

CSRF and XSS Attacks and Defense Mechanisms

Udit Kalra
B.Tech Scholar
Department of IT
MAIT (GGSIPU), Delhi

Abstract:- Ransacking for just the perfect article is that the most preferred and is sort of not easy to search out supported the present requirements. As technology is developing day by day, hacking is additionally occurring very often. In these recent times, the sector of cybersecurity is in dire need of prevention from this. Gone are the times when firewalls were able to protect your data. According to PT Security, each system contains 22 vulnerabilities out of which four are of high risk. We need to try to do this ourselves to stop cybercrime. According to Kaspersky Labs, the typical cost of a cyber-breach is \$1.23 million. This paper is on the brink to give the simplest possible ways to assist and help make secure websites. Security of Web application has become a vital challenge because of common vulnerabilities found during a web application nowadays. Web Security is a crucial step to induce through a number of your problems for an answer. Once you know that your website is safe, you will be less accentuated. There are lots of attacks accustomed hack a web site like CSRF, XSS, Command Execution, Brute Force and more. I have thoroughly researched the most general vulnerabilities and created a live environment to attack similarly to defend using the newest software. During this paper, I have discussed two such vulnerabilities. They are Cross-Site Request Forgery (CSRF) and Cross-Site Scripting (XSS) and their prevention.

Keywords:- Web Security, Cyber Security, CSRF, XSS, Application Security.

I. INTRODUCTION

In today's world of digitalization, web applications, which generally act as public-facing entities for several businesses and corporations, are often the victim of malicious attacks by hackers who wish to steal customer data or whirl their way farther into a corporation's private network. There are some web applications available which are design to be intentionally vulnerable for training purposes. What I think is that web applications must be developed by highly skilled developers who knows the importance of providing security and knows how to handle these vulnerabilities. Several companies understand the use of the word security in web applications so they use these type of developers and have trained individuals who knows about cyber security. These individuals work to stay an account on all the kind of vulnerabilities that exist and to work the way to overcome if any new threat comes.

A small change in code or a little error can cause enormous damage. Therefore it must be handled carefully to allow the best possible results. Many researchers try to search out a praiseworthy solution to unravel these problems.

This papers main objective is to assist everyone who is making a brand new website, learning about cybersecurity or anyone using some online environment in day to day life be safe from these pentesters. This paper has been divide into many sections. Previous one was the abstract, Section I is that the introduction of the subject. Section –II is about Related Work which contains the information about the topic and some work related to this paper. Section-III is about the methodology of how attack is performed. Section-IV is about the implementation and prevention of the attack. This is the most important part of this research paper. Section-V is all that says the paper review and conclusion on my research. After that are some References to some other research papers.

II. RELATED WORK

Cross-Site Request Forgery (CSRF) is an attack which compels users to perform unwanted actions on the sites on which they logged in currently. [1] Through social engineering, the attacker sends the user some certain links that are specially crafted for that user, using which an attacker may trick the users of a web application into executing actions of the attacker choosing. [2] In a successful CSRF attack, the user unknowingly can do a lot of damage such as transferring money, changing passwords, provide sensitive data. If performed on an administrative account, it can give the attacker access to the whole network and cause widespread damage. Csrft attack exploits the property of the web browser of automatically including the cookies used by a given domain to any web request. In any event of user unknowingly submitting a request to the browser, which automatically collects the cookies of the site the user is logged in and as a result, it creates a facade that the forged request becomes true. Hence, the attacker now can manipulate that request to perform any action such as returning data, modifying data and more.

Cross-Site Scripting (XSS) is a Code injection attack executed on the client-side of a Web App. Here, the attacker injects some malicious code (script) through your web browser. Now, whenever you visit that web server, the malicious script is executed. It can harm you by stealing cookies, session tokens, and much more sensitive data. It can modify the contents of the website. XSS attacks are

basically of three types- Reflected XSS (Non-persistent), Stored XSS (Persistent) and DOM(Document Object Model) XSS.

In Reflected [8] XSS attack script is executed on the victim site and not stored on the server. In Stored XSS attack script is executed and stored on the server. Whenever the malicious site is requested Stored XSS gets executed. Though, XSS DOM is a client-side server attack. Here the script is not sent to the server.

➤ Popular CSRF Vulnerabilities

1. ING Direct [11] (ingdirect.com)

A vulnerability on ING's website that allowed additional accounts to be created on behalf of an arbitrary user. Some of the people were ready to transfer funds out of users' bank accounts. This was the primary CSRF vulnerability to permit the transfer of funds from an institution.

2. YouTube [12] (youtube.com)

CSRF vulnerabilities were as discovered in nearly every action a user could perform on YouTube. The attacker using the csrf vulnerability could easily make changes on the users account such as making comments on a video, flagging a video, adding videos to favorites, collecting contacts information from the user's account.

3. MetaFilter (metafilter.com)

A vulnerability existed on MetaFilter that allowed an attacker to require control of a user's account. A forged request could be wont to set a user's email address to the attacker's address. A second forged request could then be accustomed activate the "Forgot Password" action, which might send the user's password to the attacker's email address.

