CSRF – Cross Site Request Forgery: Causes and Mitigation

Divya Aradhya

Saint Leo University

Author's Note

Contact: divya.aradhya@saintleo.edu

August 4, 2017

Table of Contents

Abstract

With a growing number of businesses going online, and an ever increasing number of websites and Internet users, attacks on websites are alarming high. The Cross-Site Request Forgery (CSRF or XSRF) is one such attack that manipulates an authenticated end user to execute malicious actions on a web application.

This paper is a study of the CSRF attacks, its relevance and significance in the realm of information security, the primary issues pertaining to it, the current mitigation methods employed (“what has been done”), and insights and remarks based on these findings and study.

Brief Description

Most websites on the Internet require a user to identify and authenticate before being authorized to access the web services. Hackers execute cross-site request forgery attacks that make use of the authentication and privileged access and run their own malicious scripts to manipulate the website information and services to their (hackers) advantage.

CSRF attacks specifically target state-changing requests more than performing direct theft of data, though this is possible too. Hackers often employ a mix of social engineering attacks (phishing through email or phone text messages) and tricks the genuine user into executing their malicious scripts.

A successful CSRF attack can lead to serious losses by changing financial transactions to transfer funds, or changing personal details and identity and credential information.

The Open Web Application Security Project – OWASP – puts Cross-Site Request Forgery attacks as #8 on their list of top ten vulnerabilities and attacks (“Top 10 2013-Top 10”)

Relevance and Significance

In the Cross-Site Request Forgery (XSRF or CSRF) attack, a crucial website session can be completely taken over by the hacker, resulting in severe losses for the genuine authenticated user, and the website itself. An intruder masquerades as a legitimate and trusted user and manipulate the session tokens and website actions. An XSRF attack can be used to modify “firewall settings, post unauthorized data on a forum or conduct fraudulent financial transactions.” (“What is Cross-Site Request Forgery?”)

More often than not, the compromised user is completely in the dark that he was a victim of a CSRF attack. The user and website may become aware of the attack only after they discover the losses that resulted from it, and at that stage, the recovery may be almost impossible.

The National Vulnerability Database (NVD) displays thousands of CSRF vulnerabilities being discovered every day in various web-based applications and tools. (“NVD – Results”)

Users of popular web services like Google, Facebook, Gmail, Paypal, and Digg, have all been subjected to CSRF attacks.

Primary Issues

The primary issue around Cross-Site Reference Forgery is that they involve a bit social engineering to launch, and then once it has launched, it is difficult to detect and stop. The attacks, though less common than other web attacks, can lead to severe financial an identity losses.

The effects of SQL injections can result in the following security breaches-

- Transfer of money from a financial institution to the hacker’s bank account
- Manipulation of the web server’s content management system to edit, alter, and even delete crucial information
- Change a user’s password, and take over their account
- Manipulate e-commerce websites and auto add items into shopping baskets
- Change the delivery address of an order (“Understanding SQL Injection”).

Below is an example of a CSRF attack that highlights the issue associated with it.

Many websites have a “My Account” webpage that displays the user’s account information and allows them to change their authentication details – email-id, password, phone number (for two-factor authentication) etc.

This change in details is made, either through an HTTP POST request, or through an HTTP GET request. Now, if an attacker is aware of the structure of these web requests, he or she can forge it and manipulate it for malicious intents.

In the below code, the data to be changed is stored in the parameter that is called as “**EmailID**”. If the genuine authenticating user can be tricked to click on a phishing link then the following code can be used to change the email address that is used for authentication and credentials check.

