International Journal of Scientific Research and Engineering Development-– Volume 6 Issue 4, July- Aug 2023

Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Cross-Site Scripting (XSS) Attacks and Defense Methods

Chetankumar Nooli*, Akshay Badiger**, Dr.Pijush Barthakur***

*(Master of Computer Application,Visvesvaraya Technological University/KLS Gogte Institute Of Technology , Belagavi Email: chetannooli94@gmail.com)

** (Master of Computer Application,Visvesvaraya Technological University/KLS Gogte Institute Of Technology , Belagavi Email: akshaybadiger007@gmail.com)

***(Master of Computer Application,Visvesvaraya Technological University/KLS Gogte Institute Of Technology , Belagavi Email: pbarthakur@git.edu)

----------**********----------

Abstract:

Cross-Site Scripting (XSS) is a type of hacking attack where bad actors sneak harmful code into websites that you trust. This code can then be unintentionally executed by other users who visit the website, leading to various security issues and potentially allowing the attackers to steal sensitive information or take control of the user's account.

XSS attacks can team up with phishing and other tricks to become even more dangerous. But the scary part is, they are already pretty hard for users to spot because they look real and trustworthy. This makes them a big threat to online services like banks and shopping websites, where attackers can steal sensitive information and cause a lot of trouble for users[1].

The innovative CSP approach is designed to combat real-time XSS attacks by leveraging a combination of tools: Web Application Firewall (WAF), Intrusion Detection System (IDS), Intrusion Prevention System (IPS), and advanced AI algorithms. Unlike traditional methods, CSP shows great promise in quickly identifying and thwarting XSS threats, making it a potent and effective solution. By integrating intelligent AI technologies with robust security measures, this approach has proven to be more reliable and capable of safeguarding online services from the malicious insertion of harmful code into trusted websites. Its success in detecting and neutralizing such attacks marks CSP as a superior and vital defense mechanism in the digital landscape [2].

Web applications are widely used in finance, e-commerce, healthcare, and other fields, making their security a top priority. These applications may have vulnerabilities that attackers exploit to steal user credentials. XSS attacks are particularly critical vulnerabilities that compromise web application security.

Keywords **—Cross-Site Scripting, Hacking attack, Harmful code, Sensitive information, Vulnerabilities, Trustworthy.**

----------**********----------

I. INTRODUCTION

As the number of users using web applications increases, it also makes them more vulnerable to hackers because sensitive data is exchanged more frequently. As a result, there is a critical need to enhance security and privacy measures for web applications to protect against hackers who exploit vulnerabilities by injecting malicious code into trusted websites, aiming to steal or damage valuable information. Despite various research efforts to address this issue, the problem of web application vulnerabilities persists[2].

One common hacking method is Cross-Site Scripting (XSS), wherein attackers inject malicious code into web pages. When the server responds, the

injected script is reflected back to the user, leading to a non-persistent XSS attack, also known as Reflective XSS. Unlike other XSS types, this attack does not store the malicious code on the targeted website permanently. Instead, it relies on user actions to trigger the execution. Although it may not directly steal confidential data, when combined with other attacks, it can create more significant threats and deception, making users vulnerable to unintentionally running harmful scripts[1].

XSS attacks can be categorized based on their impact vector and method of influence. They include Active XSS, which requires no additional user actions; Passive XSS, triggered by user interactions like clicking or hovering; Reflected XSS; Stored XSS; and XSS based on the document object model (DOM)[3].

