A Review On Cross-Site Scripting Request Forgery Attacks And Its Defense Mechanisms

Dr. Purva N. Desai
Assistant Professor
M. B. Patel Science College, Anand.
dr.purva.desai@gmail.com

Abstract:

In today's world, the Internet has become an essential and highly advanced technology, thanks to the use of various electronic devices. The wide range of online services provided by the Internet has greatly contributed to the progress of human civilization, to the extent that life without the internet seems inconceivable. However, the omnipresence of the Internet has also attracted the attention of hackers and attackers, who constantly seek new techniques to exploit vulnerabilities in web applications. According to researchers and industry experts, one of the most significant vulnerabilities found in web applications is Cross-Site Scripting (XSS). This type of attack involves injecting malicious code into a website, which can have detrimental effects on its victims. This paper focuses on two subcategories of XSS attacks: Cross-Site Scripting Redirection and Cross-Site Request Forgery. This study will highlight the significant impact of CSRF attacks on web applications and provide recommendations to protect against such attacks.

1. Introduction:

In this modern era of information and technology, a significant portion of the world's tech-savvy individuals heavily rely on the Internet. The array of services offered by the Internet has changed present civilization to climb its new and wider dimensions. It is a harsh reality that our day-to-day existence would appear unfeasible without the Internet. The prevalence of web sites and customized services in our daily lives cannot be overstated. This increased reliance has also made them a prime target for hackers and attackers. These individuals are constantly on the lookout for new techniques to infiltrate web applications and undermine the advancements made in technology. XSS, in particular, stands out as one of the most critical vulnerabilities found in web applications. XSS is one kind of application layer web attack in which they try to inject malicious scripts to perform malicious actions on any trusted web sites to fulfill only their nominal or bigger self-interest. According to Shalini and Usha (2011) [1], it is called “cross-site” because it involves interaction between two web sites to achieve the attacker's goal. In XSS, malicious code executes on the web browser side which badly affects users. In normal cases, XSS executes when the web page is loaded or an associated event occurs. XSS is not only embedded in JavaScript and HTML, but also in VBScript, ActiveX, AJAX, action scripts like flash or any other browser executable scripting language and mark-up language. For many reasons XSS can be used such as take over user’ account, spread worms, Trojan horse, control access of browser, phishing, expose of the user’s session cookie, redirect the user to some other page or site, modify presentation of content, bypass restrictions, malware attacks & DoS attack, fake advertisement, click fraud, etc.

2. Subcategories of XSS Attacks

Following are the subcategories of XSS attacks which are used to redirect a particular victim unwillingly:

- **XSSR or CSSR**

  XSSR or CSSR, which stands for Cross Site Script Redirection, is a technique utilized to redirect a user to a different webpage without their knowledge. An example of this is when a user triggers a mouse over event and gets redirected to a harmful page. This page might consist of a phishing template, browser attack code, or in certain cases, exploit the data or JavaScript URI scheme for session hijacking.

- **XSRF or CSRF**

  Cross Site Request Forgery, commonly known as XSRF or CSRF (sometimes called C-Surf), is a method used to send automated input from the user to the target site. It is worth noting that XSRF can be initiated by simply viewing a specially crafted image tag. CSRF is also recognized as a one-click attack or session riding. According to Kombade and Meshram (2012) [2], CSRF can be stored CSRF or Reflected CSRF.

Here, in both instances, the victim is redirected to a malicious web page without their consent. These attacks are formed as either stored redirection attacks or reflected redirection attacks. In stored redirection attacks, an attacker injects exploitable links or other contents in the web application itself. In reflected redirection attacks, the attacker sends exploitable links or contents to the victim via message post, e-mails, blog or instant messages. Stored redirection attacks are more effective for attackers because the user who receives the exploitable links or content is currently an authentic user performing web actions without suspicion. In contrast, reflected redirection attacks may not always be successful, as users may not be actively using the target system where the exploitable links and content are accessible. Earlier proposed methodologies only protect against stored redirection attacks but do not protect against reflected redirection attacks because source of attack is not fixed i.e. attack can be delivered via e-mails, chat room, blog, etc.