4. Play Framework [13]

A vulnerability within the Play framework can allow an entire cross-site request forgery (CSRF) protection bypass, researchers have warned. The play could be a framework for building web applications with Java and Scala. It is utilized by companies including LinkedIn, Verizon, and Walmart. The open-source framework allows users to line up a restricted set of content types it'll allow as a part of its anti-CSRF mechanism. However, researchers discovered they were able to bypass this optional functionality by sending malformed Content-Type headers to a target web app. It was found that an attacker could use a semicolon within the boundary value which doesn't fit RFC 2046, therefore circumventing the framework's blocklist function.

5. Gmail (www.gmail.com)

A vulnerability in GMail was discovered in January 2007 which allowed an attacker to steal a Gmail user's contact list. A distinct issue was discovered in Netflix which allowed an attacker to alter the name and address on the account, additionally as add movies to the rental queue etc.

➤ Popular XSS Vulnerabilities

1.) Yahoo, which was within the limelight for revealing an enormous program its users earlier this year, has fixed a highly critical cross-site scripting (XSS) security flaw in its email system that may have allowed attackers to access any email. This flaw was later discovered and reported by the Finland-based security researcher JoukoPynnonen who also got to earn \$10,000 for the work. This flaw allowed an attacker to read the victims email, creating a pandemic affecting Yahoo Mail and many other things. Unlike other email phishing scams and ransomware attacks, there is no need for the hacker to send a plague or trick the victim into clicking a specific link. The attacker would just have to send a mail to the victims account to access their emails.

2.) In 2010, a cross-site scripting (XSS) vulnerability had been identified on an American Express website secured with EV SSL and might be exploited to reinforce phishing attacks.

XSS weaknesses are the results of poor input validation into Web forms and permit attackers to return potentially malicious code to visitors' browsers.

Ensuring proper validation of all inputs in Web applications, to forestall cross-site scripting and SQL injection vulnerabilities is a requirement of the Payment Card Industry Data Security Standard (PCI-DSS).

III. METHODOLOGY

➤ [3]The csrf attack is performed as follows:

Presume a user is active on an authentic target site A through his browser. While traversing through his site the user comes upon a link provided to him by an attacker through social engineering (via email, chat etc). The user immediately clicks on the given link, but it is critical for the profitable execution of the attack that the user has the target site Active on another tab.

➤ The link will now carry the user to the malicious site.

Now here the malicious site is specially crafted by the attacker to accomplish the specific function he wants the user to do.

[4]Crafting the site requires thorough knowledge of the forms and specifics of the target site that the attacker wants the access from the user.

This site contains a script which can perform an invalid function on the site A using the sessions of the user because he is currently active on both the sites. However, the important part is to dupe the user into clicking the link through social engineering.

[5]Let's take a scenario where the user is active on site A and the attacker wants the user to change his password from a malicious site B.

To achieve this the attacker first needs to get his hands on the form of site A which changes the password of site A and create a form of the site B which tricks users on clicking

the link and thus changing the password of site A without the knowledge of the user.

[6]The form of site looks like this:

```
<form action="#" method="POST">
<input type="text" name="newpassword" value="">
<input type="text" name="confirmpassword" placeholder =
"newpassword" value="">
<button>Change</button>
</form>
</form>
```

Notice the action is the address of the page which the site A takes the user after when he changes the password. If the attacker manages to put that address and send the user a link like this :

```
<form      action="https://address_of_changed_password"
method="POST">
<br> Congratulations You have won a cash prize of
$100000/- click to avail!!!!</br>
<input type="hidden" name="newpassword" value="xyz">
<input      type="hidden"      name="confirmpassword"
placeholder = "newpassword" value="xyz">
<button>Change</button>
</form>
```

If the site manages to change the password then it is vulnerable to the csrf attack.

IV. IMPLEMENTATION/PREVENTION

For a flawlessly executed CSRF attack, the attacker should have a thorough knowledge of the varieties of the methodology used by the site. As a web developer you can prevent the execution of this attack by using the following methods:

➤ *Token-Based Authentication*

The anti [7] [8] csrf tokens are widely used technology which is highly recommended and is known to be very effective against this attack.

By using different hash functionalists the anti csrf code that you embedded in your page creates a token of certain fixed length and which always has a different value. Now, these tokens work on the principle that each page randomly generates only one token-id at a time and cannot accept two pages to exist with the same token -id. That is if you refresh the page a new token will be generated and the previous token value will be dropped, making it certain that at one instant only one page with that token value exists on the internet. Now, when the attacker would try to implement a phishing link on your site (duplicating the webpage form) he/she will automatically copy the generated token number with it. Thus creating a clash on the server which results in an error suggesting invalid token number because a page of that token-id value already is in existence.

➤ *Synchronizer token-based:*

[9]They are created on a request basis, these are server-based tokens that are better than session-based tokens as they furnish a better degree of security. Frequently session-based tokens are susceptible to browser back refresh attacks and synchronizer request based tokens prevent such attacks. On request, the server checks the individualism of the csrf tokens and upon the validation, with the user sessions tokens, the requests are conducted if the tokens are deemed not distinct or legal the requests are not passed.

➤ *Encryption Based:*

It utilizes [9] encryption rather than token based comparison. The server uses a unique key to encrypt tokens comprising session-id and Timestamp of users, and upon requesting the server to send the tokens to the user where these tokens are decrypted and if the decrypted tokens don't match the values of tokens then they are considered too meddlesome and rescinded.

➤ *Same Site Cookie Attribute:*

The same [9] site cookie attribute studies were whether or to not transmit cookies to another site. It assists the browser to choose where to send the cross-site requests together with the cookies. It always checks before sending cookies even on regular links. Now, for instance, a GitHub-like website, this may mean that if a logged-in user pursues a link to a personal GitHub project posted on a company discussion forum or email, GitHub won't receive the cookie and therefore the user won't be able to access the project.

➤ *User-based Authentication:*

Sometimes, simple user-based interaction also acts as a powerful tool against CSRF. User interaction such as:

- 1.)CAPTCHA
- 2.)OTP
- 3.)Re-Authentication

However, a powerful line of defense these mitigations turn out to, they are not supposed to just implement as the only line of defense against the attack. They should always be used as an extra measure of security.

➤ *Login Forms:*

Developers [9] frequently speculate that login forms are secure enough and need not be a spur to worry about the csrf attack, but on the contrary login, forms are also equally at risk to this attack. An attacker can effortlessly copy forms and bait users to log in again retrieving passwords and other sensitive information. Login forms can be prevented using pre-sessions and adding csrf tokens.

➤ *Don't use method override:*

Several applications are presently using [10] method-override functions to use PUT, PATCH, and DELETE requests for the usage of forms. This as a result the requests which weren't vulnerable before now vulnerable hence could cause vast damage.

For a flawlessly executed XSS attack, as a web developer you can prevent the execution of this attack by using the following methods:

- 1 - Never enter any data which cannot be trusted. Like, in a script tag
`<script>Never enter data cannot be trusted</script>`
- 2- Encode the HTML data before inserting any untrusted data in the HTML Element Content
 eg. `<body>Encode data before putting</body>`
`<div>Encode data before putting</div>`
- 3- Encode the JavaScript data before inserting any untrusted Data in the JavaScript values.
- 4- You can sanitize the code by using the express-sanitizer package to avoid the usage of the script tag.
- 5 - Avoid using untrusted JavaScript URLs. Like, href tags or iframe tags.

In the end, make sure that you validate all the untrusted URLs so that they contain only safe schemes like HTTPS.

V. CONCLUSION

The csrf and xss attacks are not to be ignored. These attacks, seems simple but can cause a prolific amount of damage to your systems, resulting in data breaches, frauds etc. They prevail today because most developers are not concerned with the security of the web application. Another reason for this attack is the lack of knowledge about cybercrimes among the users, due to which they are fall prey to social engineering attacks. Proper mitigation is unequivocally important for secure use of applications.

REFERENCES

- [1]. <https://www.netsparker.com/blog/websecurity/csrf-cross-site-request-forgery/>
- [2]. "Survey on Cross Site Request Forgery (An Overview of CSRF)" By Sentamilselvan K, Dr.S.Lakshamana Pandian
- [3]. https://en.wikipedia.org/wiki/Cross-site_request_forgery
- [4]. <https://d1wqtxts1xzle7.cloudfront.net/50582893/ICCSN>.
- [5]. Hackersploit (YouTube channel) <https://www.youtube.com/channel/UC0ZTPkdxlAKf-V33tqXwi3Q>
- [6]. OWASP CSRF [https://owasp.org/www-community/attacks/csrf#:~:text=Cross%20Site%20Request%20Forgery%20\(CSRF\)%20is%20an%20attack%20that,response%20to%20the%20forged%20request](https://owasp.org/www-community/attacks/csrf#:~:text=Cross%20Site%20Request%20Forgery%20(CSRF)%20is%20an%20attack%20that,response%20to%20the%20forged%20request)
- [7]. "Robust Defenses for Cross Site Request Forgery" Adam Barth, Collin Jackson, John C. Mitchell(Stanford University).
- [8]. "Assessment of vulnerabilities of web applications of Bangladesh: A case study of XSS and CSRF" By TanjilaFarah
- [9]. https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
- [10]. "A study of effectiveness of CSRF Guard" By Boyan Chen, PavolZavarsky, Ron Duhl and Dane Lindskog

- [11]. "Evaluation of Static Web Vulnerability Analysis Tools" By ShobhaTyagi, and KrishanKumar.
- [12]. <https://news.hitb.org/>
- [13]. <https://portswigger.net/>
- [14]. "Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks" By José Fonseca, Marco Vieira and Henrique Madeira
- [15]. "SWAP: Mitigating XSS Attacks using a Reverse Proxy" By Peter Wurzinger, Christian Platzer, Christian Ludl, EnginKirda , and Christopher Kruegel.
- [16]. "Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art" By Shashank Gupta, B. B. Gupta