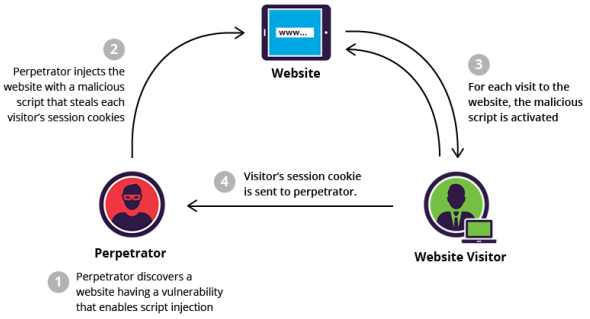

Fig 1 : General idea behind XSS attacks [1]

There are broadly two types of XSS attacks

A. Non-persistent XSS attack
B. Persistent XSS attack
C. Other XSS attack

A) Non-persistent XSS attack

Non-persistent XSS attack is a type of security vulnerability that can be found on some websites. In this attack, the attacker injects harmful code, usually in the form of a script, into the website's search results, error messages, or other responses that are sent back to the user's browser. The injected code is then reflected from the website back to the user's browser and executed without the user's knowledge.

The main objective of this attack is to steal the user's session cookie, which contains important information about the user's login session. For this attack to work, two conditions must be met: the website must allow the injection of scripts, and the user needs to click on a specially crafted link or interact with a manipulated part of the website[2].

Here's how the attack unfolds step by step:

1. The attacker sends a link to the user, containing the malicious script code. This link is often sent through email or some other web-page.
2. If the user clicks on the link, the malicious code becomes part of the request sent to the website's server. The website's security measures may not detect this malicious code.
3. The server processes the request and includes the injected script in the response sent back to the user's browser.
4. When the user's browser receives the response, it unknowingly executes the malicious script.
5. As a result, the executed script sends the user's session cookies to the attacker's domain, granting unauthorized access to the user's private data.
6. The attacker can then store these stolen cookies for future use[5].

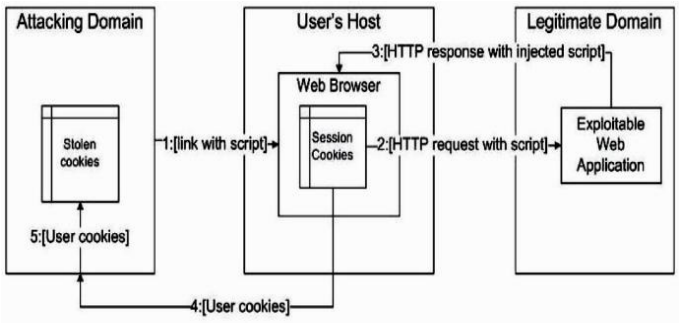

Fig 2: The Non-persistent XSS attack[4]

It's crucial to understand that non-persistent XSS attacks are among the most common types of XSS attacks and are relatively easy to detect. When a successful attack occurs, the user's browser may display a warning or alert indicating that a script is being executed.[3].

To safeguard against such attacks, websites must implement robust security measures to prevent the injection of malicious scripts and ensure that user input is properly sanitized and validated. This way, the risk of falling victim to non-persistent XSS attacks can be significantly reduced [8].

I. Non-persistent XSS vulnerability

Imagine you have found a website called "Natas," which is like a dictionary. It allows you to search for words containing a specific set of letters you enter. For example, if you type "camp" in the search box, it will show you all the words containing "camp" like "campfire,""camping," and so on.

