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

By

Md. Afzal Ismail
ID: 171-35-1799

A thesis submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Software Engineering

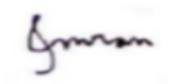
Department of Software Engineering
DAFFODIL INTERNATIONAL UNIVERSITY

Fall – 2020

APPROVAL

This thesis titled on "**An Automated detection of Cross Site Request Forgery (CSRF) Vulnerability in Web Applications**", submitted by **Md. Afzal Ismail**, **(Student ID: 171-35-1799)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

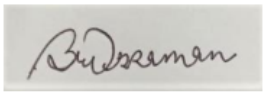


Dr. Imran Mahmud
Associate Professor and Head
Department of Software Engineering
Daffodil International University

Chairman

Syeda Sumbul Hossain
Senior Lecturer
Department of Software Engineering
Daffodil International University

Internal Examiner 1



Khalid Been Badruzzaman Biplob
Senior Lecturer
Department of Software Engineering
Daffodil International University

Internal Examiner 2

Professor Dr. Mohammed Nasir Uddin
Department of Computer Science and Engineering
Jagannath University, Dhaka

External Examiner

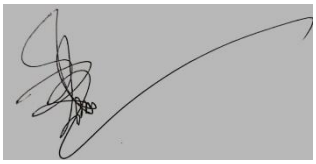
DECLARATION

I hereby declere that this thesis has been done by **me** under the supervission of **Md. Maruf Hassan, Assistant Professor,** Department of Software Engineering, Daffodil International University. It also declere that nithor this thesis nor any part of this has been submitted elesewhere for award of any degree.

_____

Md. Afzal Ismail
Student ID: 171-35-1799
Batch: 22th batch
Department of Software Engineering
Faculty of Science & Information Technology
Daffodil International University

Certified by:



_____

Md. Maruf Hassan
Assistant Professor
Department of Software Engineering
Faculty of Science & Information Technology
Daffodil International University

ACKNOWLEDGEMENT

I would first like to thank the Almighty for giving me the chance to walk through final year. I have learnt so many things in previous years which will definitely make myself worthy in recent days. I'm grateful to all of our fellow faculties for this. I would like to express my immense regards to my supervisor Md. Maruf Hassan, Assistant Professor, DIU for giving me guidelines and showing me necessary paths to do the thesis on the topic of Automatic detection of web application vulnerability which helped me a lot. Besides my supervisor, I acknowledge the authority of Cyber Security Center, DIU for giving my necessary supports in need.

TABLE OF CONTANT

Table of Contents

LIST OF TABLES

LIST OF FIGURES

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.

CHAPTER 1

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.

Background

Over the past few years, web applications have become part and particle of the businesses, organizations and solutions related to customer behavior. Datareportal says, by the end of April 2021, the number of internet users are 4.72 billion and that more than sixty percent of the world's total population[https://datareportal.com/global-digital-overview]

Moreover, most of the internet users are somewhat related or use web applications. To provide better service to customers as well as all clients, most of the business or organizations are solely depending on web applications.

It is known that a pandemic situation is going on. It also encourages clients to rely more on web applications. Datareportal also says that, the total number of internet users

around the world grew by 332 million in the past 12 months – more than 900,000 new users each day.

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. Cross site request forgery is a client-side attack that allows an attacker to execute malicious code or script to the user end and perform state changing actions or unwanted actions that the user does not intend to perform. In this attack, the attacker may use social engineering to trick the user into clicking something and execute malicious code. After clicking to the link, the user may redirect to an infected site, and from that site a forge request will be delivered to the users' browser and gets executed.

1.1 Motivation of the Research

In present days, people are abundantly dependent on the web applications to conduct their daily needs, to get their jobs done many people consecutively using web applications throughout the whole day. To perform clients or users desired jobs, user information's are stored in web applications. This information is like honeypots to the attackers.

The attackers can use or modify this user information and can-do various types of jobs if they wanted to, which attacks the attacker to attack on a system through a vulnerability and Cross site request forgery is one of them.

According to the Open Web Application Security Project also known as OWASP, If the user who has been attacked is responsible for some administrative work for any web application, attacking that user can leads serious damage and compromise that web

application through CSRF attack[https://owasp.org/www-community/attacks/csrf]

To detect CSRF vulnerabilities, there are some solutions or models out there but each solution has some limitations. In other word, we can say that the solutions are not dynamic enough to detect all types of CSRF vulnerabilities as the attack can be placed by various methods.

1.2 Problem Statement

After studying and analyzing previous works about CSRF Vulnerability detection it has found that, there is least research methodologies that has the capability to automatically detect the vulnerability efficiently and there's scope to do some improvement on this field.

The existing solutions are facing difficulties in identifying false positive/negative result to improve the accuracy.

1.3 Research Questions

With having this background, motivation and problem statement in mind, I pose the following questions:

Question 1: Is my proposed solution can effectively detect Cross site request forgery vulnerability?

Question 2: Is my implemented tool providing better accuracy considering the false negative and false positive analysis with the compared existing solution?

1.4 Research Objectives

- To propose an automated solution which can identify cross site request vulnerability

- To implement a solution named "CSRFD" based on the proposed solution
- To evaluate the potency of our designed tool "CSRFD"

1.6 Research Scope

The experimental processes have conducted on several types of web applications including banking sites, entertainment category sites, blogs, shopping etc. Hence, there is some complications regarding cyber security issues and cyber laws. For this reason, we can't test our experiment on every site available on the internet. In that case, we have tested with some demo sites owned by me and my friends.

1.7 Thesis Organization

In this research, APA referencing system has been used in this document. The paper has been furnished with five chapters which is described below:

Chapter 1: Research background, motivation, problem statement, objectives and scopes are discussed here.

Chapter 2: In this chapter the discussion about the existing related works performed. Also figured out the research gap.

Chapter 3: The discussion about the research methodology has been covered in this chapter.

Chapter 4: Result analysis and discussion of the proposed model will be covered in this chapter.

Chapter 5: Findings, conclusion and the future work will be covered in this segment.

CHAPTER 2

LITERATURE REVIEW

In this section, there will be discussion about the previous researches and findings about Cross site request forgery vulnerabilities/attacks which is focused both on server side and client side of the web applications. 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 intensified detecting model which can detect CSRF attacks. There work also can detect Session Management attacks and Broken Authentication attacks 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. When the attacker approaches, he actually exploits a vulnerability in a web application which the targeted victim uses. 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 get the access to sensitive information such as sessionID, the cookies and so on. (Nagpal, Chauhan et al. 2017).Nagpal et al. developed a system

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

The CSRF vulnerability lies in a web application where basically state changing actions are performed. Hence, to detect the vulnerability, it is very important to track where and when a security related state change action is performed within the web 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 a state 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 to give protection and mitigate the 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 does not 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.

CHAPTER 3

RESEARCH METHODOLOGY

In this section, the system architecture of CSRFD is discussed. This section is mainly divided into two major parts. In the first section there is discussion about the work flow diagram and about the model in it. In the second section, the discussion will be about the algorithm of my proposed solution.

3.1 CSRFD Model

The workflow diagram in the figure 3.1 demonstrate the system architecture of my proposed solution.

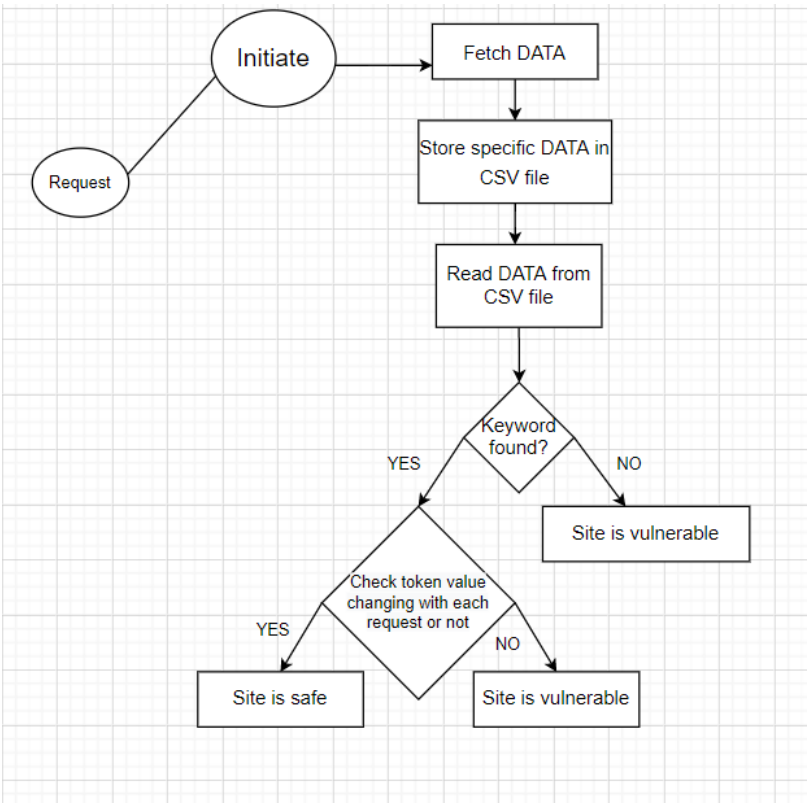


Figure 3.1: System architecture of CSRFD

This section is divided into three sub-section named:

- Web scrapping
- Analyze the data
- Response

3.1.1 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.

3.1.2 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)

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.

3.1.3 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.

3.2 Algorithm and Implementation

In this section, we will describe the central algorithm of our system CSRFD. The figure 3.2.1 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 3.2.1: 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 "CSRFD Model" section.

Figure 3.2.2 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 3.2.2: Some used conditions in our solution

Test suite setup:

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

CHAPTER 4

RESULTS AND DISCUSSION

In this section, we have discussed about the results. Analysis and discussion about the implemented CSRFD experiment will be covered.

4.1 Discussion

The results of our experiment are listed in the Table 2. In that table, 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.

Table 2: Effectiveness of CSRFD

Application	Detection		
	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	✓		

4.2 Analysis

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. So, 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 placed 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 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.

4.3 Performance

Our proposed system CSRFD detects successfully 20 web applications over 22 web applications. The Figure 4.3.1 illustrate the potency of our experiment

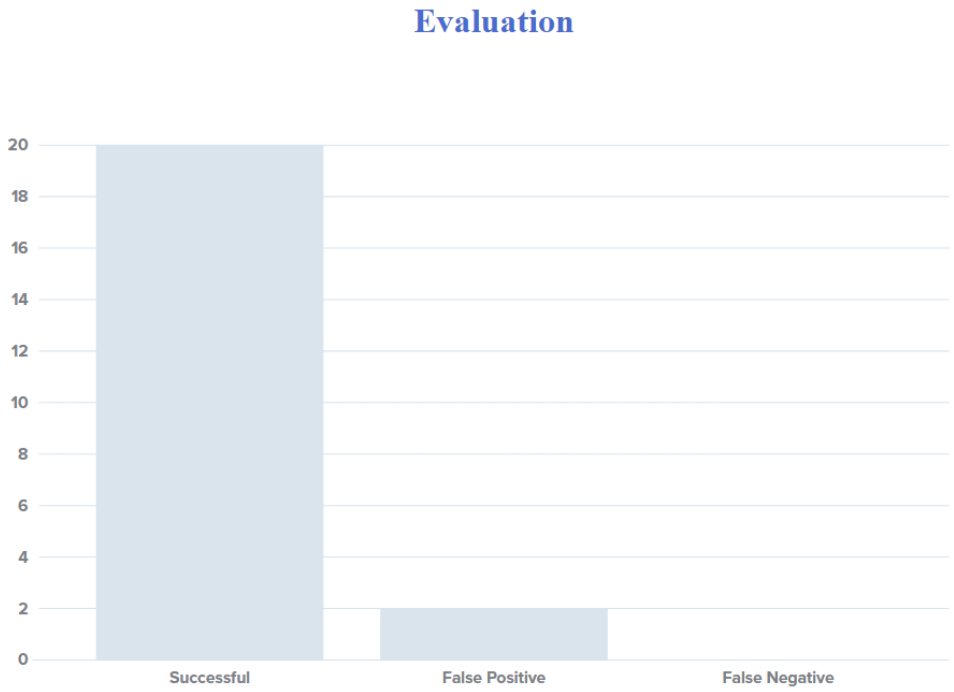


Figure 4.3.1: Potency of CSRFD

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 such as CSRFD, the performance is acceptable.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Findings and Contribution

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. Sufficiency of that tool is acceptable considering the other solutions.

5.2 Recommendations for Future Works

Our future plan is that to construct a Finite state automata (FSM) for this detection model. We will try to integrate machine learning with this model. The expectation is, integration of machine learning with this work will lead us to make a more usable tool.

REFERENCES

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 communication technology and its applications (DICTAP) (pp. 74-78). IEEE.

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.

Liu, C., Shen, X., Gao, M., & Dai, W. (2020, 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.

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.

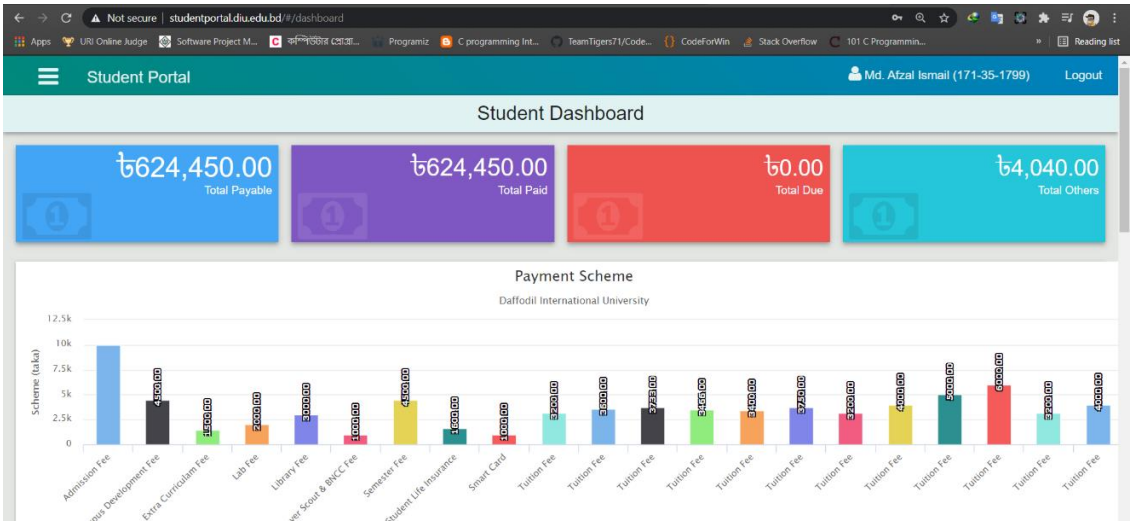
Nagpal, B., Chauhan, N., & Singh, N. (2017). SECSIX: Security engine for CSRF, SQL injection and XSS attacks. International Journal of System Assurance Engineering and Management, 8(2), 631-644.

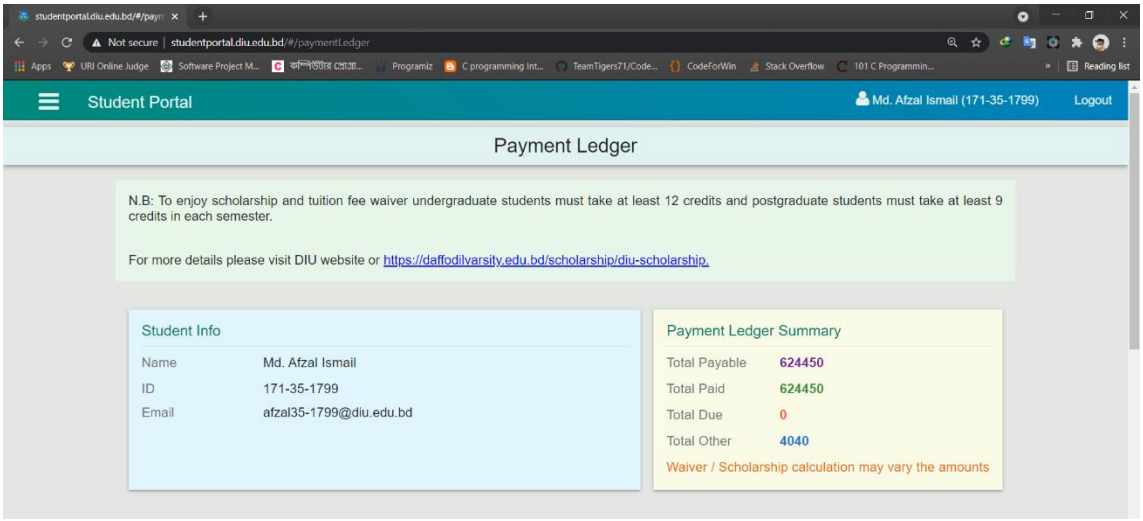
Soleimani, H., Hadavi, M. A., & Bagherdaei, A. (2017, September). WAVE: Black Box Detection of XSS, CSRF and Information Leakage Vulnerabilities. In 2017 14th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC) (pp. 19-24). IEEE.

Total Number of internet user worldwide | Datareportal. Retrieved May 14 from, https://datareportal.com/global-digital-overview

The CSRF detection cheat sheet | OWASP. Retrieved May 03 from, https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

Web Security CSRF token | Portswigger. Retrieved April 20 from, https://portswigger.net/web-security/csrf/tokens





Turnitin Originality Report

Processed on: 16-Jun-2021 17:02 +06
ID: 1607433520
Word Count: 4445
Submitted: 1

171-35-1799 By Md. Afzal Ismail

Similarity Index	Similarity by Source
12%	Internet Sources: 11% Publications: 4% Student Papers: 9%

2% match (student papers from 05-Apr-2018)
Class: Article 2018
Assignment: Journal Article
Paper ID: 941549238

2% match (Internet from 26-Aug-2020)
https://oregonsigmanu.com/online-watch-store/

1% match (Internet from 15-Mar-2020)
http://dspace.daffodilvarsity.edu.bd:8080/bitstream/handle/123456789/3802/P15037%20%2822_%29.pdf?isAllowed=y&sequence=1

1% match (Internet from 15-Mar-2020)
http://dspace.daffodilvarsity.edu.bd:8080/bitstream/handle/123456789/3553/P13659%20%2829%25%29.pdf?isAllowed=y&sequence=1

1% match (Internet from 23-Aug-2020)
http://dspace.daffodilvarsity.edu.bd:8080/bitstream/handle/123456789/3557/P13665%20%2815%25%29.pdf?isAllowed=y&sequence=1

1% match (Internet from 08-Mar-2020)
http://dspace.daffodilvarsity.edu.bd:8080/bitstream/handle/123456789/3808/P15042%20%2824_%29.pdf?isAllowed=y&sequence=1

1% match (student papers from 04-Apr-2018)
Class: Article 2018
Assignment: Journal Article
Paper ID: 940891790

1% match (Internet from 10-Jun-2021)
https://datareportal.com/global-digital-overview?rq=smartphone

< 1% match (student papers from 04-Apr-2012)
Submitted to University of Strathclyde on 2012-04-04

< 1% match (student papers from 22-Apr-2010)
Submitted to The Hong Kong Polytechnic University on 2010-04-22

< 1% match (student papers from 09-Apr-2021)
Submitted to Coventry University on 2021-04-09

< 1% match (Internet from 19-May-2020)
https://www.slideshare.net/hitesh0141/final-year-project-report-sample-for-engineers-iit-and-state-university

< 1% match (Internet from 14-Feb-2020)
https://link.springer.com/article/10.1007%2Fs13198-016-0489-0

< 1% match (Internet from 02-Nov-2020)