3. Review of Literatures

Khade *et al.* (2023) [3] proposed mitigation of technique by incorporating machine learning trained random forest algorithm to detect and prevent phishing attacks and CSRF

vulnerabilities in websites. This is a web browser extension which provides real-time protection.

Pardomuan *et al.* (2023) [4] provides server-side XSS detection inspired by Google Chrome's XSS Auditor. 442 out of 500 payloads classified correctly by their model. They achieved 88.4% attack detection accuracy. They highlight that client-side prevention is prone to bypasses due to browser inspection and adversary can manipulate victim to disable security measures.

Lu *et al.* (2022) [5] discussed a fusion verification method which combines traffic monitoring with XSS payload detection by utilizing machine learning. They proposed seven new payload features to improve detection efficiency. This method increases accuracy and classifier's total contribution rate.

Jabiyev *et al.* (2021) [6] proposed a defense approach to protect internal services from Server-Side Request Forgery (SSRF) attacks in a cloud environment. To do so, they extend the functionality of a popular reverse proxy application and deploy a set of vulnerable web applications. They stated that developers have limited awareness about SSRF vulnerability.

Rankothge and Randeniya (2020) [7] introduced A new automated tool to identify and counteract Cross-Site Request Forgery (CSRF) vulnerability. This tool incorporates a secret token pattern to implement a reliable security mechanism on PHP-based web applications, safeguarding content and functionalities without compromising the ability of authenticated users to carry out web activities securely.

Meshram and Balani (2020) [8] described a server-side proxy to detect and prevent attacks in a way that is transparent to users as well as to the web application itself and can be used to secure a number of popular open-source web applications, without negatively affecting their behavior. According to them, existing mitigation approaches are time-consuming and error-prone.

Shaikh (2019) [9] studied that traditional anti-virus and anti-spyware approaches fail to detect CSRF vulnerabilities and suggest urgency for appropriate detection and defense mechanisms against CSRF. He used a methodical approach to investigate CSRF attacks. He introduced a

novel distinctive set of algorithms that use intelligent assumptions to detect and defend CSRF. In this work, design details of a CSRF Detection Model (CDM), implantation and experimentation results of CDM are elaborated to detect, predict and provide solutions for CSRF attacks on contemporary web applications and web services.

Kavitha and Ravikumar (2015) [10] proposed an algorithm which protects against clickjacking. This algorithm follows a regex approach. In this proposed architecture, authors check all iframe available in a web page then it also checks IP address as well as domain. If a user enters a URL, it checks against the iframe URL using regex. System also maintains a list of DNS as DNSloopup. If more than 10 requests are found as bad requests, this user IP is blocked.

Sentamilselvan *et al.* (2014) [11] proposed an algorithm for stored attacks and login CSRF attacks. To prevent stored attacks, a pattern recognition method is used. To prevent login CSRF, a 2-step verification method is used. For this verification, a random number is sent to the user via mail or SMS by the server. Using this random number, one can view his/her home page.