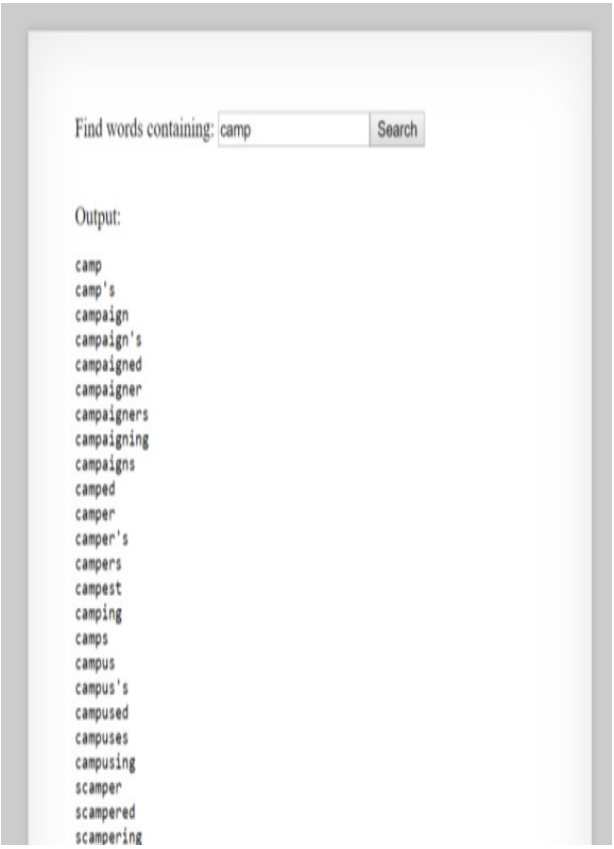

Fig 3: Natas website [1]

Now, when you looked at the website's source code, you found a piece of code that handles the search functionality. This code takes the input you provide and uses a program called "grep" to find the matching words in a file called "dictionary.txt."

```
<pre>
<?
$val = "";
if(array_key_exists("needle", $_REQUEST)) {
    $val = $_REQUEST["needle"];
}
if($val != "") {
    passthru("grep -i $val dictionary.txt");
}
?>
</pre>
```

Here's how the code works:

1. It checks if you have entered anything in the search box (called "needle").
2. If you did enter something, the value you entered is stored in a variable called "$val."
3. Then, the code runs the "grep" command, which searches for the value you entered in the "dictionary.txt" file.
4. If there are any matches, the website displays them as search results.

Now, the problem with this code is that it doesn't validate or check the input you provide properly. It directly uses the value you entered in the "grep" command without any filtering or security checks.

This creates a vulnerability that hackers could exploit. They can craft special input, containing not only regular search words but also harmful commands. When the website processes this malicious input, it will execute those harmful commands as well.

because the website doesn't properly validate the input, it becomes an easy target for exploitation by attackers who can trick the site into running harmful commands. This is why it's crucial for developers to implement proper input validation and security measures to protect against such attacks[1].

B) Persistent XSS attack:

Persistent XSS attacks are a subset of Stored XSS attacks. In this type of attack, the injected script is permanently stored in the server's databases through

various means, like comment fields, logs, or forums. Whenever the victim requests information from the server, the injected script is retrieved and executed, potentially leading to cookie theft or the download of a Trojan horse program.

It works in the following steps:

1. The attacker identifies a vulnerability in a web application and inserts a malicious script code into it.
2. When a user interacts with the application and sends a request, the web page containing the malicious code is accessed.
3. The malicious script is then sent to the user's web browser as part of the application's response.
4. The script gets executed in the user's browser and can steal sensitive information, such as session cookies.
5. The stolen cookies are sent back to the attacker, who stores them on their domain for later use.

Cookies are small pieces of data stored in a user's web browser, containing user-related information. Attackers can exploit XSS vulnerabilities to steal these cookies and gain unauthorized access to specific web resources based on the user's stored data[4].

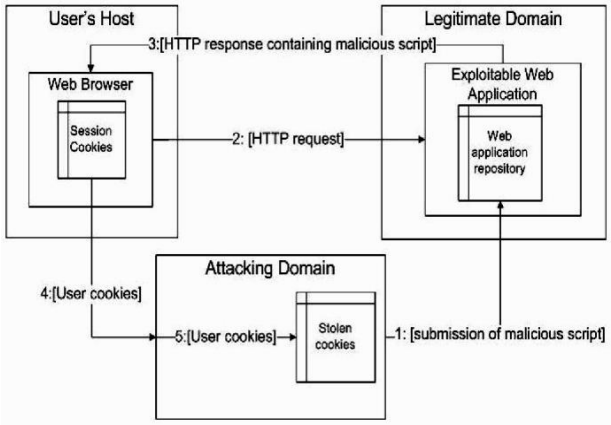

Fig 4: The Stored XSS attacks [4]

I. Persistent XSS vulnerability :

In an XSS attack, attackers can insert malicious scripts into the website, and when other users view the comments, these scripts get executed in their web browsers. The attackers can then steal sensitive information, like session cookies, or perform malicious actions on the users' computers.

In this particular forum, the vulnerability is demonstrated by showing the code snippets responsible for storing and displaying comments. The code responsible for saving comments doesn't verify the input, which means any text, including harmful scripts, can be stored in the database.

Similarly, when the comments are retrieved from the database and displayed on the website, there is no proper validation or filtering of the content. [1,4]This allows attackers to inject HTML tags or scripts into the comments, which will then be rendered and executed when viewed by other users.

To exploit this vulnerability, an attacker could craft a comment containing a malicious script, and when other users view the comment, the script would be executed in their browsers, leading to potential data theft or other malicious activities.

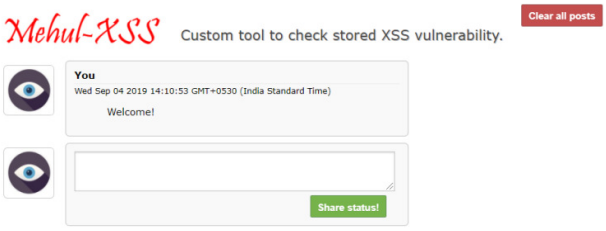

Fig 5: Tool to check Persistent XSS vulnerability [1]

C) DOM based XSS attack:

A DOM-based Cross-Site Scripting (XSS) attack is a type of cyber-attack that targets web applications and websites. In this attack, malicious code is injected into the user's browser, taking advantage of vulnerabilities in the Document Object Model (DOM) rather than the HTML code.

The process of a DOM-based XSS attack can be broken down into the following steps:

1. The attacker sends a harmful link containing a malicious script to the user through email, a bulletin board, or a deceptive webpage.

2. When the user clicks on the link, their browser sends a request to the associated website, and the server responds with a seemingly harmless page.

3. However, the malicious script embedded in the page is executed on the user's browser, allowing it to access sensitive data such as cookies.

4. The attacker then gains access to the user's stolen cookies, which can be used for unauthorized activities.

DOM provides a platform- and language-neutral interface that allows scripts and applications to access and modify the content of HTML and XML documents. Exploiting weaknesses in DOM, the attacker can modify the targeted website's DOM, making it execute the malicious script.

To prevent XSS attacks, it is crucial to detect vulnerabilities in web applications. Various methods and tools like BeEF, Xenotix XSS, Acunetix, XSpider-MAX-Patrol, Nemesida-Scanner, and Wapiti are available for this purpose. However, some existing security tools might be resource-intensive, affecting computer performance.

DOM-based XSS attacks trick users into executing malicious scripts that appear trustworthy and are often embedded in web applications. For instance, attackers may exploit JavaScript code used for form validation, allowing them to inject their harmful code and compromise the user's browser[5].

To protect against DOM-based XSS attacks, web developers should prioritize input validation and data sanitization. Users should exercise caution when clicking on links from unknown sources and keep their browsers and security software up-to-date to minimize the risk of falling victim to such attacks. By staying vigilant and implementing proper security measures, the impact of DOM-based XSS attacks can be significantly reduced.

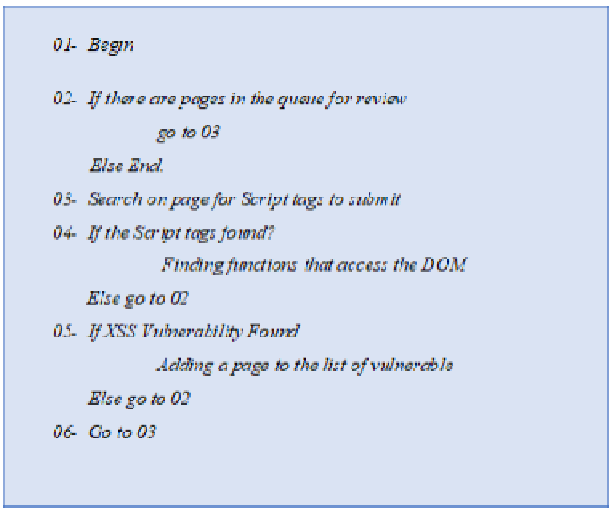

Fig 5: Search Algorithm for XSS vulnerability [3]

XSS Vulnerabilities Detection Software:

In today's digital landscape, data protection software and equipment play a crucial role in safeguarding sensitive information from security threats. However, many of these solutions suffer from cumbersome operation and redundant performance, causing slow device performance and increased resource consumption for users. Additionally, while certain tools can detect Cross-Site Scripting (XSS) vulnerabilities, they come with drawbacks related to web application construction. Notably, web applications often authorize internet resources to expand consumer privileges, granting authorized users' greater access to web resource functionalities compared to unauthorized users.

The existing XSS vulnerability detection software, such as the Online Web Security Scanner, focuses on searching for vulnerabilities only in the open, publicly accessible parts of websites that do not require access authorization. However, thisapproach overlooks the possibility of XSS vulnerabilities existing in hidden or restricted areas of a web resource.

Consequently, there is a pressing need for applications that can thoroughly search for XSS vulnerabilities in web applications. Addressing this demand, this paper presents a program developed in the Delphi programming language, designed to leverage the symmetrical utility of high-quality algorithms for discovering XSS vulnerabilities.

The proposed program offers the following functions:

1. Detection of all types of XSS vulnerabilities, including reflected, stored, and DOM-based XSS.
2. Pre-authorization in web applications and cookie storage to simulate user access.
3. Compilation of internal URLs in a web application to ensure comprehensive vulnerability scanning.
4. Generation of reports detailing the detected vulnerabilities.
5. Provision of actionable recommendations to address the identified vulnerabilities.

To detect reflected XSS vulnerabilities, the paper presents a search algorithm. Reflected XSS involves malicious code embedded in the HTTP response, rather than being stored in the application. The algorithm triggers the submission of forms using the POST method, inserting values in the sent elements and receiving an HTML message in response. The incoming message is analyzed for vulnerabilities. If the analysis reveals that the incoming JS code sets the value stored in the document object model to true (document. vulnerable = true), the page is marked as containing a potential threat of the corresponding type. Otherwise, the page is deemed safe and not added to the final list of vulnerabilities[4].

Detection Techniques

Detection techniques are important for keeping web applications safe from Cross-Site Scripting (XSS) attacks that can harm users. Here are some important ways to detect and prevent XSS attacks[3,2]:

1. Check Inputs Carefully
   Make sure to carefully check and clean anyinformation provided by users on the website. This way, harmful scripts that attackers might try to inject can be blocked before causing any damage.

2. Encode Output Data:
   Before showing any information on the website, convert special characters into safe code. This helps in preventing harmful scripts from running in users' browsers.

3. Set Content Security Policies (CSPs):
   Website administrators can set rules for which sources are safe to fetch content from. By doing this, unauthorized scripts can be blocked, protecting the site from various types of XSS attacks.

4. Use Web Application Firewalls (WAFs):
   Add an extra layer of protection to the web application by using WAFs. These security tools analyze incoming requests and responses for suspicious patterns, blocking potential XSS attacks.

5. Regularly Check for Vulnerabilities:
   Perform frequent security checks to find and fix weak points before attackers can exploit them. This proactive approach keeps the website safe and improves security over time.

6. Test Content Security Policies:
   Before implementing strict CSPs that might block important resources, use CSPRO mode to test and generate reports on potential policy violations. This helps fine-tune CSPs without disrupting the website's normal functions.[6]

By using these detection techniques, web administrators can reduce the risk of XSS attacks and make their web applications more secure for users.

II. CONCLUSIONS

In conclusion, Cross-Site Scripting attacks are dangerous threats to web applications and users. They can lead to data breaches and harm user trust. Detecting and preventing XSS attacks require a proactive approach, using secure coding practices, input validation, and advanced security mechanisms like CSPs and WAFs. By staying informed and implementing robust security measures, website owners can better protect their applications and ensure a safer online experience for users.

REFERENCES

[1] An analytical study on Cross-Site Scripting, 2020.Mehul Singh Department of CSE ASET, Amity University, Noida,India
https://ieeexplore.ieee.org/abstract/document/9132894/
mehulsingh11@gmail.com
Prabhishek Singh Department of CSE ASET, Amity University, Noida
prabhisheksingh88@gmail.com
Dr. Pramod Kumar Department of CSE Krishna Engineering College, Ghaziabad, Uttar Pradesh
drpramoderp@live.com

[2] Detection and Prevention of Cross-site ScriptingAttack with Combined Approaches.
https://ieeexplore.ieee.org/abstract/document/9369796/
Hsing-Chung Chen1,2,*, (Senior Member, IEEE), Aristophane Nshimiyimana1, Cahya Damarjati1,3, Pi-Hsien Chang1,4,*, 1Department of Computer Science & Information Engineering, Asia University No.500, Liufeng Road, Wufeng District, Taichung City, Taiwan 2 Dept. of Medical Research, China Medical University Hospital, China Medical University, Taiwan 3 Dept. of Information Technology, Universitas Muhammadiyah Yogyakarta, Yogyakarta, I ndonesia 4 Information Management Center, Taichung City Government *Corresponding authors: Hsing-Chung Chen (e-mail: cdma2000@asia.edu.tw, shin8409@ms6.hinet.net), Pi-Hsien Chang (e-mail: ps491@taichung.gov.tw)2021

[3] Detection of Web Cross-Site Scripting (XSS) Attacks.Mohammad Alsaffar,Saud Aljaloud, Badiea Abdulkarem Mohammed,Zeyad Ghaleb Al-Mekhlafi,Tariq S. Almurayziq ,Gharbi Alshammari and Abdullah Alshammari, 2022
https://www.mdpi.com/2079-9292/11/14/2212

[4] A Comparative Analysis of Cross Site Scripting (XSS)Detecting and Defensive Techniques. Shaimaa Khalifa Mahmoud Marco Alfonse Computer Science Department, Computer Science Department,Faculty of Computer and Information Sciences, Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt Ain Shams University, Cairo, Egypt shaimaa_khalifa.cs@yahoo.com marco@fcis.asu.edu.eg Mohamed Ismail Roushdy Abdel-Badeeh M. Salem Computer Science Department, Computer Science Department, Faculty of Computer and Information Sciences, Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt Ain Shams University, Cairo, Egypt mroushdy@cis.asu.edu.eg abmsalem@yahoo.com
https://ieeexplore.ieee.org/abstract/document/8260024/

[5] Adaptive Cross-Site Scripting (XSS) attacks and defense mechanisms:classification and state-of-the-artShashank Gupta1 • B. B. Gupta,2023
https://link.springer.com/article/10.1007/s13198-015-0376-0

[6] Automated Detection of Cross-Site Scripting in Websites,2022
Ahmed E, Mohamed K, Hamdy N, Badreldin E, OsamaRMilitary Technical College, Cairo, Egypt,ahmedehab25299@gmail.com,
https://iugrc.journals.ekb.eg/article_302316_2f0dec1b617235f54e54d7b4baf17075.pdfmshaqwieer@yahoo.com,hamdunasser22@gmail.com,badr7189@gmail.com,osamaradwanborhan@gmail.comSupervisor: Dr. Khaled MetwallyMilitary Technical College, Cairo, Egypt, k.metwally@mtc.edu.eg

