ISSN XXXX XXXX © 2019 IJESC

Research Article Volume 9 Issue No.3

Prevention Against CSRF Attack using Client Server Mutual Authentication Technique

Sheeghrata Agnihotri[1], Pawan Patidar[2]
Department of Computer Science and Engineering
Laxmi Narayan Collage of Technology, Indore, India

Abstract:
The web has become an indispensable part of our lives. Unfortunately, as our dependency on the web increases, so does the interest of attackers in exploiting web applications and web-based information systems. Previous work in the field of web application security has mainly focused on the mitigation of Cross Site Scripting (XSS) and SQL injection attacks. In contrast, Cross Site Request Forgery (XSRF) attacks have not received much attention. In an XSRF attack, the trust of a web application in its authenticated users is exploited by letting the attacker make arbitrary HTTP requests on behalf of a victim user. The problem is that web applications typically act upon such requests without verifying that the performed actions are indeed intentional. Because XSRF is a relatively new security problem, it is largely unknown by web application developers. As a result, there exist many web applications that are vulnerable to XSRF. Unfortunately, existing mitigation approaches are time-consuming and error-prone, as they require manual effort to integrate defense techniques into existing systems. In this paper, client server mutual authentication technique has been proposed .This technique separates the identification and authentication steps. Authentication token is provided to each user which helps to prevent this attack. Tokens are provided to the user in the form of image which are encoded and decode using base64encoding and decoding technique. This encoding and decoding technique is used for improving security. We provide experimental results that demonstrate that client server mutual authentication technique provides better solution against the CSRF attack, which is done with the help of IFRAME. Attack made through POST or GET request, using Javascript and IMG teg of HTML are thwarted.

Keywords: CSRF, XSRF, Client Server Mutual Authentication, Token, Identification etc.

I. INTRODUCTION

Use of internet is increasing very rapidly with the fast changing technology. It is now being used for every possible functionality that can be performed online. Web applications are playing important role to provide these functionalities. Web applications have now become part of life of human beings. These applications help to reduce their efforts for activities such as reservations, online banking etc. Some are aimed at entertainment or connecting users socially such as Facebook, Myspace etc. With all these facilities and convenience, they have also brought some problems related to security. Attacks on web application may result in huge loss in term of loss in data reputation etc. Due to their popularity Web applications have become a major target for hackers. Web applications run in the browser. Web applications are accessed through a browser. Any security escape clause in programs may prompt exploiting vulnerability in a web application. Well known client side attack is CSRF (Cross Site Request Forgery) attack A report submitted by Open Web Application Security Project (OWASP) in the year 2013, on vulnerabilities in critical web applications ranks Cross Site Request Forgery (CSRF) attack at position seventh[10]. CSRF attack is known by various different names, including Session Riding, XSRF, confused deputy Sea Surf, Cross-Site Request Forgery, and Hostile Linking. Social engineering (such as sending a link via email or chat) helps, an attacker may trick the users of a web application by executing actions of the attackers choice[7] Attacker inherits the identity and privileges of the victim to perform an undesired function on the victims behalf. Many sites, browser request automatically include any credentials associated with the sites, such as the users session cookie, IP address, Port, Windows domain credentials, etc. Therefore, if the user is authenticated currently to the legitimate site, the site does not have any method to find difference between a forged request and a legitimate request sent by the victim. CSRF attacks target the functionality that causes a state change on the server, such as changing the victims email address, password, purchasing choice etc [10] [11]. HTTP is the most common stateless protocol used for accessing website. It is not able to determine whether all the requests belong to a single user or from different users. Thus there is no straightforward mechanism to identify requests of a user authenticated on a web server. One way to overcome this problem is to preserve user-specific state in client-side cookies [12]. CSRF is common attack for which few mitigation solutions have been proposed. The solution includes use of client site proxy solution, client Side Browser plug-in, Origin Header, server site proxy, NOScript and CsFire etc [5][6][7][3] [8][1]. These solutions do not provide the complete protection against CSRF or require significant modification individual web application be protected.

In this paper, a approach is presented that provides protection from CSRF attacks. A client server mutual authentication technique has been used. A shared secret between client and server is used to prevent this attack. The shared secret cannot be stolen by an attacker, and the browser cannot be lured into leaking the secret.

Rest of the paper is organized as follows: In the next section the background details are given. In III section related work has been discuss. In section VI proposed method is presented. Results and Discussion on them are given in section V. The paper is concluded in section VI along with some discussion on future work.

II. BACKGROUND

Cross-site request forgery attack is also called one-click or session riding and abbreviated as CSRF or XSRF. It is a type of attack in which exploitation of a website is done by issuing the unauthorized command from a user to the trusted website. In this chapter different types of CSRF attacks are discussed. Also the tools and technologies using of the project are presented.. different type of CSRF attack are discuss below.

A. Types of CSRF attacks

CSRF attacks can be classified into two major categories reflected and stored/local [9]. Reflected CSRF: In reflected CSRF helplessness, the assailant utilizes a frame work outside the application to open the casualty to the adventure connection or substance. This should be possible utilizing a blog, an email message, a text, a message board posting, or even an advertisement posted in an open spot with a URL that a casualty sorts in. Reflected CSRF assaults will frequently come up short, as clients may not be right now signed into the objective framework when the exploits are attempted. The trail from a reflected CSRF assault might be under the control of the attacker, however, and could be erased once the adventure was finished.

Stored CSRF: A put away CSRF defenselessness is one where the aggressor can utilize the application itself to give the casualty the adventure join or other substance which coordinates the victims browser over into the application, and causes assailant controlled activities to be executed as the casualty. If any web application is venerable to CSRF attack then the malicious code is stored by the attacker using IMG, IFRAME tag or javascript. When the CSRF attack is stored in the site then the possibility of this attack is high because the victim is more excited to view the page containing the attack then some random page on the Internet. This vulnerability are more likely to succeed, since the user who receives the exploit content is almost certainly authenticated to perform actions. Stored CSRF vulnerabilities also have a more obvious trail, which may lead back to the attacker.

B. CSRF Attack Vector

A web application is vulnerable against CSRF attacks since it believes the session, between the server side part of the web application and the customer, No approval in individual solicitations are made by the customer [13]. This empowers an aggressor to trap the accidental client in sending a vindictive request to the server, which is trusted by the server, since the customer is validated and trusted inside the session. Vindictive URL utilized as a part of a CSRF assault is frequently installed inside a ¡img¿ HTML tag on an pure looking page so that a web program will naturally play out a GET ask for to the URL without client assent. Whenever the CSRF attack is performed by the HTTP POST Request rather than HTTP GET request the attack is slightly more complex. From will be created by attacker using HTML element or JavaScript for performing this attack. Because of this attacker have some degree of control over the malicious site in which attacker will have to embed their own link in this site. In this attack attacker gain control over the site either by being the site owner or finding some XSS vulnerability in the site. A user is venerable as long as he is logged in to a web application. A single mouse click or just browsing a page under the attackers control can easily lead to unintended requests. Most web applications are not aware of this fact, leaving their users in danger. A typical CSRF attack is shown in figure 1.

Figure.1. CSRF Attack

Some work related to mitigating of CSRF is as discussed in next section.

III. RELATED WORK

In last few years, researchers have made contribution towards prevention of CSRF attack. CSRF vulnerabilities have been known and in some cases exploited [12]. Ramarao R, et al [5] presented a client-side proxy solution that recognizes and prevents CSRF attack utilizing IMG component or other HTML components which are used to get the realistic pictures for the website page. This intermediary can assess and alter clients demands and the applications replies (output) naturally and transparently expand applications with the secret token approval method William Zeller, et al [6] actualized a client-side browser plug-in that can shield users from specific sorts of CSRF assaults. They executed their tool as an extension to the fire fox web browser. Clients needed to download and introduce this expansion for it to be effective against CSRF assaults. Their augmentation works by capturing each HTTP request and deciding whether it ought to be permitted. This decision is made using the following rules to start with, any solicitation that is not a POST solicitation is permitted. Next, if the requested for site and target site fall under the same-source strategy, the solicitation is permitted. Next, the site requesting the permission to make a solicitation utilizing adobes cross-domain policy for the target site. Nanad jovanovic, et al [7] proposed a mitigation mechanism that is based on server side proxy that detect and prevent CSRF attack and it is transparent to both user and web application. It provide complete automatic protection From XSRF attack. It is the mitigation mechanism

that provide provide just partial security by replacing GET Request by POST Request or depending on the information in the Referrer header of HTTP solicitations. Johns, et al [3] proposed Request Rodeo for prevent CSRF attack. Apart from thisRequestRedo with the exception of client side SSL provide protection against the misuse of implicit authentication mechanism. It enable user to protect themselves against CSRF attack. It is same as the local proxy on users computer. Tatiana Alexenko, et al [8] developed mozilla firefox web browser extension to protect users browsing history. It generates the HTTP request to random URLs from the users browsing history. Before loading page it previews the HTML code and detect the potentialCSRF attack. The detector would first find all form tags and check the action attribute of the form tags for deep linking. When CSRF detector found such forms it prompts the user and ask the user if he want to add the pair of URL of website and URL of form action to white list A solution to prevent CSRF attacks, a web application has to make sure that the incoming form data has originated from a valid HTML form. Valid" in this context means the submitted HTML form was generated by the actual web application in the first place. It also has to be ensured that the HTML form was generated especially for the client. To enforce these requirements, hidden form elements with random values have been employed. These values are used as one time tokens[2]. CsFire is an integrated extension into Mozilla browser to mitigate CSRF attacks. CsFire is the only system that provides formal validation through bounded model checking to defend against CSRF in the formal model of the web developed by Akha we et.al [1]. CsFire strips cookies and HTTP authorization headers from a cross-origin request. The advantage of stripping cookies and HTTP authorization headers is that there are no side-effects for cross-origin requests. No Script ABE [2], or Application Boundary Enforcer, restricts an application within its origin, which effectively strips credentials from cross-origin requests, unless specified otherwise. The default ABE policy only prevents CSRF attacks from the internet to an intranet page. Request Policy [4] protects against CSRF by blocking all cross-origin requests. In contrast to stripping credentials, blocking a request can have a very noticeable effect on the user experience. When detecting a cross-origin redirect, Request Policy injects an intermediate page where the user can explicitly allow the redirect. It includes a predefined white list of hosts that are allowed to send cross-origin requests to each other. Users can add exceptions to the policy using a white list. Burns and Schreiber [16] have use a cryptographic tokens optional HTTP referrer header to verify Action Formulators, require changes to application state to be done only with HTTP POST operations and use a simplified CSRF Prevention Token. The approach to mitigate CSRF attack is presented in the next section.

IV. PROPOSED APPROACH

In this section, proposed client server mutual authentication technique is discussed to mitigate the CSRF attack. In this method Authentication and identification have been separated. Thus complete authentication consists of two steps:-

1) Identification through username and password.
2) Authentication through token.

The identification and authentication in web session relies on visual authentication tokens which can be easily remembered and recognize by the user. After login, the user is equipped with the shared secret that is not stored in his browser. The former universal token then serves as the identification that is complemented by the shared secret as the authentication for security critical operation. The shared secret cannot be stolen by an attacker and the browser cannot be lured into leaking the secret.

Figure.2. CSRF Prevention

When the user accesses a web application, for protection from CSRF attack, server follows the mechanisms as given below.

- When new user arrives, he is required to register to the website and then provide with the registration key of his choice.
- Next, the user logs into the web server.
- Next, the server provides an encoded token to the user which appears on the screen in decoded form when user enters the registration key provided to him during