HTTP GET example-

```
<html>
            <body>
                        <H1> My Bank Account Page </H1>
                        <img src=https://mybank.com/MyAccount.aspx?EmailID=hacker@forgery.com width=”1” height=”1” />
            </body>
</html>
```

HTTP POST example-

```
<html>
            <body>
                        <form name=”My Bank Account” method=”post” action=https://mybank.com/MyAccount>
                                    <input type=”hidden” name=”EmailID” value=” hacker@forgery.com”>
                        </form>
                        <script>document.MyAccount.submit()</script>
            </body>
</html>
```

The GET page would look blank as the image is just 1 pixel by 1 pixel, in dimension, and is invisible. The POST would also look empty. (“Cross-site request forgery: Lessons from a CSRF attack example “)

The user will just think the page didn’t load and not realize that he has been hacked. However, at the background, the code has changed the authenticating email-id in the bank account. The hacker can now use the “Forgot password” facility of the bank website and the new password will be sent to hacker@forgery.com. The hacker can now log into the bank account and transfer funds to his account, and can essentially manipulate any other financial transactions.

The user will not be able to log into his account – and by the time he contacts the bank officials, convinces them of his identity, gets the account email-id and password reset, the attacker has long made away with the loot.

Current Mitigation Solutions

Having realized the gravity of losses caused by CSRF attacks, organizations like the OWASP have complied the findings of various security research publications and put out guidelines for best practices. ("Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet")

In order to prevent CSRF injections, the primary defenses would include-

- Checking the Origin Header – if the origin header is present, its value needs to be verified to match the target origin
- Checking the Referrer Header – if the origin header is not present, the hostnames in the referrer header should be verified to match the target origin
- Blocking - If both origin and referrer headers are missing the request should be blocked
- Synchronizer CSRF Tokens – Requiring all state change operations to have a secure verifiable random token that is-
    - Unique
    - Large in value and random
    - Generated by a random number generator that is cryptographically secure
    - Hidden in the form
- Double Submit Cookie – this is a random value that acts as both a cookie and a request parameter

- Re-Authentication – asking a user to type in their password again for critical operations (transfer of funds, change in address, purchases etc.)
- One-Time token – multi-factor authentication involving an external device like a phone
- CAPTCHA – to ensure the genuine user is aware of the request and prove they are human and authenticating it
- Security Awareness – Building security awareness amongst users on CSRF, phishing, authentication, and security.

Insights and Remarks

Based on the research and findings, it is clear that CSRF attacks can cause complete breach of confidentiality, integrity, and even availability of data, though the risk of integrity is the highest. The repercussions of CSRF attacks have caused severe financial damages to end users, organizations, and online services. This problem, though less common than SQL injections and XSS attacks, can be hard to mitigate as it involves a combination of social engineering and technical hacks.

End users are manipulated to click on links that lead them to authenticate and let malicious scripts take over. It is important to educate them on the dangers of phishing and identifying phish attacks and being cautious.

CSRF attacks are seen to be significantly harder to defend against than XSS or XST attacks, even though they are less common in occurrence than both are. One of the reasons for this is that CSRF attacks have not received as much attention. Another important reason is the fact that it can be difficult to determine that if the action requested by a user was a genuine request or a manipulated attack, as the CSRF launches itself after authentication. It is difficult to program the server and web services to differentiate between an authentic HTTP request and a forged request.

While strict precautions can be used to verify the identity of a user attempting to access a website, regular frequent users may get annoyed by frequent requests for authentication and can stop using the web application altogether. The use of cryptographic tokens, however, can provide frequent authentication in the background serving to protect the user, and make their web experience safe and pleasant.

This research for this paper does highlights the grave repercussions of a CSRF attack, and the need for more awareness about them for developers, security professional, and end users.

References

Top 10 2013-Top 10. (n.d.). Retrieved from https://www.owasp.org/index.php/Top_10_2013-Top_10

What is Cross-Site Request Forgery? (2006, October). Retrieved from http://searchsoftwarequality.techtarget.com/definition/cross-site-request-forgery

NVD - Results. (2017, August 4). Retrieved from https://nvd.nist.gov/vuln/search/results?adv_search=false&form_type=basic&results_type=overview&search_type=all&query=CSRF

Shapland, R. (n.d.). Cross-site request forgery: Lessons from a CSRF attack example. Retrieved from http://www.computerweekly.com/tip/Cross-site-request-forgery-Lessons-from-a-CSRF-attack-example

Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet. (n.d.). Retrieved from https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet