International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 08 | Aug 2020 www.irjet.net p-ISSN: 2395-0072

A Review on Cross-Site Request Forgery and its Defense Mechanism

Puneet Kour[1]

[1]*Student, Dept. of Computer Science and IT, Jammu University, Jammu and Kashmir, India*

Abstract - *Cross-Site Request Forgery is also one of the topmost vulnerability that is exploited by the hackers to carry out cyber attacks. In Cross-site request forgery (CSRF) attack the attacker send a malicious link to a legitimate user of a website. When the user click that malicious link, a forged request is sent on the behalf of the user. This attack on the websites have been increased every year and many firms become victims of it. Many solutions have been proposed to overcome CSRF attacks such as the HTTP header, Random Token validation, Client Site Proxy, but hackers are exploiting CSRF attack is still going on. In this paper, the author carried out a study on the CSRF attack and its defense mechanism. This paper also throws light on the merits and the demerit of the existing defense mechanism.*

***Key Words*: CSRF Attack, CSRF Tester, CSRF Prevention, Vulnerabilities.**

1. INTRODUCTION

Businesses greatly rely on data, information sharing over the Internet is everyday business. So, there are securing the sensitive data from misuse due to unauthorized access has become important. The attackers always keep on looking for flaws, vulnerabilities, and bugs in the web applications, browsers, servers, and the other components of the web to carry out malicious activities. Website Security Statistic Report 2015 by WhiteHat [1] shows that 64% of the public administration sites, 55% of the food and transportation sites are always vulnerable. 86% of web applications had authentication issues, access control, and confidentiality as per HP 2015 Cyber Risk Report [2]. According to the 2016 Vulnerability Statistics Report by Edgescan [3], 61% of the web application vulnerabilities lead to browser attacks. Internet Security Threat Report 2017 by Symantec [4] concludes that approximately 90-95 new vulnerabilities are discovered every week.

One of the best-known Web Application vulnerabilities is Cross-Site Request Forgery (CSRF). According to the Open Web Application Security Project (OWASP) [5], cross-site request forgery is listed as one of the top 10 web application vulnerabilities leading to a security breach in a past few years. The top most web vulnerabilities as per OWSAP are shown in Table 1.

Table -1: Top 10 web vulnerability as per OWASP

S. No.	OWSAP TOP 10 VULNERABILITIES
1.	Injection
2.	Broken Authentication and Session Management
3.	Cross-Site Scripting (XSS)
4.	Insecure Direct Object References
5.	Securing Misconfiguration
6.	Sensitive Data Exposure
7.	Missing Function Level Access Control
8.	Cross-Site Request Forgery (CSRF)
9.	Using Known Vulnerable Components
10.	Unvalidated Redirects and Forwards

The Cross-Site Request Forgery (CSRF) attack was first introduced by Peter Watkins in June 2001 posting to the Bugtraq mailing list [6]. CSRF attack also known as XSRF, Sea Surf, Confused Deputy, One-click Attack, or Session Riding [7]. CSRF flaws exploit cookies, browser authentication, or client-side certificates to authenticate attackers. Attacker tricks the web browser into un-notically executing an unwanted instruction in an application to which a user is logged in.

In a successful CSRF attack against the authenticated user, an attacker initiates arbitrary HTTP request using the user credentials to the vulnerable web application and compromises the end-user data that effectively bypasses the underlying authentication mechanism operation. If the victim account has administrator rights, this attack can compromise the entire web application [8]. The Fig.1 shows the processing of CSRF attack:

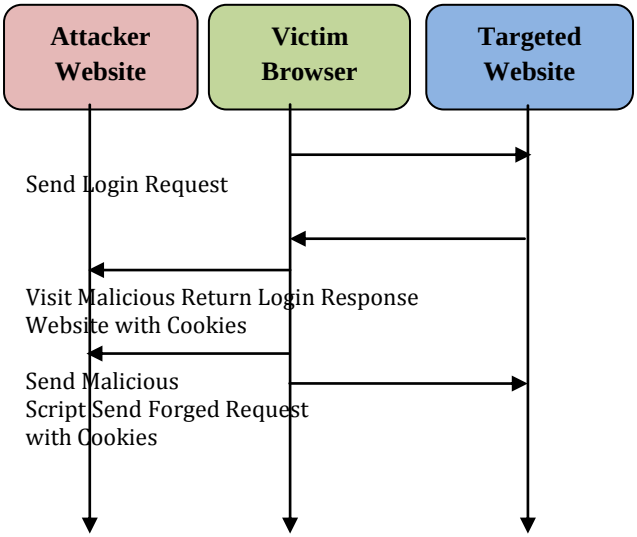

Fig -1: Processing of CSRF attack

2. RELATED WORK

The goal of the CSRF attack is by intrusion, to execute unwanted instructions on a web application in which victim is currently authenticated. Various defense mechanisms has been proposed by the researchers research some of them are listed as:

Wasim Akram Shaik, Rajesh Pasupuleti [9] proposed a novel approach for avoiding CSRF attack using TowFish security, by which the user can easily recognize whether the website is vulnerable to the attack. The TwoFish security approach work in two phases, in the first phase MD5 encryption is used to calculate a hash value for URL and in second phase image-based authentication is used to validate the image of the respective website.

Emil Semastin, Sami Azam et.al. [10] proposed a solution that integrates the passing an unpredictable token through a hidden layer and validates the hidden layer on the server-side with the passing token through URL which can offer double-layer protection against Cross-Site Request Forgery attack.

Jaya Gupta, Suneeta Gola [11] presented server-side protection again CSRF attack and implement a hybrid approach which is named as CSRF gateway in which token is generated when the HTTP request is generated by the user on the server-side, it creates a session and embeds token to the session using custom tag library which provides the more secured way to insert token and application detects the CSRF attack has been occurred or not and sends the user back to the login page.

Ramarao R. et.al. [12] proposed a client-side proxy solution that detects and prevents CSRF attack using HTML elements which are used to access the graphic images of the webpage. The client-side proxy is also able to examine and alter client requests.

Nikunj. J. Tandel, Kalpesh M. Patel [13] presented a review on CSRF attack, and its existing CSRF defensive mechanisms like secret validation token, referer headers, custom HTTP headers, and origin headers and culminate to build a robust CSRF defensive mechanism.

Boyan Chen, Pavol Zavarsky et.al. [14] examined the open source CSRF defense mechanism named CSRF Guard and analyzing some scenarios where CSRF Guard should be implemented and how it able to block CSRF vulnerabilities, explores limitations of CSRF Guard and some approaches to bypass The CSRF Guard.

Adam Barth, Collin Jackson, Mitchell Stanford [15] study CSRF vulnerabilities contain login CSRF vulnerabilities and provided three major CSRF defense techniques in which the HTTP Referer header could provide an effective defense solution in their experiment.

Sheeghrata Agnihotri, Pawan Patidar [16] proposed a client-server mutual authentication technique in which identification and authentication steps are separated. An authentication token is provided to each user and token is provided to the user in the form of an image which is encoded and decode using base64encoding and decoding technique.

In the above section, many author present different mechanisms to prevent this type of CSRF attack. Now, we are going to present a review of the CSRF attack and its defense mechanism.

3. CATOGRIES OF CSRF

As we know CSRF is one of the powerful vulnerabilities in web applications in which an attacker performs an unwanted action to the victim browser without the knowledge of the user. This vulnerability is categories as following:

3.1 Stored CSRF

A stored CSRF is one where the attacker can use the application itself and provide an exploit link, or the other content which directs the victim's browser to perform attacker-controlled action in the web application. Stored CSRF vulnerabilities are more likely to succeed since the user who receives the exploit content or link is almost certainly currently authenticated to perform action [10]. Any application that uses HTML tags is vulnerable to CSRF attacks. An attacker stores the malicious code using LINK, IMG, IFRAME, or SCRIP tags. Examples of such are shown below:

- <link> tag

 CLICK HERE TO GET REWARD

- <img> tag

- <iframe> tag
  <iframe scr = http://127.0.0.1/xyz.php height =100% width =100%>
  </iframe>
- <form> tag
  <form action = http://127.0.0.1/xyz.php method = "POST">
  </form>
- <script> tag
  <script>
  document.location = http://127.0.0.1/xyz.php/?password_new=123abc&password_conf=123abc&Change=Change#;
  </script>

3.2 Reflected CSRF

In this type of Reflected CSRF, the attacker uses a system outside the application to bare the victim to exploit links. This can be done using an email message, a blog, an instant message, or even an advertisement posted in a public with a URL that a victim type in. This attack will be a failure, as users may not be currently logged into the target system when the exploit is tried. The trial from a reflected CSRF attack may be under the control of the attacker and could be deleted once the exploit was completed. Here the page can be submitted automatically. Example of such, are shown below:

- Auto Post Forms
  <body onload = "document.form[0].submit()">
  <form method = "POST" action= "http://127.0.0.1/xyz.php">
  <input type = "hidden" name = "xyz" value = "987ads"/>
  </form>

3.3 Login CSRF

In a login CSRF attack, personal data of victims is collected. The main thing is this type of attack both the user and the attacker are authorized, users. But the attacker will send his identity to the user through some link. Thus users will now enter using the attacker's identity. Then all the information related to the user will get stored in the user browser, but actually in the attacker identity. Now, the attacker will see this information on the user. Therefore, they input personal data to the webserver such as credit card numbers or search keywords. Moreover, by using a Gadget, adversaries can execute malicious gadgets registered to the adversaries' accounts within the victim's local machine. Both CSRF and Login CSRF are ingenious exploitations based on the common belief that every web request made by a browser is initiated under users' supervision. However, in the real world, there are many ways for malicious adversaries to instigate page requests from the victim's browser [8].

- <form action= https://www.xyz.com/ method=POST target=invisibleframe> <input name=username value=attacker> <input name=password value=xyz>
  </form>
  <script>
  document.forms [0].submit( )
  </submit>

4. EXISTING DEFENSE MECHANISMS

Some defense policies are to b adopted by users and developers to get rid of such type of attack to some extent. The following defense mechanisms are:

- Use Logout Policy- If users want to move to some other site (any website or it may be the untrusted website), logoff web application which is not in use. It will automatically logout from currently using the site. So that the user wants to login again.
- Use POST form rather than GET- In GET forms values and variables in URL are visible to anybody, instead of GET use POST forms for secure submission.
- Use Random Token value in the code- Passing pseudo-random token through the hidden field is considered as the most effective solution to prevent against CSRF attack. At the server-side, the token is matched with the token in the session. If both tokens are matched, then it will be understood that request is generated from the same origin.
- Use Random Token value in the URL- Passing the unpredictable random token through URL is the next effective solution to prevent against CSRF attack.
- Use Random Token value in the cookies- We know every time cookies are generated when the user is logged in to the website. This technique is simple to implement when a user login to the website, the site should generate a random token value and set it as a cookie on both the user's machine and at the server-side. With each transaction request by the user the token value is matched at the server-side, the server accepts it as a legitimate request or it would reject the request.
- Use Referer Header- An HTTP request's Header Referer indicates the URL of the webpage which contained the HTML link or form that was responsible for the request's creation. The Referer is communicated through the HTTP header. If the Referer Header present, it distinguishes the original request from the cross-site request because it contains the URL of the Request. A site can protect

against cross-site request forgery attacks by checking whether the request was issued by the site itself or not. If this is not the matching case, the request is usually rejected [17].

- Use Custom HTTP Headers- Custom HTTP headers can be used to prevent CSRF attack because the browser feature prevents sites from sending custom HTTP headers to another site, but allows other sites to send custom HTTP headers to themselves using AJAX (XMLHttpRequest) [17]. For example, the header.js JavaScript library uses this approach and attaches the X-Requested By header with the XMLHttpRequest value. Google Web Toolkit also endorses that web developers protect against CSRF attacks by attaching an X-XSRF-Cookie header to XMLHttpRequest that contains a cookie value.
  To use custom headers as a CSRF defense, a site must issue all state-updating requests using XMLHttpRequest, attach the custom header, and reject all state-updating requests that are not attached with the header. For example, to defend against login CSRF, the site must send the user's authentication credentials to the server via XMLHttpRequest [18].

- Use Origin header- If the origin header is present, browsers send Origin header with POST requests that check the origin that initiated the request. If the browser is unable to specify the origin, the browse sends the null value.

Table -2: Merits and the demerits of existing defense mechanism

Existing Solution	Merits	Demerits
Use Post Forms	Secure submission of form values	Do not remain in the browser history
Client-server proxy	Easy to find attack	It would not detect login CSRF
Captcha	Easy to use for auto submission of HTNL forms	Expensive and required more memory
Random Validation Token	Extremely simple to implement	Random tokens would not detect Login CSRF
Use Referer Header	Easy to implement	Not all browsers support this header
Custom HTTP Header	Use to find same site request and CSRF	Many browsers doesn't support custom header
Origin Header	Another way to find same site request	Many browsers disable to suppoet header

5. DISCUSSION

In the end, we conclude that there is still a need for some relevant solution to get scour of this type of CSRF attack because every year hackers find out another approach to revivify the attack successfully. Many authors study to overcome the CSRF attack, also the above word can be extended to provide better solutions for CSRF attack by means of parsing techniques to identify the attacking spots before the attacker attack.

6. CONCLUSIONS

We know the need for the internet is increasing day-by-day but at the same time, the attacks also increasing. Users should be aware of such vulnerabilities on the internet. Cross-Site Request Forgery is one such vulnerability and these attacks can be easily executed by the attackers by simply executing a malicious script on the victim browser. Existing solutions and defense mechanisms to mitigate CSRF attacks do not provide complete protection. The above work can be extended to provide some better solutions against

CSRF attack or may combine two defense mechanisms in the future.

REFERENCES

[1] WhiteHat Security, Inc., Website Security Statistics Report (2015), Santa Clara, CA 95054, 2015. https://www.prnewswire.com/news-releahttps://www.prnewswire.com/news-releases/whitehat-security-2015-website-security-statistics-report-reveals-the-need-to-identify-security-metrics-most-important-for-vulnerability-remediation-300086119.html

[2] HP Security Research, "Cyber Risk Report", 2015. https://www.govtech.com/library/papers/HP-Security-Research-Cyber-Risk-Report-2015-1480.html

[3] BCC Risk Advisory Ltd., Vulnerability Statistics Report Edgescan, 2016, http://www.edgescan.com

[4] Symantec Corporation, Internet Security Threat Report, 2017. http://www.symantec.com/threatreport/

[5] OWASP (Open web application security project) top ten project, http://www.owasp.org/index.php/ Category: OWASP-Top_Ten_Project; 2013 http://www.owasp.org/index.php/CrossSiteRequestForgery

[6] P.W. Cross Site Request Forgeries, 2001. www.securityfocus.com

[7] Kafer K. (2008). Cross Site Request Forgery. Technical report, Hasso-Plattner-Institute.

[8] Sentamilselvan K, Dr. S. Lakshamana Pandian, Dr. K. Sathiyamurthy, Survey on Cross Site Request Forgery, International Conference on Research and Development Prospects on Research on Engineering and Technology, pp. 159-164, 2013.

[9] Wasim Akram Shaik, Rajesh Pasupuleti, Avoiding Cross Site Request Forgery (CSRF) Attack Using TwoFish Security Approach, International Journal of Computer Trends and Technology, pp. 68-72, 2015.

[10] E. Semastin, S. Azam, B. Shanmugan, K. Kannoorpatti, M. Jonokman, G. N. Samy, S. Perumal, "Prevention Measures for Cross Site Request Forgery Attacks on Web-based Applications", Indian Journal of Engineering and Technology, pp. 130-134, 2018.

[11] J. Gupta, S. Gola, "Server Side Protection againstCross Site Request Forgery using CSRF Gateway:, Journal Information Technology & Software Engineering, 2016.

[12] Ramarao R. Tool , "Preventing Image Based CSRF attacks 2009.

[13] N. J. Tandel, Prof. K. M. Patel, "Defensive Mechanisms of CSRF Attack", International Journal of Engineering Development and Research, pp. 734-737, 2014.

[14] B. Chen, P. Zavarsky, R. Ruhl and D. Lindskog, " A Study of the Effectiveness of CSRF Guard", IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International Conference on Social Computing, pp. 1269-1272, 2011.

[15] A. Barth, C. Jackson, M. Stanford, "Robust Defenses for Cross-Site Request Forgery", Association for Computing Machinery ACM, 2008.

[16] S. Adnihotri, P. Patidar, "Prevention against CSRF Attack using Client Server Mutual Authentication Technique", International Journal of Engineering Science and Computing, pp. 20393-20398, 2009.

[17] X. Lin, P. Zavarsky, R. Ruhl, D. Lindskog, "Threat Modeling for CSRF Attacks", IEEE International Conference Computational Science and Engineering, 2009.

[18] N. J. Tandel, K. M. Patel, "Defensive Mechanisms of CSRF Attack", International Journal of Engineering Development and Research, pp. 734-737, 2014.

[19] K. R. Suneetha, R. Krishnamoorthi, "Identifying User Behavior by Analyzing Web Server Access Log File", International Journal of Computer Science and Network Security(UCSNS), Vol. 9, Issue 4, 2009.