross Site Scripting has become one of the most common hacking method or website attack as presented in [1], with almost every website demanding the user to have JavaScript turned on making it the number one vulnerability on the internet.

One of the main reasons that Cross Site Scripting accounts for almost 84% of all the web security vulnerability is that its detection, its detection makes it even harder to figure that which random webpage or website has the maliciously injected code in it or not because basically its hidden but it will still be executed, So we can say that Cross Site Scripting uses website as a medium to attack the user of that particular website rather than being an attacker on that very website.

This vulnerability is basically used by the attackers or hackers with the motive to bypass the access control or steal the session I.D. of the user or the victim leading to very serious damage to one.

The web is based on something named as HTML, i.e. HyperText Markup Language, which follows its own syntax having different tags and the use of angular brackets "<TEXT>", an HTML document starts with an angular bracket like for example: -

<HTML>
……..
……..
</HTML>

Which ends with the angular brackets like this, anything between the angular bracket is read as an instruction so if I want something in italics I use "ITALICS TEXT" (italic tag) to make the text italic or "BOLD TEXT" (bold tag) to make it bold. These angular brackets wherever they are in the document implements that an instruction is coming here, to make it easy to understand we do something known as " Escaping" instead of using an angular bracket we use "<" which ,means when the user reads it, it will become an angular bracket and that worked finely and that is how the internet used to work in the early days of its development i.e. your document will take a second to go and a second to come back and angular brackets won't mess anything up. Therefore, after some years when the internet began to adapt itself among the people, we came to know about more interactive things and then JavaScript gets invented. JavaScript is a programing language that sits in the middle of the webpages using the script tag like for example

<SCRIPT>
……..……....
…….....
</SCRIPT>

Nothing in the above section will appear to the user's screen but actually, you have here a completely separate programing language consisting of variables, operators, calculations and all

These can affect them, it's very complicated and powerful document it's the way every big thing work on the web from games to social media and obviously lot more. The problem with JavaScript is that it's very dangerous. Imagine if whatever you want to do on web you are able to do it on web using JavaScript codes, for example let's take an example of a login page of an online bank, instead of taking the username and password and sending to the bank first it should be sent to someone else and when they have got them then the user should log into the account. This explains the development of cross-site scripting as presented in [2].

If as a web developer one forgets to convert "<" angular brackets into "<" (less than sign), the JavaScript will get executed quietly leading to serious damage to the user.

The main contribution of this paper is the presentation of various inhibition practices against cross-site scripting attacks on web applications. Other valuable contribution of this paper includes the elaborated discussion of various cross-site scripting attacks along with their comprehensive working and consequences.

The rest of the paper has been divided as given: Section II discusses the respective preliminaries; Section III presents various existing categories of cross-site scripting attacks; Section IV is the result that proposes various techniques and procedures to prevent the same. Finally, the conclusion is discussed in Section V.

II. PRELIMINARIES

A. *Client-Side Scripting*

Client Side refers to running scripts, mostly browser by providing it an environment or a platform. In this case, the processing takes place on the client end i.e. the user end. The web server transfers the source code to the client's computer and through the internet and the browser finally run it.

B. *Server-Side Scripting*

Similarly, like Client-Side Scripting as its name suggests Server Side Scripting refers to the web server which is

basically responsible for running of the script unlike Client Side, Server Side Scripting has an advantage of to alter the user's response based upon their requirement like solving queries in data stores or accessing rights.

C. JavaScript

JavaScript is a web development programming language developed by Sun Microsystems with the motive to improve and enhance the websites by adding interactive and dynamic elements into the field of web development. JavaScript is a kind of client-side scripting language, i.e. web browser processes the source code rather than the web server. Similarly, to PHP & ASP which are server-side scripting languages, its code also can be inserted into the webpage anywhere easily.

D. Internet Cookies

Whenever a user visits a website, the web server passes messages to the web browser which are known as cookies which are stored by the web browser n "cookie.txt" file, the basic use of internet cookies is to track the user's activity on the web, for example, your name or one's interests. They are used for many other purposes like identification cards, or for the purpose of online shopping.

E. Session ID

Whenever a user visits any website on the internet, he/she is assigned to a session ID, its job is to specify the time duration of the user on a specific website. When a user closes and opens the browser again, a new session ID is provided to him/her, after long time of inactivity the web server automatically terminates the current session and provides the user with a new unique session ID to ensure privacy, its basic purpose is to ensure that the session of the user on any website is secure and protected in the hands of the user himself.

F. Document object model (DOM)

It is an API for XML & HTML document. It is responsible for defining the logical structure of the document. DOM enables the programmers to build and create documents, performing modification, addition or deletion of the elements and even structure navigation.

III. Categories of Cross Site Scripting

The Cross Site Scripting attacks, although do not have a perfect standard for classification but they are commonly divided into three major types: i.e. Persistent (non-reflected or stored), Non-Persistent (reflected) and DOM-based followed by Self cross-site scripting & mutated cross-site scripting (mXSS) as shown in figure 1.

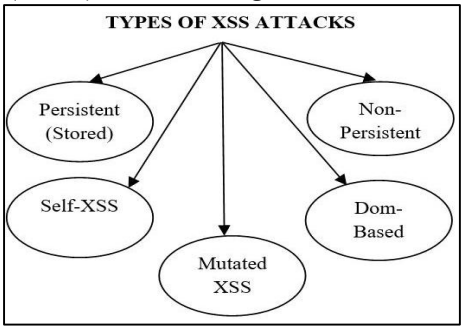

Fig. 1: Types of XSS Attacks

A. Persistent

As its name suggests persistent scripts are the script that persists on the website or webpage. They usually consist of malicious code stored in the website's database as discussed in [3]. The script tag allows the attacker to link a JavaScript file. It goals to target the targeted site to get the permanent code, it requires something which can be interactive like the database or file storage. These scripts don't require the creation of a link to attack the host rather the attacker tries to spread it using spams or other social media platforms.

For performing XSS using Persistent Script the attacker tries to search a website which is more vulnerable for the attack i.e. if it allows the sharing of content among its user,

1) First of all the attacker follows some initial steps by choosing a website to make it work for example social media websites, blogging websites, etc.
2) Once he targets the website, he tries to inject the malicious code into the data which later gets executed.
3) Normally the attacker uses the input fields to inject the code, but he may use a tool that manages to inject codes automatically by itself.

If we assume that there is a random matrimonial website which is vulnerable to XSS attack, it contains the interested people with their private information i.e. their name, address, contact number, email id, etc. which is only visible to the people who have been added by the user in their list. Now suppose a hacker tries to steal the information of some particular user, the field where he is supposed to register or login, he in injects the JavaScript doe into the webpage with instructions to copy all the information he wishes to steal and gets provided to him. When that injected code will be executed, as long as it's a valid code, it will perform its task and will make the attacker get his job done without alerting or providing in sort of clue.

Therefore, Persistent XSS attack has vast consequences, the main reason of it being mostly invisible to the host or the user, most of them fail to detect or use user access controls wisely and secondly the direct insertion of the malicious code into the webpages. The main goal is to steal cookies and data over the internet.

B. Non-persistent

Going by its name a Non-Persistent Script is the one which the only effect's the user running the script. This script indicates that the website's creator may have forgotten to protect the input fields. Non-Persistent Script is harnessed by adding a malicious code at the end of a URL/Link.

For executing a Non-Persistent XSS attack that attack may do the following:

1) Similarly, to the working of Persistent XSS attack, the hacker searches for a vulnerable website which he wants to set as its target.
2) He/She studies the features of the website to determine its security and vulnerability such as-ability to display the searched text, ability to display the username of the logged user, etc.
3) After deciding his targeted website, the hacker injects the code to check whether the script returns in its actual form into the relevant field.

4) The malicious code is placed at the end of the URL so that it's not easy to be identified by the user.

If a hacker wishes to steal the internet cookies and session id of someone innocently with the help of Non-Persistent script, what he will do is that first, he will identify a vulnerable website, after that he will code his own URL and try to embed it into a link and will forward it a group of people by emailing to them the same. This link seems to be normal to the user because of the fact the code is embedded at the end , even if a single person or user clicks on the link many chains of people connected to the host will get affected, as soon as the user clicks on the fake link , he/she gets redirected to the site where the attacker wants them to go, after that the user mostly are fooled and become vulnerable to the attacker getting their information or session id or cookies being hijacked from their own account by the attacker.

Similar to Persistent XSS attack, this attack also have an advantage of hiding its malicious nature or its behavior in front of the user, most of the users are unable to detect the fault in the link which the attacker forwards them directly or indirectly which makes them vulnerable to be hijacked and the hacker comfortably steals all the information from the user's account by getting their session id or internet cookies.

C. DOM-based

DOM-Based XSS attacks completely depend upon the way the HTML page is handled. It's only possible when the HTML page is handled inappropriately. The attacker focuses on the objects of DOM-like document's location, documents.url & document referred in order to manipulate them and perform an XSS attack.

A dom-based XSS attack is implemented by executing the following steps:

1) The attacker creates a link to modify the DOM of the victim.
2) Then the attacker forwards the link to the user.
3) As soon as the victim clicks on the code, the victim gets redirected to some website.
4) The browser of the user creates a DOM object and executes the hacker's script.

If the attacker decides to design the code in order to enable the user to choose his/her preferred language along with a default parameter to provide a default language. Then the attacker embeds the code into a link which is sent to the victim, whenever the victim clicks it, he/she is redirected to some website, there the browser creates the object of the DOM and for that HTML page, which makes the code gets executed and allows the hacker to steal the user's cookies and account information.

As explained in [4], DOM-Based XSS attacks are the most dangerous attacks we have over the internet, 50% of web vulnerability is caused because of it. DOM-Based XSS attack has been declared as a web threat even by the top websites like Google, Alexa & Yahoo. The biggest advantage it has over the other types of XSS attack it's that it can't be restricted by server-side filters because of the "#" as anything after the hash will not be sent to the server.

D. Self-XSS

Self XSS attack depends upon social engineering with a goal to fool the user and making him execute the malicious code of JavaScript by the browser, the fact that this method relies on social engineering does not make it a true XSS attack but its consequences were similar to a regular XSS attack makes it count too.

E. Mutated XSS

In a mutated XSS attack, the hacker injects the code which seems to be safe initially but gets modified at the time of parsing by the browser as presented in [5]. This makes the identification or the detection extremely impossible to sanitize for example adding or rebalancing quotation marks to unquote CSS font-family parameters.

IV. RESULTS

Cross Site Scripting attacks can make web applications and computer systems highly vulnerable, as discussed in [6]. After reviewing the versatile nature of various Cross-Site Scripting attacks, the following techniques may be adopted in order to decrease the chances of a successful XSS attack on one's system.

A. String escaping or encoding

Escaping is considered as the basic or the primary defense mechanism against any form of XSS attack on the website or the webpage, escaping targets on the basic principle of XSS i.e. code injection, the attacker always requires to do code injection in different ways, were escaping stands as an obstacle, the injected code is in JavaScript which gets executed. For example – <font color="green">, this command instructs to print the text in green color, what escaping basically does is that it converts "<" to "<" (Less than sign)

This mechanism doesn't allow the injected code to be a working code as <SCRIPT>, the less than sign is replaced by &lt. thereby forbidding the malicious code to do any sort of damage. This technique is used by every big and invulnerable website like Google or Yahoo.

B. Sanitizing HTML or Javascript Input

A successful XSS attack includes a perfect injected code which contains obviously inputs which are necessary to perform an XSS attack like different tags and symbols, etc. Thus, there should be sanitization of the input which a user inserts in any field before executing it, this refers to the detection of unwanted and extra declaration, removal of them, this will automatically disable the injected code to perform any sort of task on the website or webpage. Thereby to make a website more invulnerable the input given by the user must pass through sanitizing engine for detection of unwanted HTML or JavaScript input.

C. Internet cookie security

Beside sanitizing, filtering content, there are some other technical parameters which need to be emphasized when it comes to XSS attack prevention. The main motive of an XSS attack is to basically steal information, let it be email id, password, account details, private information, etc., internet cookies contains many informative details about the user

which are saved in the "cookie.txt" file as depicted in [7], having information about one's details and interests. By stealing the cookie, the attacker already gets the information to hijack into someone's emails, accounts, and other important stuff.

Therefore, for preventing cookie stealing many of web applications use the concept of a session cookie which is tied to IP address of the originally logged in user, then only it allows the cookie access to that IP.

D. Disabling Scripts

Operations without client-side scripting are provided by very few web applications, most of them don't permit the insertion of unescaped code into the webpages, decreasing the vulnerability to XSS attack. This concept was used by Internet Explorer under an option of "Security Zone", Opera –"Site-Specific preferences." And other browsers like Firefox to increase XSS security. The problem which rose from disabling the script was, it made websites less functional & responsive. Secondly, people were not able to figure out how to secure their browser properly. Moreover, many of the sites do not work without client-side scripting, resulting in people to disable these features for protection making them vulnerable to XSS attack.

E. Emerging Defensive Systems

Three classes are emerging about XSS security, they further include auto-escaping templates, Content Security Policy & JavaScript sandbox tool, these techniques guarantee a reduced XSS attack on the web as presented in [8]. However, highly robust and fully automated defensive systems against Cross-site scripting attacks are yet to come in the near future.

V. CONCLUSION

There is no doubt about the fact that Cross Site Scripting is currently one of the most common vulnerability we have over the internet. Surely, Cross Site Scripting is extremely dangerous because of its consequences and different types of forms it's divided into. Currently there is no website in the whole World Wide Web which can assure that its 100% free from any type of XSS attack, maybe someday there will be or maybe not, the logic which XSS uses, is what the base of any website or webpage is, therefore it's difficult to assure a 100% XSS attack free website. However, by keeping in mind the precautions discussed in this paper we can create a system that is more robust and less vulnerable to cross-site scripting attacks.

REFERENCES

[1] "Full List of Incidents". Web Application Security Consortium. February 17, 2008. Retrieved May 28, 2008
[2] Grossman, Jeremiah (July 30, 2006). "The origins of Cross-Site Scripting (XSS)". Retrieved September 15, 2008.
[3] Yusof, Imran, and Al-Sakib Khan Pathan. "Preventing persistent Cross-Site Scripting (XSS) attack by applying pattern filtering approach." The 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M). IEEE, 2014.
[4] Stock, Ben, et al. "Precise client-side protection against DOM-based cross-site scripting." 23rd {USENIX} Security Symposium ({USENIX} Security 14). 2014.
[5] Wang, Yi-Hsun, Ching-Hao Mao, and Hahn-Ming Lee. "Structural learning of attack vectors for generating mutated xss attacks." arXiv preprint arXiv:1009.3711 (2010).
[6] Wassermann, Gary, and Zhendong Su. "Static detection of cross-site scripting vulnerabilities." 2008 ACM/IEEE 30th International Conference on Software Engineering. IEEE, 2008.
[7] Ter Louw, Mike, and V. N. Venkatakrishnan. "Blueprint: Robust prevention of cross-site scripting attacks for existing browsers." 2009 30th IEEE symposium on security and privacy. IEEE, 2009.
[8] Gupta, Shashank, and Brij Bhooshan Gupta. "Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art." International Journal of System Assurance Engineering and Management 8.1 (2017): 512-530.

