ISSN No:-2456-2165

An Automated Detection System of Cross Site Request Forgery (CSRF) Vulnerability in Web Applications

Md. Afzal Ismail
Department of Software Engineering
Daffodil International University, Bangladesh

Md. Maruf Hassan
Assistant Professor, Department of Software Engineering
Daffodil International University, Bangladesh

Abstract:- **In the modern era of technology, the usage of web applications has become enormous. Web applications are now dealing with much more sensitive data. As web applications dealing with sensitive data, they are encountering lots of threats. Intruders are always trying to find new ways to penetrate these applications and misuse them. The attackers use vulnerabilities to perform those attacks. Cross site request forgery aka CSRF is one of the vital threats and top ranked web application vulnerability. CSRF attack is a type of attack where end users are forced to perform unwanted actions on a web application in which they are currently authenticated. In some previous researches, several numbers of case studies are found. In many researches, different types of models are proposed and developed. To secure the web applications that are vulnerable to CSRF vulnerability, many more studies need to be done in this field. Therefore, there is not enough studies on automated system to detect this CSRF vulnerability. Therefore, the key focus of this research is to develop an automated web application vulnerability detection model for detecting the CSRF vulnerability in web applications. My proposed solution is to do real time scan of CSRF vulnerability in given URL.**

Keywords:- *CSRF Vulnerability, Cyber Security, Automated detection Tool, Web Application Vulnerability.*

I. INTRODUCTION

Cross site request forgery is a web application vulnerability. If this vulnerability exists in any web applications means that there is some weakness in system or it is misconfigured. It allows an attacker to access sensitive data, modify data, perform state changing actions etc. To make this attack possible, an attacker tricks the user and make requests from users' browser and to the applications where the user is currently authenticated. Successful CSRF attack can lead to serious security breaches for both the website as well as the end user.

According to datareportal, the number of internet users are getting humongous day by day and by the end of April 2021, the total number hits the milestone of 4.72 billion overall users and most of the internet users are using web applications.[https://datareportal.com/global-digital- overview]

As the targeted user community is huge, and getting larger day by day, web application security has become a major issue because it constantly dealing with sensitive information and personal data. Moreover, exploiting web application vulnerability of web applications is increasing due to the system flaws.

II. RELATED WORK

Cross site request forgery is a client-side attack where users tricked into doing some actions or perform unwanted activities. There's a vulnerability called Cross-site scripting in the web application. The reason behind this vulnerability is improper input validation. This vulnerability creates so many problems. These problems can be happened for both server-side and client side of that web application with the help of CSRF attacks. (Nadar, V. M. et al. 2018).

V. M. Nadar et al. developed an enhanced detecting model that can detect cross-site request forgery attack, Broken Authentication and Session Management attack within the same simulation environment. Their work has only checked for the malicious script (V. M. Nadar et al. 2018).

The Open Web Application Security Project also known as OWASP has listed SQL injection, CSRF and XSS vulnerabilities as the most frequently exploited vulnerabilities. An intruder or attacker fixes the target or victim user and try to executes malicious JavaScript in the target's browser. By approaching an attack with this process, the attacker never directly targets his victim but he exploits avulnerability in a web application which the targeted victim visits. The malicious script that an attacker wants to execute is a key factor in these types of attack. If the attacker has become successful and plants the malicious script successfully and executed, the attacker might be able to access sensitive information such as sessionID, cookies etc. (Nagpal, Chauhan et al. 2017).

Nagpal et al. developed a system engine which detects SQL injection and stored CSRF attacks. Their study only works forweb applications based on php. (Nagpal, Chauhan et al. 2017).

The CSRF vulnerability lies in a web application wherebasically state changing actions are performed. So, to detectthe vulnerability, it's very important to track where and when a security related state change action is performed within theweb application. (Liu, Shen et al. 2020).Liu et al. developed a CSRF vulnerability detection model based on graph data mining which can only detect the vulnerability accurately if astate changing attack happens. (Liu, Shen et al. 2020).

Web applications usually use one type token which is called secret validation token to prevent CSRF attacks. Using the secret validation token is a well stablished server-side protection against Cross-site request forgery attacks. The token basically works by validating the token information which is send along with the other information to the http request to determine that the request is coming is actually coming from an authorized user or not.(Laila, Moustafa,.2018). Laila and Moustafa have developed a web browser extension for mitigating CSRF attacks.(Laila, Moustafa,.2018).

To perform a CSRF attack, the attacker doesn't need to modify anything within the user's response or request. It will be enough for the attacker if the user visits the malicious websites of the attacker and from this malicious website, the attack will be launched. The author's also included that, if any web application is vulnerable to CSRF attack, the web application will be eventually exploitable by any malicious websites on the web.(Stefano, Conti et al.2019).Stefano et al. developed a solution using machine learning for the black box detection of CSRF vulnerability.(Stefano, Conti et al.2019).

Contemplating the previous works best of my knowledge, and the nature of the works, there are minimal number of works to detect cross site request forgery vulnerability or attack automatically. Furthermore, most of the works are to detect the attack not the vulnerability. Therefore, a system is proposed to automatically detect CSRF vulnerability.

III. THE PROPOSED METHODOLOGY

In this section, the system architecture of our "CSRFD", ourproposed solution is discussed. The workflow diagram in the figure 1 demonstrate the system architecture of our proposedsolution.

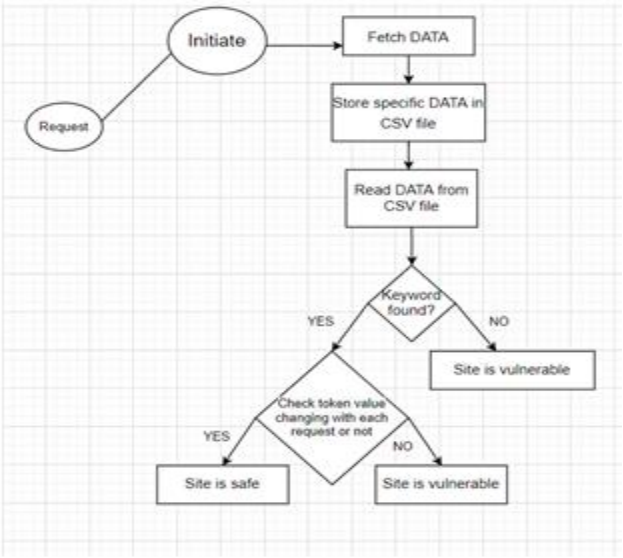

Figure 1: System Architecture

This section is divided into three sub-section named:

- Web scrapping
- Analyze the data
- Response

Web Scrapping: To implement a CSRF vulnerability detector we have first used a web scrapping technique to extract data from URL. Web scrappers work by collecting URLs of the pages from which pages we wanted the data. In scrapping, it makes a request to the targeted URL and fetch necessary data and gives us the room to save that data into CSV files or in other formats. In CSRFD, we have used Beautifulsoup(BS4) to extract data from URL. It will first take the targeted URL and make a request to that URL. Thereafter, it will fetch necessary data and save it in a CSV file.

Analyze the data: In scrapping section, we have saved the data into CSV file. It is time to use that data. The system will open the CSV file and read the data from that file. Subsequently reading that data, the system will look for CSRF tokens. As far as CSRF vulnerability is concerned, to prevent CSRF vulnerability, it is best method to use a secret token that the attacker cannot get. An attacker may get the sessionID but we need an additional token along with that sessionID which will be total unknown to the attacker. Hence, the attacker cannot misuse the sessionID. Here is the key thing, to mitigate this vulnerability, a very much common and renowned approach is to use CSRF tokens, or we can say Anti CSRF tokens. Web applications are usually developed based on frameworks, and various frameworks offer various types of CSRF tokens or anti CSRF tokens. According to the portswigger, a CSRF token is a type of token which is unique and secret, which is an unpredictable value that an attacker cannot guess. The value is generated by the server-side on the web application. This CSRF token is placed in the client-side of the web application in such way that it included with every upcoming http request that the user has made. [https://portswigger.net/web-security/csrf/tokens]

ISSN No:-2456-2165

These tokens are present in the hidden field and encrypted in such way that attackers cannot guess the actual data. Moreover, this value changes with every request. Hence, the attackers have no chance to reuse a previous piece of token data.

Here, two major conditions have encountered.

1. If a web application using CSRF tokens, it will free from CSRF vulnerabilities.
2. The token of that field has to change with every request.

Hence, in our system, we will first look for the csrf tokens, and if the token found, we will again send the request to that url and check that the previous value of the token is matching with the current token value or not.

Response: Based on the conditions, our proposed system will give necessary responses. There is some difference in the naming convention of that CSRF token or anti CSRF token. For example, .NET framkework uses the name "requestverificationtoken". On the other hand, Laravel framework use this token as "_token", Rubi uses as "authenticity_token". Most commonly used framework these days Django uses it as "csrfmiddlewaretoken" etc. In our system, it will first look for the token, if the token is not found than the site may vulnerable to CSRF attacks. But if the token exists, the web application may not be vulnerable to CSRF. However, to make sure that the web application is actually free from CSRF vulnerability, we have to check for the randomness of that token. In that case, we need to check if the token value is changing with every request or not. Based on the responses, we can come to an interpretation about the web application.

IV. THE PROPOSED ALGORITHM & IMPLEMENTATION

In this section, we will describe the central algorithm of our system CSRFD. The figure 2 shows the algorithm of our solution –

ALGORITHM 1: ALGORITHM FOR CSRFD

```
1:  Take seed URL;
2:  if seedURL=valid then
3:      Execute URL for source code using BS4;
4:      Open a CSV file;
5:      Store data into CSV file;
6:      Read CSV file and assign data into a list;
7:      value ← list[0]; // Storing data in a variable
8:          if value <> keyword // Checking for specific keyword
9:              return response;
10:         else
11:             Send request in seed URL again;
12:             get data;
13:             if value==data
14:                 return response;
15:             else
16:                 return response;
17:     else
18:         return exception;
```

Figure 2: Algorithm

First of all, the system will take the seed URL. Now, it will check whether the URL is given in an appropriate format or not. If the format of URL is wrong, it will throw an exception to provide the URL in the correct format. The correct format will be shown in that exception. If the URL is valid and the format is appropriate, then the web scrapper will take the seed URL and send request to that targeted URL. In our system, the Beautifulsoup, A python-based library will do the scrapping for our system. The version of Beautifulsoup that we have used to develop our system is BS4 version. After sending a request to that targeted URL, the web scrapper will fetch the data from that web application that we will need. After fetching that data, we will open a CSV file to store data, and necessary data will be stored in that CSV file. Now, it will read that CSV file and store necessary data in a list. Promptly, the system will check for conditions, and based on that condition, it will give us response. The conditions that will be checked described in the "Response" section.

Figure 3 will give a glance over some of the base conditions that are used in our solution:

```
if value.find('RequestVerificationToken') == -1:
    if value.find("authenticity_token") == -1:
        if value.find("csrfmiddlewaretoken") == -1:
```

Figure 3: Some used conditions in our system

Test Suite up : There are 22 web applications be selected to perform our tests. Here, we have collected some commonly used web applications by some users and included own site to perform works. Table 1 indicating the application names and the types of that application. While implementing our tool CSRFD, we have used a normal computer running on 64-bit Windows 10. The spec sheet kind of looks like this (1.80 GHz, Intel i7, 8GB RAM).

Table 1: Details of test applications

Application	Type
YTS	Entertainment (Movie)
Zedge	Personalization
Rokomari	Online Book store
Netflix	Streaming site
Programming World	Blog
Standard Chattered	Online Banking
Charismatic	Online Cloth store
Brac bank	Online Banking
Artstation	Showcasing Platform
Facebook	Social Site
Sundorban Courier	Courier Service
DailyMotion	Video Sharing
Twitter	Social platform
Digital Photography School	Learning
fmovies	Free streaming site
7lakesbeauty	Beauty Shop
Mayo Clinic	Hospital
freepik	Resource site for graphics
LaReve	Clothing store
EB medicine	Evidence based medical

	help
DiabaticExpo	Health
BMIobject	Architecture Designing

V. EXPERIMENTAL RESULTS

The results of our experiment are listed in the Table 2. In thattable, there is two main columns, where column 1 will refer to the application name, and column 2 is referring to the results of our experiment. Column 2 is divided into 3 sections. The first section is called “Successful”. In this portion, we will consider that is our automated tool successfully detects or not. The second section is called “False Positive” and in this section we will consider that is our tool showing any non– vulnerable application vulnerable. In the third section named “False negative” is basically a consideration about is our tool showing any vulnerable site non-vulnerable or not.

	Detection		
Application	**Successful**	**False Positive**	**False Negative**
YTS	✓		
Zedge	✓		
Rokomari	✓		
Netflix	✓		
Programming World	✓		
Standard Chattered	✓		
Charismatic	✓		
Brac bank	✓		
Artstation	✓		
Facebook		✓	
Sundorban Courier	✓		
DailyMotion	✓		
Twitter		✓	
Digital Photography School	✓		
fmovies	✓		
7lakesbeauty	✓		
Mayo Clinic	✓		
freepik	✓		
LaReve	✓		
EB medicine	✓		
DiabaticExpo	✓		
BMIobject	✓		

Table 2: Effectiveness of CSRFD

Our proposed system CSRFD detects successfully 20 web applications over 22 web applications. The Figure 4 illustrate the potency of our experiment vulnerability successfully in 20 web applications. The false positive number is also minimal. The number is only 2. But the number of false positive is 0(Zero). That means, if our system gives us response that a web application is not vulnerable, the experimental results refers the web application is expected to be safe. Moreover, if it is showing a response to the web application vulnerable, it is very likely to be vulnerable, but we need to keep in mind that the response may be a false positive that sometimes.

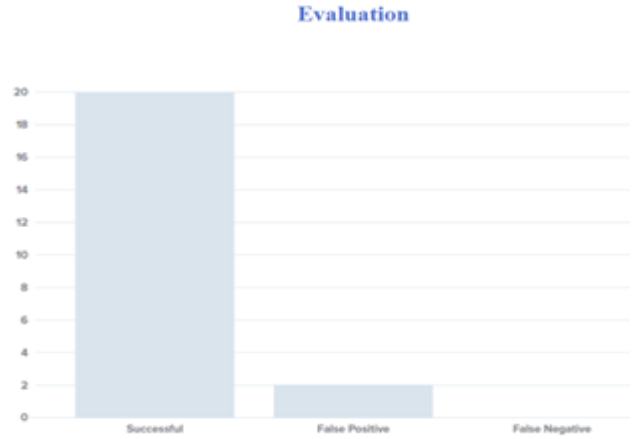

Figure 4: Potency of CSRFD

The effectiveness of our tool CSRFD achieved 90.91 percent accuracy with 9.09 percent of false positive and 0(zero) percent of false negative rate. For an analytical type tool suchas CSRFD, the performance is acceptable.

VI. CONCLUSION

Web applications are usually restricted their operations/options based on the user type and user authentication. For example, A web application may have a case scenario where without being logged in, the user cannot access some of the services that the application provides. Hence, in our system we cannot perform the experiment on each forms of web applications because of proper authentication. Hence, the concept is that, checking in the Sign up or sign in forms. The justification for the concept is that, our system works based on token analysis and these tokens are need to place in each form where post request applies. Based on this concept, we have performed our experiment. Total number of 22 web applications have taken and the result is satisfactory. From table 2, we can see that among these 22 web applications, our system detects. An automated tool is implemented to detect the Cross-site request forgery vulnerability. The implemented solution uses a web scrapper to fetch data and find vulnerabilities within the web form. This tool can perform a crucial role in the detection of Cross-site Request Forgery vulnerability. The effectiveness of this tool is 90.91 percent. Our future plan is that to construct a Finite state automata (FSM) for this detection model.

REFERENCES

[1]. Farah, T., Shojol, M., Hassan, M., & Alam, D. (2016, July). Assessment of vulnerabilities of web applications of Bangladesh: A case study of XSS & CSRF. In 2016 sixth international conference on digital information and communicationtechnology and its applications (DICTAP) (pp. 74-78). IEEE.

[2]. Lalia, S., & Moustafa, K. (2019, April). Implementation of Web Browser Extension for Mitigating CSRF Attack. In World Conference on Information Systems and Technologies (pp. 867-880). Springer, Cham.
[3]. Liu, C., Shen, X., Gao, M., & Dai, W. (2020,
[4]. September). CSRF Detection Based on Graph Data Mining. In 2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE) (pp. 475- 480). IEEE.
[5]. Nadar, V. M., Chatterjee, M., & Jacob, L. (2018). A Defensive Approach for CSRF and Broken Authentication and Session Management Attack. In Ambient Communications and Computer Systems (pp. 577-588). Springer, Singapore.
[6]. Nagpal, B., Chauhan, N., & Singh, N. (2017). SECSIX: Security engine for CSRF, SQL injection and XSS attacks. International Journal of System Assurance Engineering and Management, 8(2), 631-644.
[7]. Soleimani, H., Hadavi, M. A., & Bagherdaei, A. (2017, September). WAVE: Black Box Detection of XSS,CSRF and Information Leakage Vulnerabilities. In 2017 14th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC) (pp. 19-24). IEEE.
[8]. Total Number of internet user worldwide | Datareportal. Retrieved May 14 from, https://datareportal.com/global- digital-overview
[9]. The CSRF detection cheat sheet | OWASP. Retrieved May 03 from, https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
[10]. Web Security CSRF token | Portswigger. Retrieved April 20 from, https://portswigger.net/web-security/csrf/tokens