Prevention Of Website from Cross Site Scripting

Amish Mishra, Dr. Sapna Juneja*

* Department Of Computer Science, KIET Group of Institutions, Delhi-NCR, Meerut Road, Ghaziabad

***Abstract*- The Web is an enormous part of many organizations and business activities. It holds hefty amount of confidential data which needed to be secured. Most Common vulnerability threat in Website hacking is Cross Site Scripting. Around 40% attacks on Websites globally occurred due to XSS (Cross Site Scripting). Cross Site Scripting targets both vulnerable and non-vulnerable website. To protect and prevent the website from XSS we must know the complexity and different ways of prevention. However, the research of finding the optimal way of detection of Cross Site Scripting is still in progress. XSS attacks can be used to steal user information, such as cookies or session tokens, which can be used to gain access to the victim's account. XSS can also be used to inject malicious code into a page, which can be used to redirect the user to a malicious site, or to execute malicious code on the user's machine. However, in this paper we will be focusing on XSS attacks and its types. In addition, we will introduce some techniques to prevent user's data from XSS attacks.**

***Index Terms*-** XSS (Cross Site Scripting); Website Security; Cyber Security; Threats on web; Script Injection Attacks;

INTRODUCTION

Cross Site Scripting is commonly known as XSS is a commonly a website security vulnerability which enables attackers to inject malicious scripts into the website. When the user views the webpage, the malicious code is executed which allows the attackers to steal information or perform other malicious actions. XSS can be used to steal sensitive information such as cookies and session IDs, hijack a user's session, or redirect the user to a malicious website. Additionally, it can be used to execute arbitrary code on a victim's computer, potentially allowing the attacker to gain full system privileges (Vogt et al., n.d.). One of the many reasons for the popularity of the
XSS vulnerability is that web-based application developers have little or no security background. (Shalini & Usha, n.d.).

THREATS OF XSS

Session Hijacking: -
An attacker can also use an XSS attack using JavaScript to obtain the victim's session ID. When an attacker sends a victim a malicious link containing malicious JavaScript, when the victim clicks on the link, the JavaScript is executed, performing the attacker's instructions.

Misinformation/Disinformation: -
Attacker can perform any change in the website when he has enough access. Attacker can change as well as delete the content of the website. This can lead to the damage the reputation of the website.

Inserting Hostile Content: -
Such as adding some ActiveX controls on the web page.

Phishing Attacks: -
This type of attack is performed to steal usernames, passwords, credit card information, social security numbers, and/or other sensitive data.

Browser Hijacking: -
By controlling the user's browser, an attacker can add JavaScript code to redirect the user anywhere.

Pop-up Flooding: -
A pop-up is a type of denial-of-service attack in which an attacker opens multiple pop-up windows on a victim's computer, preventing the victim from using the computer or accessing the Internet.

Other Threats: -

Scripts can track what you're doing, such as the history of visited websites and tracking information you've posted on the site, and access to personal data like credit card, Bank account.(Nithya et al., 2015)

RELATED WORK

Until now, there are many defensive techniques to prevent XSS, including the following aspects: static analysis, dynamic analysis, black box testing, white box testing, anomaly detection, etc..(Shalini & Usha, n.d.).

Existing techniques such as tags, filtering special characters, and maintaining lists of vulnerable websites cannot eliminate XSS vulnerabilities.(Gupta & Sharma, 2012)

XSS defense techniques can be largely classified into detection techniques and prevention techniques.

XSS detection techniques focus on identifying gaps in web application code that can lead to vulnerabilities. Most vulnerability detection techniques focus on server-side application code.(Nadji et al., n.d.)

Another client-side approach has been suggested by the researchers and aims to detect information leaks by corrupting browser input. Mechanisms have been proposed to detect malicious Java scripts. In this mechanism, the browser's built-in script monitoring component and its IDS process audit logs and compare them with signatures of known malicious behavior or attacks.

XSS defense techniques can be broadly divided into detection techniques and prevention techniques. XSS detection techniques focus on identifying gaps in web application code that can lead to vulnerabilities. Most vulnerability detection techniques focus on server-side application code.

I. Types Of Xss Attacks

There are three types of XSS attacks:

- Persistent XSS: -
  Persistent XSS involves an attacker submitting malicious code to a web application. This can be done through comments on a forum, blog post, or social media post. Once the malicious code is submitted, it is stored in the web application database. When another user accesses the web application, malicious code is executed and that user is infected. This type of attack is known as stored XSS.

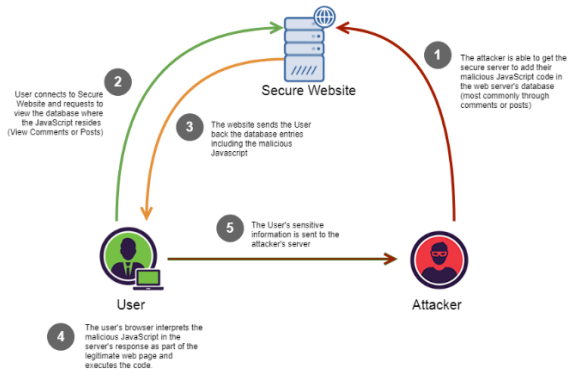

Figure -1

- Reflected XSS: -
  Reflected XSS involves an attacker submitting malicious code to a web application. This can be done through a URL or form submission. The malicious code is then reflected back to the user, typically in an error message. When the user visits the malicious URL or views the malicious form submission, the code is executed and the user becomes infected. This type of attack is called as non-persistent XSS.

Vulnerabilities associated with XSS for web applications

- Allowing untrusted users to submit comments or other content to the website without proper validation. ·
- Failing to properly encode user input before displaying it on the page. ·
- Allowing users to upload files without proper validation, which could lead to script injection. ·
- Embedding third-party content on the website without proper validation. ·
- Creating custom error pages that allow attackers to inject malicious code.

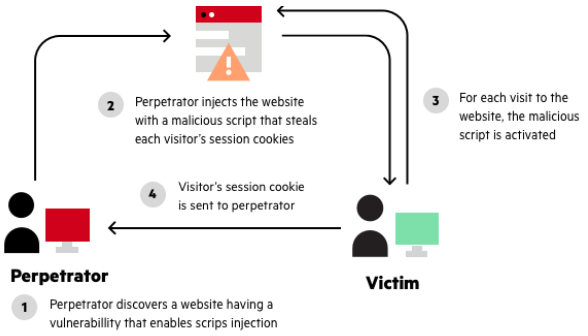

Figure 2

- DOM-based: -
  The DOM is an internal data structure that stores all the objects and properties of a web page. For example, each tag used in the HTML code represents a DOM object. In addition, the website's DOM contains information about properties such as the page URL and meta information. Developers can refer to these objects and properties with JavaScript and modify them dynamically.

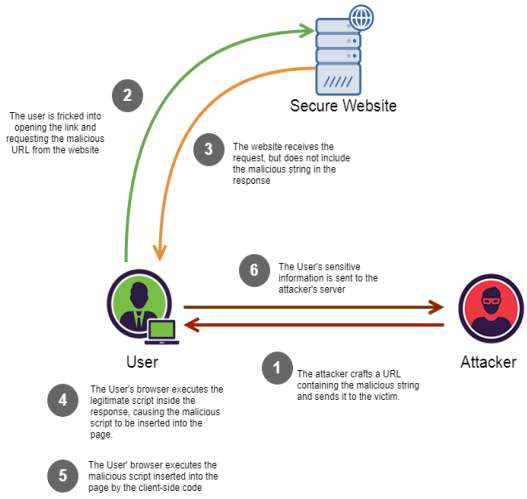

Figure - 3

II. Defense Requirements

Based on these empirical observations, we formulate the following four requirements for Cross-Site Scripting mitigation:

- Defenses should not rely on server-side sanitization of untrusted data. Instead, it should provide a second layer of defense to protect against holes created by error-prone sanitization mechanisms.
- Defenses should restrict untrusted data in a manner consistent with browser implementations and user configurations.
- Defenses should address attacks that target both server-side and client-side languages.
- Defenses should proactively protect against attacks without relying on detecting common symptoms of malicious activity, such as: B. Cross-domain theft of sensitive information. (Nadji et al., n.d.).

III. Proposed Model

This model of XSS prevention is based on a new algorithm that is designed to specifically address the problem of XSS attacks. This algorithm is designed to automatically detect and prevent XSS attacks by analyzing the code of a web application and identifying potential vulnerabilities. This algorithm is also designed to be highly effective against a wide range of XSS attack vectors, including those that are not yet known. The new algorithm is based on a number of factors, including the identification of code that is potentially vulnerable to XSS attacks, the analysis of the code to identify the specific vulnerabilities that are present, and the use of a heuristic to determine the likelihood that an XSS attack will be successful. The model defines when an attacker attacks the website from then the website checks that in input fields there are malicious scripts are present or not. If yes then the access of the website to the user will be restricted and the website will display error log and false request. On the other hand, if the malicious script is not detected then website will intercept the request and display the required information to the user. With the help of this technique the website will be secured from execution of unwanted scripts from the external sources.

X-XSS-Protection directives

A 0 value disables the XSS Filter, as seen below.

X-XSS-Protection: 0;

A 1 value enables the XSS Filter. If a cross-site scripting attack is detected, in order to stop the attack, the browser will sanitize the page.

X-XSS-Protection: 1;

A 1; mode=block value enables the XSS Filter. Rather than sanitize the page, when an XSS attack is detected, the browser will prevent rendering of the page.

X-XSS-Protection: 1; mode=block

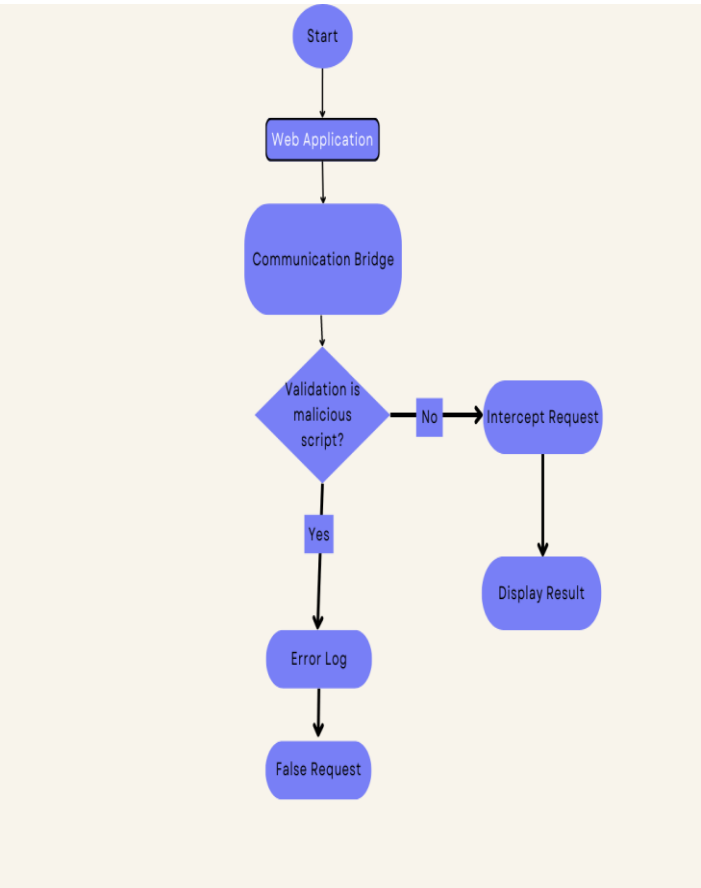

Figure - 4

Conclusion

The proposed solution/approach is very efficient and platform independent. As a result, the website can be prevented by attackers from injecting scripts into the website using input fields. This technique instantly identifies the threat and takes action automatically. Because it stops the execution of JavaScript and other HTML tags on the site by an external user/attacker. Cross-site scripting attacks are one of the most common types of security vulnerabilities on the Internet. All browsers should include its client-side XSS to mitigate unpatched XSS vulnerabilities. Cross-site scripting is a web-based attack technique used to obtain information from a victim's computer or exploit other vulnerabilities to launch additional attacks. These practices use policies, personnel, and technical measures to protect against XSS and other web attacks.

References

I. Exploitation of Cross-Site Scripting (XSS) Vulnerability on Real World Web Applications and its Defense Gupta S, Sharma L *International Journal of Computer Applications (2012) 60(14) 28-33*

II. Secure Web Applications Against Cross Site Scripting (XSS): A Review International Research Secure Web Applications Against Cross Site Scripting (XSS): A Review - Mohan V

III. Document Structure Integrity: A Robust Basis for Cross-site Scripting Defense Nadji Y, Saxena P, Song D

IV. Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis
Vogt P, Nentwich F, Jovanovic N, Kirda E, Kruegel C, Vigna G

V. A survey on detection and prevention of cross-site scripting attack
Nithya V, Lakshmana Pandian S, Malarvizhi C *International Journal of Security and its Applications (2015) 9(3) 139-152*

VI. Riding out DOMsday: Towards Detecting and Preventing DOM Cross-Site Scripting
Melicher W, Das Melicher W, Das A, Sharif M, Bauer L, Jia L Jia L*(2018*

VII. Prevention Of Cross-Site Scripting Attacks (XSS) On Web Applications In The Client Side
Shalini S, Usha S

VIII. Cross-Site Scripting Attack Review Emergency Response View project Energy Prediction of Residential Buildings in Qassim Region Using Machine Learning View project Cross Site Scripting Attack Review
Hussein D, Ibrahim D, Alotaibi A, Alghufaili L

AUTHORS

First Author – Amish Mishra, Bachelor of Technology, Department of Computer Science, KIET GROUP Of Institutions, Delhi-NCR, Meerut Road Ghaziabad, amishmishrami6@gmail.com
Second Author – Dr. Sapna Juneja, Department Of Computer Science, KIET GROUP Of Institutions, Delhi-NCR, Meerut Road.
.