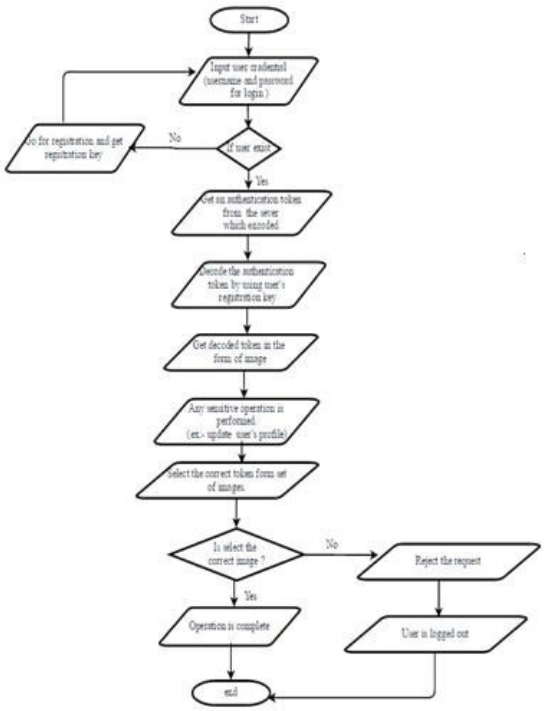

Figure.3. Flow Diagram

- registration process. the encoding and decoding is done by the base64 encoding and decoding technique.
- On receiving the token, the user continues to fill all the required information corresponding to particular operation and submits.
- During the interaction of client with server, the attacker can hijack the HTTP session and can insert or modify information related to user.
- Therefore, to ensure the authentic information exchange between client and server, the server asks the client to select the token that given to him at the time of login.
- The server facilitates the user by displaying multiple image token from which the user needs to select the token corresponding to him.
- On receiving the valid token, the server completes the required operation otherwise reject the request.

In the next section, the testing of proposed method is presented.

V. RESULTS AND DISCUSSION

In this section, the testing and result of the protection mechanism from CSRF attack discussed. Based on the method used by the client for communication with the server. The attacker uses either GET request/POST request for modifying the information. Next, the CSRF attack performed on the server. When the client uses POST request to communicate with the server is discussed.

A. Test Case 1: CSRF Attack Using POST Request

- The target website for this example will be local host/ demo/index.php.
- User has account on his website.
- The user must be authenticated with the target website. Once the victim is authenticated, the attacker can include a link or script in a third-party website that the victim visits.
- The attacker uses an HTTP POST request to realize a CSRF attack. The code when attacker use POST request is as shown in fig.4.
- It is very difficult for the target website to distinguish between legitimate and rogue HTTP POST requests, since the requests are sent from a trusted browser.
- Thus, when the victim visits that website or link, the rogue script will be executed without the victim being aware of it.
- That means that if no prevention measures are in place, a CSRF attack can be performed transparently without the victim or target website realizing it.
- By analysing packets the attacker uses CSRF to change the information on the victims profile.

```
<form name="badform" method="post"action="http://10.1.100.195/csrf3/updated.php
address  <input type="text" name="address" value= "bhimrajaji"/>
contact   <input type="text" name="contact" value="65843712"/>
pincode <input type="text" name="pincode" value="34589" /></td>
username<input type="text" name="username" value="hacker@gmail.com"/></td>
</form>
<script type="text/javascript">
      document.badform.submit();
   </script>
```

Figure.4. Code when attacker uses POST request

This attack is prevented by client server mutual authentication technique as given below-

- The user logs in the website as he gets a registration key show in Fig 5.
- User gets a no encoded Authentication token it is to be remembered by him. A sample token is shown in Fig 7.
- User performs some operation. for example updating profile shown in Fig 8.

Figure.5. User login

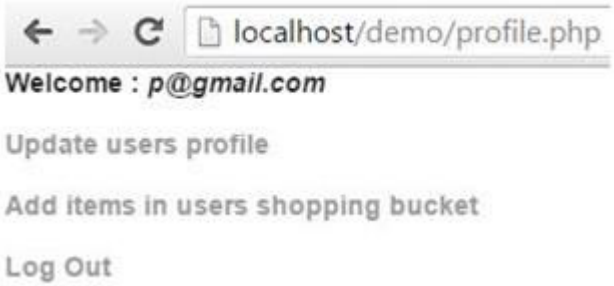

Figure.6. User's profile

- Before completing the chosen operation he chooses the correct token from the set of tokens as shown in fig 9.
- After choosing correct token operation is completed at the server if token is incorrect then user is logged-out from server without completion of operation.

Figure.7. Authentication token

Next, another test cases which the CSRF attack can be performed using GET request is discussed.

B. Test Case 2: CSRF Attack Using GET Request

- In the CSRF attack below, the data to be changed is contained in a parameter called Email Address.
- If the user can be tricked into visiting a website under the attackers control, the following code can be used to change the email address stored as a login credential on that site, The

page can be presented as anything: it could be blank, or it could be a replica of the website thats under attack.

Figure.8. Update From

Figure.9. Set of tokens

- All it needs is the code above, which displays an image; this image does not need to exist, and it only covers a 1x1 pixel area, so it does not arouse suspicion.
- As soon as the user's browser loads the page, the code will automatically submit the request to change the users email address.
- As long as the victim is logged into the website at the time, it will be processed exactly as if the victim had clicked the link.

Here we use the client server mutual authentication technique to prevent these types of attacks.

V. CONCLUSION

Cross Site Request Forgery is one of the common vulnerabilities in the Internet. It remains challenging for the researchers to provide a better solution for mitigating this attack. There were many organizations which affected by this cross site request forgery attack. Defense mechanisms and existing solutions for cross site request forgery are working in some extend only. The above work can be extended to provide suitable solutions for the cross site request forgery attack by means of applying techniques to preventing the attack before the attackers attack. The CSRF protection system achieved the following goals.

```
<html>
<body>
<img src=""http://localhost/demo/display.php?username=hacker@gmail.com" width="1" height="1"/>
</body>
</html>
```

Figure.10. Code when attacker uses GET request

- We want to provide the safety against the cross site request forgery attack.
- Provide the better solution for prevent CSRF attack. We evolve the identification and authentication step
- For session stabilization.
- We provide a solution that does not temper with the symptoms of some vulnerability but resolve the underlying problem of web session-based deficiencies.
- It properly differentiates the authenticated user.
- It provides the protection against the login CSRF attack and auto submission of from.

VI. REFERENCES

[1].Philippe De Ryck, Lieven Desmet, Thomas Heyman, Frank Piessens, and Wouter Joosen. Csfire: Transparent client-side mitigation of malicious cross-domain requests. In Lecture Notes in Computer Science, pages 1834. Springer Berlin / Heidelberg, 2010.

[2].Giorgio Maone. No script 2.0.9.9. http://noscript.net/, 2011.

[3].Martin Johns and Justus Winter. Request Rodeo: client side protection against session riding. Proceedings of the OWASP Europe 2006 Conference, refereed papers track, Report CW448, pages 517, 2006.

[4]. Justin Samuel. Request policy 0.5.20. http://www .request policy.com, 2011.

[5]. Ramarao R. Tool preventing image based CSRF attacks. http://isea.nitk.ac.in/rod/csrf/PreventImageCSRF/. May, 2009.

[6]. W. Zeller and E. W. Felten, Cross-Site Request Forgeries: Exploitation and Prevention, Technical Report, Princeton University, 2008.

[7]. Nenad Jovanovic, Engin Kirda, and Christopher Kruegel. Preventing cross site request forgery attacks. In IEEE International Conference on Security and Privacy in Communication Networks (Secure Comm), 2006.

[8]. Tatiana Alexenko Mark Jenne suman Deb Roy and Wenjun Zeng, Cross-Site Request Forgery: Attack and Defense. In Proc. IEEE Communications Society (CCNC), 2010.

[9]. Sapna Choudhary, Bhupendra Singh Thakur, DES Encryption Attack detection in Client-Server Communication, International Journal of Advanced Research in Computer Science and Software Engineering. Volume 4, Issue 3, March 2014.

[10]. OWASP. The ten most critical web application security vulnerabilities.

[11]. Sentamilselvan.K,S Lakshmana Pandian, Dr.K. Sathiya murthy. Survey on Cross Site Request Forgery. IEEE International Conference on Research and Development Prospects on Engineering and Technology (IEEE ICRDPET-2013). Vol. 5. No. 5. IEEE, 2013.

[12]. Kappel, Gerti, Birgit Prll, Siegfried Reich, and Werner Retschitzegger. Web engineering. John Wiley Sons, 2006

[13]. Wedman, Shellie, Annette Tetmeyer, Hossein Saiedian. "An analytical study of web application session management mechanisms and HTTP session hijacking attacks." Information Security Journal: A Global Perspective 22, no. 2 (2013), 55-67.

[14]. Chen, Eric Y., Sergey Gorbaty, Astha Singhal, and Collin Jackson. ”Self-exfiltration: The dangers of browser-enforced information flow control.” In Proceedings of the Workshop of Web, vol. 2. 2012.

[15]. W. Zeller and E. W. Felten. Cross-Site Request Forgeries: Exploitation and prevention. Technical report, October 2008. http://www.freedom-to-tinker.com/sites/ default /files/ csrf. pdf.

[16]. J. Burns. Cross Site Reference Forgery: An introduction to A common web application weakness. http://www .isecpartner s.com/documents/XSRFP aper:pdf; 2005: