DOI:

Journal of Computers and Intelligent Systems

Vol. 2, No. 2, 2024, pages 67 - 72

Journal Homepage:

https://journals.iub.edu.pk/index.php/JCIS/

Cross-Site Request Forgery Attacks and Preventions

Muhammad Sajjad[1], Aasir Mehmood[2], Dr. Muhammad Saifullah [3], Junaid Iqbal Baig[4]

[1] University of Central Punjab, FoIT & Computer Science, Pakistan

[2] University of Central Punjab, FoIT & Computer Science, Pakistan

[3] Government Sadiq Egerton Graduate College, Bahawalpur, Pakistan

[4] COMSATS University Islamabad, Vehari Campus, Pakistan

ARTICLE INFO

Article History:

Received:	May	22, 2024
Revised:	May	23, 2024
Accepted:	May	23, 2024
Available Online:	May	24, 2024

Keywords:

CSRF
OWASP
HTTP Header
Prevention

Classification Codes:

Funding:

This research received no specific grant from any funding agency in the public or not-for-profit sector.

ABSTRACT

CSRF stands for Cross-Site Request Forgery, which is among the top web vulnerabilities in which the attacker maliciously exploits a website using victims' credentials and sends unauthorized actions/calls on a trusted web application. In Cross-site request forgery, the attacker sends a malicious forged link to the user. Upon clicking, the forged request is sent on behalf of the user which results in data leakage. Till today, numerous defense mechanisms (on both the client and server sides) have been proposed as the result of increasing attacks and leakage of personal data. Such mechanisms include HTTP header, validation of random tokens, hybrid-model HTTP and content analysis, client-server proxy, and so on. However, even today, such attacks exist and occur. This report analyzes various existing defense mechanisms and models, critically assesses each of them, and addresses the voids in each of them. It also describes how combining two mechanisms help overcome the flaws.

Corresponding Author's Email:

Citation:

1. Introduction

With the advancement in science and technology, everything is shifting on the web. This immaculate change has countless benefits as most of the services which were once done physically are now being done remotely on a website. Besides the advantages, there are also a number of security issues that one has to tackle and go through in order to make the data protected as everything is done remotely. In this project, we'll be looking at one of the significant web security vulnerabilities i.e. Cross-Site Request Forgeries (CSRF). CSRF has been mentioned as one of the top 10 web vulnerabilities mentioned by the Open web application security project (OWASP).[7] It is a malicious process in which the attacker fools the browser into doing some tasks for the attacker which ultimately results in data leakage. As data leakage is a nonacceptable issue in security, therefore, it needs to be prevented.

There are numerous journal papers on the prevention of CSRF and we'll be reviewing them and expecting 1 to figure out which prevention might be the best. As mentioned earlier, the masses have greatly switched towards the internet. This really means a lot of credential information such as business information, payment credentials, personal information, etc. is transferred across the internet. Such information has to be protected and no compromise on it can be tolerated by anyone. CSRF is one of the major web vulnerabilities and there are different kinds of Cross-Site Request Forgeries (CSRF) attacks being done by the attackers. In this report, we researched various latest research articles and figured out multiple kinds of the client as well as server-based defense mechanisms. We also dug in depth about how serves/clients can protect themselves and the browsers they're using from cross site request forgery done by third-party attackers. We figured out the signatures through which the user can be on the verge of being attacked. Furthermore, there are many journals issued on this issue and we briefly went through a few of them and competitively figured out which prevention technique(s) can make the user more protected from the vulnerabilities than the others.

Using a Cross-Site Request Forgeries attack attacker can change the password of the users and can get access to the user's account. Users' accounts can contain a lot of critical information like their personal information and card credentials so if they get leaked it can cause a lot of damage to users which no one would want. The attacker can execute fake transactions so due to this the consequences of CSRF attacks can be severe. For example, the attacker can modify the request and the script to transfer 200 dollars to the attacker's account. The attacker can send that link to different users of the bank so that if any logged-in user clicks on that link 200 dollars will be transferred from that user's account to the attacker's account. So, it is important to address the Cross-Site Request Forgeries attacks and their preventive measures.[9]

In this article, we reviewed numerous journal papers on CSRF prevention to identify the most effective techniques. Our research is valuable for both client-based and server-based defense mechanisms. Moreover, the article examines that how they protect users and browsers from third-party attacks. We also identified specific attack signatures, helping users recognize potential threats.

2. Related Work

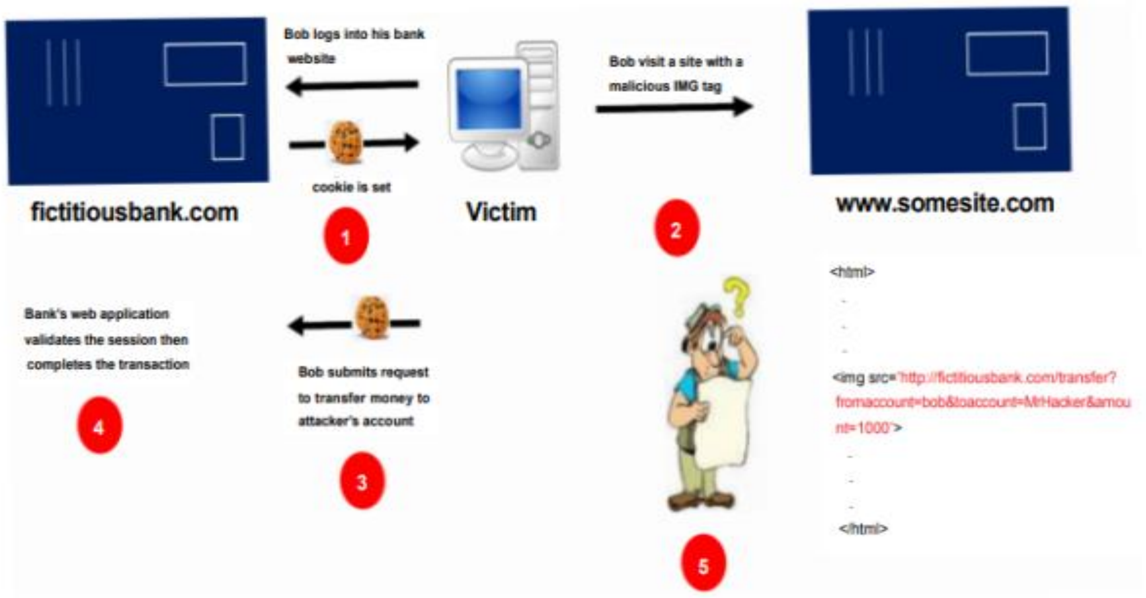
In this paper, the researchers say that there are many existing solutions to CSRF attacks like the client site proxy, the custom HTTP header, and the browser plugin but these solutions are partially protected and are not sufficiently resistant to stop this attack, So researchers present a new way to prevent CSRF 2 attacks, In CSRF attacks the attacker uses the concept of all requests must be user-initiated that are sent from the user and exploit this concept by using a session cookie to send its own payload after the identification of the session cookie.[11] CSRF attacks can be used by users to change their email, make a transfer, and change a password. CSRF attacks are not possible if a few conditions are met those conditions are the following 1: There should not be any unpredictable parameters 2: Only cookie sessions should be used by the application 3: The last one is that there should be a relevant action that can be triggered by the attacker.[1]

To prevent Cross-Site Request Forgeries (CSRF) attacks, the researchers proposed the use of CSRF Token. Three things must be taken care of while using CSRF Token, it should be associated with the user's session, designed in a way so that it should not be predictable, and should be validated each time.[2] One possible way of generating the token is to use a cryptographic strength pseudorandom number generator. Another possible way to create is by employing a strong hash of the entire structure and associating the result with entropy, this also increases the security. To validate the token, it should be stored on the server side and sent in each request sent to the server and the server should verify the token. Validation of tokens also depends on the request method because some applications don't verify tokens on the GET request due to which GET request can be used to trigger a CSRF attack. So tokens should be verified at all request methods. CSRF Token should not be associated with the user session because in this case, the attacker can use his own token as a token belonging to the same session is not verified again.[8]

3. Methodology

The article briefly explains different types of Cross Site Request Forgeries (CSRF) attacks and then proposes a design of a CSRF detection model and its working. The CSRF has three basic components/actors i.e., the attacker, the victim, and the user-trusted website of the victim. The currently working CSRF models are working on two different i.e., The proposed model is made for Google Chrome and is a hybrid CSRF defense method with major properties including HTTP request analysis and content analysis and if, any, CSRF attack is detected, it is passed to the CSRF processing module before passing the request to the server and vice versa. Then using the SimHash Algorithm, if the two hash values don't match then it is considered as an attack, and the user is wanted ultimately and upon the user's order, the request will be passed forward.

- Typical HTTP Request: The input validation by each user should be checked by manual and automated testing techniques.
- Cookie Options Mitigate Input: Securing cookie is an effective method to prevent XSS
- Attack.
- Web Application Firewalls: By using web application firewalls the user can protect the old application. No General solution has found for XSS because one can't be able to protect all the applications from the XSS attacks.
- Client based CSRF defense methods.
- Server based CSRF defense methods [2]


Bob logs into his bank website
cookie is set
fictitiousbank.com
Victim
Bob visit a site with a malicious IMG tag
www.somesite.com
1
2
Bank's web application validates the session then completes the transaction
4
Bob submits request to transfer money to attacker's account
3
5

```
<html>
 -
 -
 -
<img src="http://fictitiousbank.com/transfer?
fromaccount=bob&toaccount=MrHacker&amou
nt=1000">
  -
  -
  -
</html>
```

Figure 1: Diagram of Anatomy of Cross Site Request Forgery (CSRF) Attack.

4. Analysis and Discussion

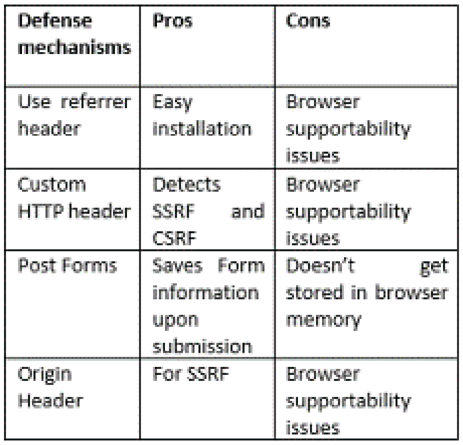
Our countermeasure significantly reduces the vulnerability to CSRF attacks. In the absence of CSRF countermeasures, any origin on the web can launch an attack against a target origin. With our countermeasure in place, the only way an origin can be targeted is if it has explicitly granted control to another origin through a cross-origin POST or redirect. In a previous section, we presented arguments highlighting the difficulty for an attacker to create unintended delegations.[3] To conduct our analysis, we used a real-life dataset of HTTP requests collected over a period of 10 weeks from 50 unique sources. This extensive business analysis revealed that only 1.17 out of 4.7 million requests were treated as delegations according to our approach. We manually examined all 55,300 of these requests and classified them into various categories, as summarized in Table 2, representing different types of commercial transactions. For each category, we evaluated the susceptibility to attack.

A. Third Party Service Mashups

One category, known as ”Mixing of Third-Party Services”, involves the integration of various third-party services into websites, excluding single sign-on services. This integration is usually achieved by including scripts that can trigger GET and/or POST requests from different origins to the included AJAX APIs. In addition, service providers themselves often use cross-origin redirectors to delegate tasks to content delivery networks. Thus, if origin A contains a third-party service S, it becomes vulnerable to CSRF attacks by S. However, it is worth noting that this attack surface is relatively insignificant since S can already launch more significant attacks via a script that exceeds the impact of CSRF -attacks. Additionally, ad providers P redirects content to delivery services D are also vulnerable to CSRF attacks by D when users interact with the ads. However, this attack surface remains relatively insignificant due to the nature of the fiduciary relationship between P and D, which is usually established through legal contracts or service-level agreements (SLAs).[14]

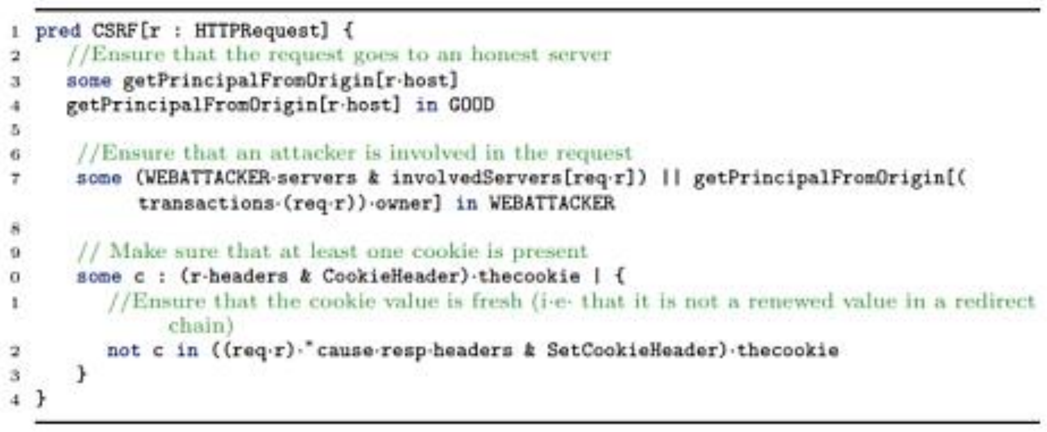
B. Multi-origin Websites

Another category called “Multi-origin Websites” is for larger companies and associations that manage websites. POST requests between origins and redirects between these origins allow similar origins to attack each other. i.e. google. could potentially attack google.com. But even in this case, our countermeasures soften the attack surface. By implementing our countermeasures, we effectively reduced the attack area of CSRF attacks, as demonstrated by our experimental analysis.[10]



Defense mechanisms	Pros	Cons
Use referrer header	Easy installation	Browser supportability issues
Custom HTTP header	Detects SSRF and CSRF	Browser supportability issues
Post Forms	Saves Form information upon submission	Doesn’t get stored in browser memory
Origin Header	For SSRF	Browser supportability issues

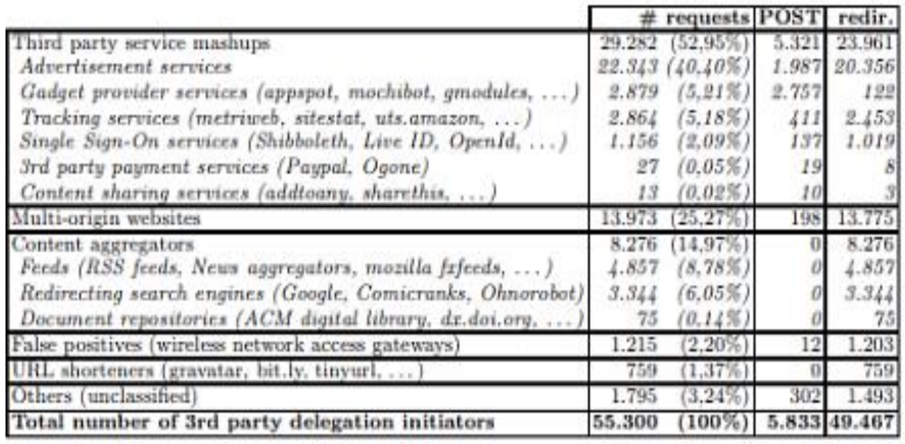
Figure 2. Comparison of Different Defense Mechanisms



```
pred CSRF[r : HTTPRequest] {
  //Ensure that the request goes to an honest server
  some getPrincipalFromOrigin[r·host]
  getPrincipalFromOrigin[r·host] in GOOD

   //Ensure that an attacker is involved in the request
   some (WEBATTACKER·servers & involvedServers[req·r]) || getPrincipalFromOrigin[(
        transactions·(req·r))·owner] in WEBATTACKER

   // Make sure that at least one cookie is present
   some c : (r·headers & CookieHeader)·thecookie | {
      //Ensure that the cookie value is fresh (i·e· that it is not a renewed value in a redirect
           chain)
      not c in ((req·r)·˜cause·resp·headers & SetCookieHeader)·thecookie
   }
}
```



	# requests	POST	redir.
Third party service mashups	29.282 (52,95%)	5.321	23.961
Advertisement services	*22.343 (40,40%)*	*1.987*	*20.356*
Gadget provider services (appspot, mochibot, gmodules, ...)	*2.879 (5,21%)*	*2.757*	*122*
Tracking services (metriweb, sitestat, uts.amazon, ...)	*2.864 (5,18%)*	*411*	*2.453*
Single Sign-On services (Shibboleth, Live ID, OpenId, ...)	*1.156 (2,09%)*	*137*	*1.019*
3rd party payment services (Paypal, Ogone)	*27 (0,05%)*	*19*	*8*
Content sharing services (addtoany, sharethis, ...)	*13 (0,02%)*	*10*	*3*
Multi-origin websites	13.973 (25,27%)	198	13.775
Content aggregators	8.276 (14,97%)	0	8.276
Feeds (RSS feeds, News aggregators, mozilla fxfeeds, ...)	*4.857 (8,78%)*	*0*	*4.857*
Redirecting search engines (Google, Comicranks, Ohnorobot)	*3.344 (6,05%)*	*0*	*3.344*
Document repositories (ACM digital library, dx.doi.org, ...)	*75 (0,14%)*	*0*	*75*
False positives (wireless network access gateways)	1.215 (2,20%)	12	1.203
URL shorteners (gravatar, bit.ly, tinyurl, ...)	759 (1,37%)	0	759
Others (unclassified)	1.795 (3,24%)	302	1.493
Total number of 3rd party delegation initiators	**55.300 (100%)**	**5.833**	**49.467**

Figure 3. Code Igniter

5. Conclusion

To conclude technology is advancing day by day. The number of hackers is also increasing and hackers are also learning new techniques for attacking and bypassing the implemented security. CSRF is a web vulnerability that needs to be addressed and for the protection against these vulnerabilities, there should be some reliable mechanism attacks. CSRF attacks are risker and CSRF attacks are only possible if three conditions are met. Considering these three conditions many solutions are suggested including token value as a hidden field in the form, checking the referer header, using only post requests, and checking the origin header, captcha, and token value disclosed in the URL. But each of these has its own pros and cons. To overcome the shortcoming of these solutions the best solution is to use double-layer protection which is the combination of a token in the URL and a token in the hidden input field, it also increases the security as multiple tokens are used. And it provides protection against the POST and the GET requests. So double-layer protection is the most effective solution.

References

1. Patil, A., et al. (2019). Analysis of cross-site request forgery attack on WebKit. *IJRAR-International Journal of Research and Analytical Reviews (IJRAR)*, 6(2), 145-155.
2. Semastin, E., et al. (2018). Preventive measures for cross-site request forgery attacks on web-based applications. *International Journal of Engineering and Technology UAE*.

3. Zhang, J., Hu, H., & Huo, S. (2021). A browser-based cross-site request forgery detection model. *Journal of Physics: Conference Series*, 1738, 012073. https://doi.org/10.1088/1742-6596/1738/1/012073

4. Zhang, J., Hu, H., & Huo, S. (2021). A browser-based cross-site request forgery detection model. *Journal of Physics: Conference Series*, 1738, 012073. https://doi.org/10.1088/1742-6596/1738/1/012073

5. Code Igniter. (n.d.). Retrieved from http://www.codeigniter.com/

6. Metafilter. (n.d.). Lost password? Retrieved from http://www.metafilter.com/login/lostpassword.mefi

7. Ruby on Rails. (n.d.). Retrieved from http://www.rubyonrails.org

8. ING Press Release. (2006, August). Retrieved from http://www.rsa.com/pressrelease.aspx?id=7220

9. Suyash Bhogawar, C. A., Nuthakki, S., Venugopal, S. M., & Mullankandy, S. The Ethical and Social Implications of Using AI in Healthcare-A Literature Review.

10. Yager, N., & Amin, A. (2004). Fingerprint verification based on minutiae features: A review. *Pattern Analysis and Applications*, 7(1), 94-113.

11. Nuthakki, S., Kumar, S., Kulkarni, C. S., & Nuthakki, Y. (2022). Role of AI Enabled Smart Meters to Enhance Customer Satisfaction. *International Journal of Computer Science and Mobile Computing*, *11*(12), 99-107.

12. Schreiber, T. (2004). Session riding: A widespread vulnerability in today's web applications. Retrieved from http://www.securenet.de/papers/SessionRiding.pdf

13. Shiflett, C. (2004, December). Security corner: Cross-site request forgeries. Retrieved from http://shiflett.org/articles/cross-site-request-forgeries

14. Shiflett, C. (2006, October). The crossdomain.xml witch hunt. Retrieved from http://shiflett.org/blog/2006/oct/the-crossdomain.xml-witch-hunt