4.Attack Scenario

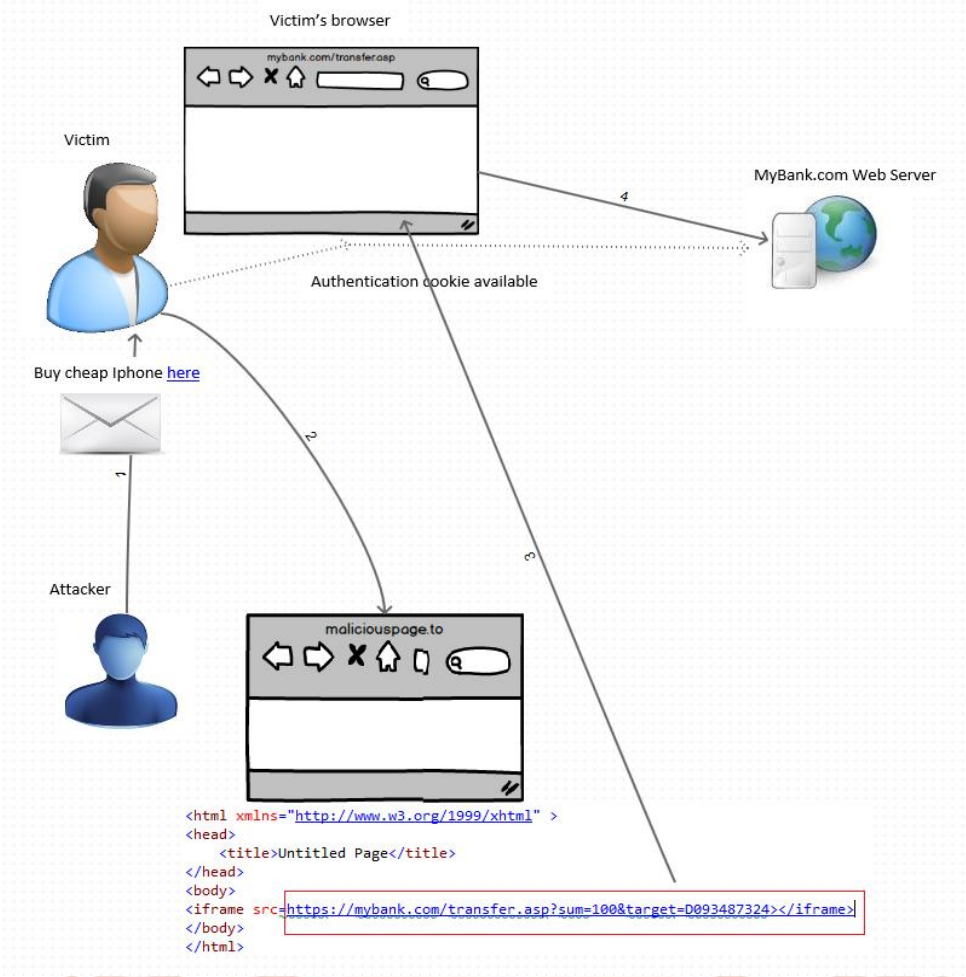

Figure: Scenario of request forgery attack

These attacks are divided into two parts: one is considered as a simple redirection attack and another is considered as a request forgery attack. In a simple redirection attack, somehow the attacker succeeds in adding a malicious URL reference in a given web page by using XSS. Here, the victim is automatically redirected to a vulnerable site unwillingly by referring to this malicious URL. This form of attack is also known as one-click attack. In request forgery attack, if the victim is logged into some genuine website then a session is created to continue transactions. Here, if the victim opens a new tab in the browser and tricks into the attacker's website, a script is automatically executed which fetches the victim's session id. Using this cookie/session id, the attacker performs transactions through legitimate websites as if he is an authorized user. This type of attack is also known as session riding attack. The given figure showcases an instance of a request forgery attack.

5.Recommendations for preventing XSSR and XSRF attacks

To mitigate the risk of redirection attacks, website developers can employ a data table or checklist that encompasses a comprehensive list of URLs supported by the website. This can be accompanied by a series of preliminary checks, such as: 1) Taking a web page as input and retrieving all the element tags associated with the given web page. 2) Verifying whether these element tags contain attributes like 'href' or 'src'. If such attributes are not available then continue with web page processing. If 'href' or 'src' attribute is identified then fetch attribute value for and perform further examination. 3) Compare attribute values with data entries of the data table. If a value match is found, the web page processing can proceed. 4) If its attribute value does not match with the data table's entries then notify the status as a vulnerable request and the web link request is prohibited.

To protect against request forgery attacks, web developers must establish secure sessions. Furthermore, they should verify specific information for every web page request. It is essential to frequently change the session id as a precautionary measure. To enhance security, consider implementing the following measures: 1) The user's identification, browser information, and IP address should be stored as session variables. Encrypting these values will enhance the security of the user's session. 2) In order to keep track of page loads effectively, it is necessary to utilize the pageLoadCnt variable as a counter. This variable should be initialized to zero for each page, and its value should be incremented by one only when a registered user requests a web page that is restricted to them. 3) The value of the pageLoadCnt variable should be checked for each page request made exclusively by registered users. If the value of this variable surpasses 3, it becomes necessary to regenerate the current session id and reset the counter variable back to zero. 4) Fetch the browser details and IP address of the web user for each page that is being requested and save them in temporary variables. 5) Evaluate the temp variables against the session variables. If details are matched, set the isAttackFound variable to false. Otherwise set the isAttackFound variable to true. If isAttackFound is true, treat the web page request as coming from an unauthorized user and destroying the current session values. If isAttackFound is false, consider the web page request as coming from an authorized user and grant access to the web page.

6.Conclusion

Websites are designed to foster global connectivity, offer information and services, simplify user tasks, and reduce transaction time and human effort. The World Wide Web (WWW) serves as a crucial tool for establishing communication networks across the world. It is essential that individuals do not abuse this platform for personal gain or malicious purposes. Cross-Site Scripting is the simplest way for an attacker to gain a user's confidential information. This paper aims to provide recommendations for the identification and prevention of redirection attacks by closely monitoring URL requests and validating user sessions. By ensuring the legitimacy of URL requests and session values, individuals can be protected from unauthorized activities and potential security threats while using the internet. Furthermore, a comprehensive defense strategy against XSS attacks includes server-side, client-side, and proxy-based techniques, along with continuous user awareness campaigns.

7.References

[1] Shalini, S., & Usha, S. (July, 2011). Prevention Of Cross-Site Scripting Attacks (XSS) On Web Applications In The Client Side. IJCSI International Journal of Computer Science, Vol. 8(4), pp. 650-654.

[2] Kombade, R. D., & Meshram, D. B. (February 2012). CSRF Vulnerabilities and Defensive Techniques. I. J. Computer Network and Information Security, 1, pp. 31-37.

[3] A. Khade, J. Iyer, M. Inbarajan and V. Yadav, Mitigating Cross-Site Request Forgery Threats in the Web. (2023) 7th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2023, pp. 695-698, doi: 10.1109/ICOEI56765.2023.10125633.

[4] Chrisando, Ryan, Pardomuan., Aditya, Kurniawan., Mohamad, Yusof, Darus., Muhammad, Azizi, Mohd, Ariffin., Yohan, Muliono. (2023). Server-side Cross-site Scripting Detection Powered by HTML Semantic Parsing Inspired by XSS Auditor. Pertanika journal of science and technology, doi: 10.47836/pjst.31.3.14, pp. 1353-1377.

[5] Jiazhong, Lu., Zhitan, Wei., Zhi, Qin., Yan, Chang., Shibin, Zhang. (2022). Resolving Cross-Site Scripting Attacks through Fusion Verification and Machine Learning. Mathematics, doi: 10.3390/math10203787.

[6] Bahruz Jabiyev, Omid Mirzaei, Amin Kharraz, and Engin Kirda. (2021). Preventing server-side request forgery attacks. In Proceedings of the 36th Annual ACM Symposium on Applied Computing (SAC '21). Association for Computing Machinery, New York, pp. 1626–1635. https://doi.org/ 10.1145/3412841.3442036

[7] W. H. Rankothge and S. M. N. Randeniya, Identification and Mitigation Tool For Cross-Site Request Forgery (CSRF). (2020). IEEE 8th R10 Humanitarian Technology Conference (R10-HTC), Kuching, Malaysia, 2020, pp. 1-5, doi: 10.1109/R10-HTC49770.2020.9357029.

[8] Ms. Diksha P. Meshram, Ms. Nisha Balani. (2019). Cross Site Request Forgery Prevention System. International Journal of Future Generation Communication and Networking Vol. 13, No. 2s, (2020), pp. 1169–1173.

[9] Roshan, Shaikh. (2019). Defending Cross-Site Request Forgery (CSRF) Attacks on Web Applications. ETD Collection for Pace University. AAI13904278.

[10] Kavitha, D., & Ravikumar, S. (2015, June). Enhanced Vulnerability Analysis For Clickjacking Web Attack And Providing Security Using Whitelisting URL Analyzer. International Journal of Engineering and Computer Science (IJECS), 4(6), pp. 12652-12657.

Sentamilselvan, K., Lakshmana Pandian, S., & Ramkumar, N. (November 2014). Cross Site Request Forgery: Preventive Measures. International Journal of Computer Applications, 106(11), pp. 20-25.