Design and Implementation of an Automated System for Detecting CSRF Vulnerabilities in Web Applications

Submitted By

Md. Ashiqur Rahman

ID: 211-35-3162

Department of Software Engineering

Daffodil International University

Supervised By

Mr. A.H.M Shahariar Parvez

Associate Professor

Department of Software Engineering

Daffodil International University

Bachelor of Science

DAFFODIL INTERNATIONAL UNIVERSITY

APPROVAL

This thesis titled on "**Design and Implementation of an Automated System for Detecting CSRF Vulnerabilities in Web Applications**", submitted by **Md. Ashiqur Rahman (ID: 211-35-3162)** 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

---------- **Chairman**

Dr. Imran Mahmud
Associate Professor & Head
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

---------- **Internal Examiner 1**

Nuruzzaman Faruqi
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

---------- **Internal Examiner 2**

Md. Rajib Mia
Lecturer (Senior Scale)
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

---------- **External Examiner**

Md. Fazle Munim
Associate Director & Vice President
Government & Public Sector
Ernst & young (EY)

SUPERVISOR'S DECLARATION

I/We* hereby declare that I/We* have checked this thesis/project* and in my/our* opinion, this thesis/project* is adequate in terms of scope and quality for the award of the degree of *Bachelor of Science/ Master of Science.

(Supervisor's Signature)

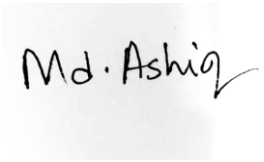
Full Name : Mr. A.H.M Shahariar Parvez

Position : Associate Professor

Date : 03 January 2025

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Daffodil International University or any other institution.



(Student's Signature)

Full Name : Md. Ashiqur Rahman

ID Number : 211-35-3162

Date : 03 January 2025

Design and Implementation of an Automated System for Detecting CSRF Vulnerabilities in Web Applications

Md. Ashiqur Rahman

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Science

Department of Software Engineering (Major in Cyber Security)

DAFFODIL INTERNATIONAL UNIVERSITY

January 2025

ACKNOWLEDGEMENTS

First and foremost, I want to express my gratitude to Almighty for allowing me to complete my senior year. I've gained a lot of knowledge throughout the years, which will undoubtedly help me in the present. For this, I want to thank all of our faculty members. I want to sincerely thank my supervisor, Mr. A.H.M. Shahariar Parvez, Associate Professor at DIU, for providing me with invaluable guidance and illuminating the required steps for me to complete my thesis on the subject of Automatic detection of web application vulnerabilities. In addition to my boss, I thank the Cyber Security Center at DIU for providing me with the assistance I needed.

DEDICATION

I dedicate this work to my mentors and instructors; thank you for mentoring me with your wisdom and expertise; this has been much helped in crafting my work. Finally, I dedicate my thesis to all future academics and students aiming to significantly contribute in the subject of online application security and cybersecurity. This work should motivate more creativity and commitment in striving greatness.

ABSTRACT

In the current technical world, there are extensive use of web applications. More confidential information is being handled by web apps. Sensitive data processing web applications face several risks. Hackers are constantly trying to come up with new ways to break into and take advantage of these applications. These attacks are carried out by exploiting vulnerabilities. It is one of the most critical threats to our web application and is known as cross-site request forgery (CSRF). Cross-Site Request Forgery (CSRF) The CSRF attack is an attack that forces end users to execute unwanted actions on web applications in which they are currently authenticated. Several past studies have identified many case reports. There are always models proposed and developed in any research. There is a lot of additional work that remains in this area to protect web applications that are potentially vulnerable to CSRF. Therefore, little research has been done on automated systems to detect this CSRF application layer attack. This work aims to design an automated model for the identification of CSRF threat in website applications. I recommended real time scan CSRF vulnerability in the given address.

KEYWORDS: Cybersecurity, CSRF vulnerability, Automated detection system, Web Application Vulnerability.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CSRF	Cross Site Request Forgery
OWASP	Open Web Application Security Project
SQL	Structured Query Language
XSS	Cross-site scripting

CHAPTER 1

INTRODUCTION

CSRF, or Cross-Site Request Forgery, is a kind of online application vulnerability. This indicates that the system is either inadequately protected or misconfigured if this vulnerability is present in any online applications. It allows an attacker to get confidential information, modify data, and execute state-altering activities, among other capabilities. To do this, an attacker deceives the user into sending requests from the user's own browser to the applications where the user is already authenticated. A successful CSRF attack poses significant security risks to both the end user and the website.

Web applications have become a fundamental part of business, organizations and consumer behaviour-based solutions over the years. According to Datareportal, there are now a total of 5.52 billion internet users worldwide as of October 1, 2024 the highest percentage which 67.5 percent of the world's total population. (*Digital Around the World — DataReportal – Global Digital Insights*, n.d.) Such a high number makes it so that internet users are now a "supermajority", with over twice as many people using the internet as those who don't. Internet users are still increasing, too, with the latest data showing that world's connected population grew by 151 million users in the 12 months to October 2024. (*Digital Around the World — DataReportal – Global Digital Insights*, n.d.)

As the web booms, web apps are becoming critical to communication, commerce, education and entertainment. As of 2024, there are 1.13 billion websites on the World Wide Web, according to a curate-labs survey. From online banking, e-commerce platforms, social media, and enterprise management system web applications, they

changed how individuals and organizations use technology. (*How Many Websites Are There In The World (2024)*, n.d.)

With the exponential rise of internet and web applications, there has also been a growing number of cybersecurity threats like XSS, SQL Injection, SSRF, broken authentication Etc. While there are many threats to web applications, CSRF is one of the most vulnerable attack vectors. CSRF stands for cross-site request forgery and is a web security vulnerability that enables an attacker to force users to execute unwanted actions on a different web application in which the user is authenticated. It lets an attacker partially break the same origin policy, which is supposed to prevent websites from messing with each other. (*What Is CSRF (Cross-Site Request Forgery)? Tutorial & Examples | Web Security Academy*, n.d.)

In the world of web application security, preventing CSRF is crucial as it can lead to significant data breaches, unauthorized actions, and loss of sensitive information. Despite improved security measures, CSRF vulnerabilities still plague many web applications Introduction CSRF is a well-known threat to modern web applications, as it allows malicious actors to perform unauthorized actions on behalf of legitimate users. The conventional approaches to recognizing and lessening CSRF adversities generally need manual intervention, making it a tedious procedure that can be easily overlooked.

1.1 Motivation of Research

With billions of user's access web applications for everything from shopping to banking and communication, their security is more relevant than ever. CSRF attacks are one of the most concerning vulnerabilities that allow an attacker to perform an action on a targeted site on behalf of a victim user. The Open Web Application Security Project, often referred to as OWASP, articulated this in relation to the viewpoint of combating CSRF. Targeting a user responsible for administrative tasks inside the program might result in significant harm and compromise the web application via a

CSRF attack. These types of attacks can result in the theft of data, unauthorized transactions, and compromised accounts or systems.(*Cross Site Request Forgery (CSRF) | OWASP Foundation*, n.d.)

Considering that many web applications do not properly implement CSRF preventive measures, this approach leaves users vulnerable to attacks. Current means of identifying these vulnerabilities tend to be manual, tedious and subject to human error. This led to the requirement of an automated system that can promptly and accurately detectable CSRF vulnerabilities, and assist developers in securing their applications and protecting users. This research team's goal is to combat this problem and make the world wide web a safer experience for us all.

1.2 Problem Statement

A thorough revisit and analysis of the earlier works on CSRF vulnerability detection revealed that the previously used methodologies either did not achieve automation, or did so but with very low efficiency. There is an opportunity for doing better in this part. Existing solutions face challenges like, but not limited to, reducing false positives or false negatives, which is crucial to ensure accuracy and reliability. Overcoming these constraints is an important avenue for continued research and development.

1.3 Research Questions

In light of this context, motive, and issue definition, I provide the following inquiries:

Question 1: Does my suggested solution identify Cross-Site Request Forgery vulnerabilities?

Question 2: Does my implemented tool correctly compare the false negative and false positive analyses with the current one?

1.4 Research Objective

1. To create an automated system for identifying Cross-Site Request Forgery (CSRF) vulnerabilities in web applications

1.5 Research Scope

Numerous web application kinds, such as banking websites, entertainment websites, blogs, retail websites, etc., have been the subject of experimental procedures. As a result, there are certain complexities with regard to cyber legislation and security problems. Because of this, We cannot conduct our experiment on every website available on the internet. In that instance, my buddies and I have experimented using a few sample websites.

1.6 Thesis Organization

APA reference system has been applied in this research on this paper. There are five chapters in the paper; these are detailed below:

Chapter 1: Here we address research background, motivation, issue definition, aims and scopes.

Chapter 2: This chapter addresses the current relevant works produced. also worked out the research void.

Chapter 3: This chapter delineates the research strategy and methodology.

Chapter 4: This chapter will address result analysis and discussion of the suggested model.

Chapter 5: This part will address the results, conclusion, and future direction of work.

CHAPTER 2

LITERATURE REVIEW

This section compares the existing relevant papers. It mainly centers on the mechanics of how those deployed technologies function, such as counteracting different kinds of attacks, tactics, and performance measures.

Cross-Site Request Forgery (CSRF) is one of the biggest threats to web applications due to the absence of adequate validation mechanisms. Ismail (2021) developed a framework to automate the identification of CSRF flaws in web applications. It builds up a rule-based model for tracing and flagging potential CSRF attack vector which makes vulnerability detection in easy and effective manner. This method gives developers actionable intelligence to fix security gaps proactively. Although the methodology is primarily built on rule-based detection, it does not provide the ability to prevent attacks in real-time and also cannot adapt to the fluctuations in attack patterns. (Md Afzal Ismail, n.d.)

Cross Site Request Forgery is a very common and very dangerous vulnerability that exploits authentication of users on web applications. This is often due to poor session management practices and hashing defences(Nadar et al., 2018). V. M. Nadar et al. It is a hybrid approach of both token-based methods, and added session management techniques as protection against CSRF. Their framework will also find Broken Authentication and Session Management vulnerabilities in the same environment as well. Their approach is more indeed successful with identification and mitigation of common attack vectors but does fails at well tested but anti-scalability attack techniques.(Nadar et al., 2018)

An increased availability of Cross-Site Request Forgery (CSRF) vulnerabilities due to insufficient API protection. API attacks are very common as API traffic is unencrypted. Shun Wang et al. (2022) proposed a detection and defence mechanism specific to API-level CSRF vulnerabilities. Using genetic algorithms, their system looked for requests that had been concatenated together and took measures to mitigate these attacks with behaviour modelling. Testing revealed high accuracy rates and low levels of false positives. Even though these proposed solutions work well at a conceptual level but there face practical challenges such as how should all this been integrated with legacy systems and how to scale out.(Wang et al., 2022)

Cross-Site Request Forgery (CSRF) has been one of the most prominent security issues, and the SameSite cookies have finally become a browser detection mitigation. Luca Compagna et al. (2021) is a preliminary study of the adoption and efficacy of SameSite cookies for CSRF; a follow-up study (2022) indicates the design that preferred SameSite cookies significantly dropped CSRF attack vector usage. Their findings reveal that while this mechanism is widely leveraged on many major platforms, it is not the case for smaller sites which suffer from low awareness of this mechanism and thus need to better understand it. The paper works out how the SameSite cookie mechanism is limited in itself by browser support and user settings that rely on user.(Compagna et al., 2021)

A significant addition is the implementation of SameSite cookies meant to mitigate Cross-Site Request Forgery (CSRF) attacks by restricting which sites can send cookies. Soheil Khodayari and Giancarlo Pellegrino (2022) analyzed the deployment, security and the sufficiency of SameSite cookies in their work. They also found improvements within large websites that haven't translated to the companies' implementations, however, and noted many misconfigurations that make these implementations ineffective. The SameSite policy has limitations, including inadequate protection against certain advanced attack methods, as noted by the same study. Their findings show insufficient documentation and better developer outreach to implement SameSite cookies correctly. (Khodayari & Pellegrino, n.d.)

However, CSRF, as the name implies, is a cross-site request forgery, which occurs when trust relationships are abused across a web application to perform actions on behalf of an user who has had their credentials compromised. T. Saleh et al. (2024) for developing a scenario-based simulation framework to model and analyze Cross-Site Request Forgery (CSRF) attack scenarios. Models provide insight into how good current defenses would actually be by permitting more detailed analysis of attack vectors and attack surfaces across various scenarios. In traditional testing the critical weaknesses tend to be ignored, but the simulation has surfaced those tapes. However, the framework has little focus on simulation and does not extend to providing automated or near-real-time mitigation approaches (Saleh et al., 2024).

After establishing CSRF defenses, you should follow it up with a few other steps to benefit enhanced security for modern applications. Xhelal Likaj et al. In their study, (2021) check the efficiency of CSRF defences for different Web frameworks. Moreover, the study found significant variation in the CSRF protection implementations, with some implementations being robust and enabled by default and others weak or completely disabled. Another contributing factor to this low quality phenomenon is ignorance of the developer and also lack of adequate documentation. These findings demonstrate the need for standardization and learning to raise CSRF security in web frameworks (Likaj et al., n.d.).

Cross-Site Request Forgery (CSRF) vulnerabilities are challenging because they have far-reaching effects that are hard to exploit in today's modern web applications. Giancarlo Pellegrino et al. (2017) introduced a tool called Deemon for detection of CSRF vulnerabilities based on dynamic analysis and property graphs. They are an approach that models the behavior of the application and finds discrepancies that indicate vulnerabilities. The system was not only highly accurate in detecting CSRF attacks, but also was highly adaptable to various applications. On the other The use, however, is based on the analysis of runtime which may be subject to performance overhead, making the application in large scale or in real time unfeasible (Pellegrino et al., 2017).

Cross Site Request Forgery (CSRF) — Web Apps Top 10 Security Issues. Giuseppe Beltrano et al. Proposed CSRF detection approach for code which might contain CSRF vulnerabilities (2023) using deep learning-based detection system. This involves utilizing neural networks to analyze web traffic and spot anomalies that could indicate CSRF attacks. Detected new attack types with very good accuracy and adaptability. The limitations of the study also did not mention the challenges on training the model from diverse datasets to achieve generalizability and scalability for real-world applications (Beltrano et al., 2023).

Cross-Site Request Forgery (CSRF) CSRF attacks exploit inadequate input validation in online applications. Chang Liu et al. (2020) were the pioneers in proposing a CSRF detection approach via graph data mining techniques. The method uses graphs that may be created from interactions with the online application to identify unusual patterns that may suggest cross-site request forgery. The technique effectively identified sophisticated attack scenarios that beyond the capabilities of feature-based detection inherent in conventional rule-based systems. Nevertheless, dependence on the precise configuration of the graph poses difficulties in the realm of dynamic or large-scale applications (Liu et al., 2020).

Cross-Site Request Forgery (CSRF) vulnerabilities are still hard to find in black-box testing cases. In 2019, Stefano Calzavara et al. presented Mitch, a system that uses machine learning to enhance CSRF vulnerability identification. They propose to exploit supervised learning in order to review the behavior of web applications and recognize the pattern of the attack. The model reaches high performance in the identification of the vulnerabilities, while the source is not given access to the code, which allows this tool to be spread in various spheres. On the other hand, the research uncovers plus sides regarding simple content handling but also points out drawbacks such as the need for elaborate training data for its sustainability (Calzavara et al., 2019).

CSRF attacks trick users to execute malicious commands on their online accounts. P. Yadav and C. D. In 2017, D. Parekh published a paper outlining techniques to prevent CSRF security problems. The researchers emphasized the importance of token-based

technique, SameSite cookies, server-side validation in preventing CSRF vulnerabilities. They also stress that assigning the responsibility for secure coding to developers would help prevent errors from the beginning. Nonetheless, the report points out that advanced, automated detection techniques have not been implemented greatly in legacy systems.(Yadav & Parekh, 2017)

Cross-Site Request Forgery attacks exploit the trust of a user's browser to carry out malicious fraud. Lalia and Moustafa (2019) proposed a web browser extension to mitigate CSRF attacks. Their solution stops and checks requests before they get to the server, adding extra protection on the client-side. This method can avoid CSRF vulnerabilities without requiring any changes to the web application's backend. Using browser extensions may not be a good idea as it may depend on user uptake and browser compatibility (Lalia & Moustafa, 2019).

CHAPTER 3

METHODOLOGY

It will make use of a CSRF Detection System to assess the vulnerability of web applications to CSRF attacks, by inspecting their defense techniques. This system works by pulling HTML forms from a URL, looking at CSRF tokens in the forms, and comparing token properties (the presence of a CSRF token, length of a CSRF token, and if a CSRF token is unique or not) to determine if a web application is sufficiently protected. It also tests cookies using Selenium for the existence and configuration of the SameSite attribute, which provides an additional layer of defense against CSRF attacks.

This approach applies to both static and dynamic analyses, utilizing libraries like requests, BeautifulSoup, and Selenium for extraction and assessment. It generates a security assessment calling a site "safe" or "vulnerable" based on token and cookie security implementation, giving actionable insights to developers and security professionals.

3.1 Tools and Libraries

To guarantee that web app security analysis is efficient and thorough, the CSRF Detection System uses a variety of tools and libraries. Using Python as the main programming language allows for easy integration with requests library for grabbing the HTML page and with BeautifulSoup to parse and extract forms and fields. Selenium (for browser automation) and webdriver-manager (to simplify the browser driver initialization) are used to inspect dynamic website elements (like cookies with SameSite values). This can be done using a headless Chrome browser, which allows you to perform these actions without needing a graphical user interface (GUI) to open. Pattern matching, heuristics and machine learning technique are used together to let

the system detect CSRF vulnerabilities automatically without compromising accuracy when it adaption multiple web environment.

3.2 System Architecture

Figure 3.2.1 illustrates the system architecture of my suggested solution using a workflow diagram.

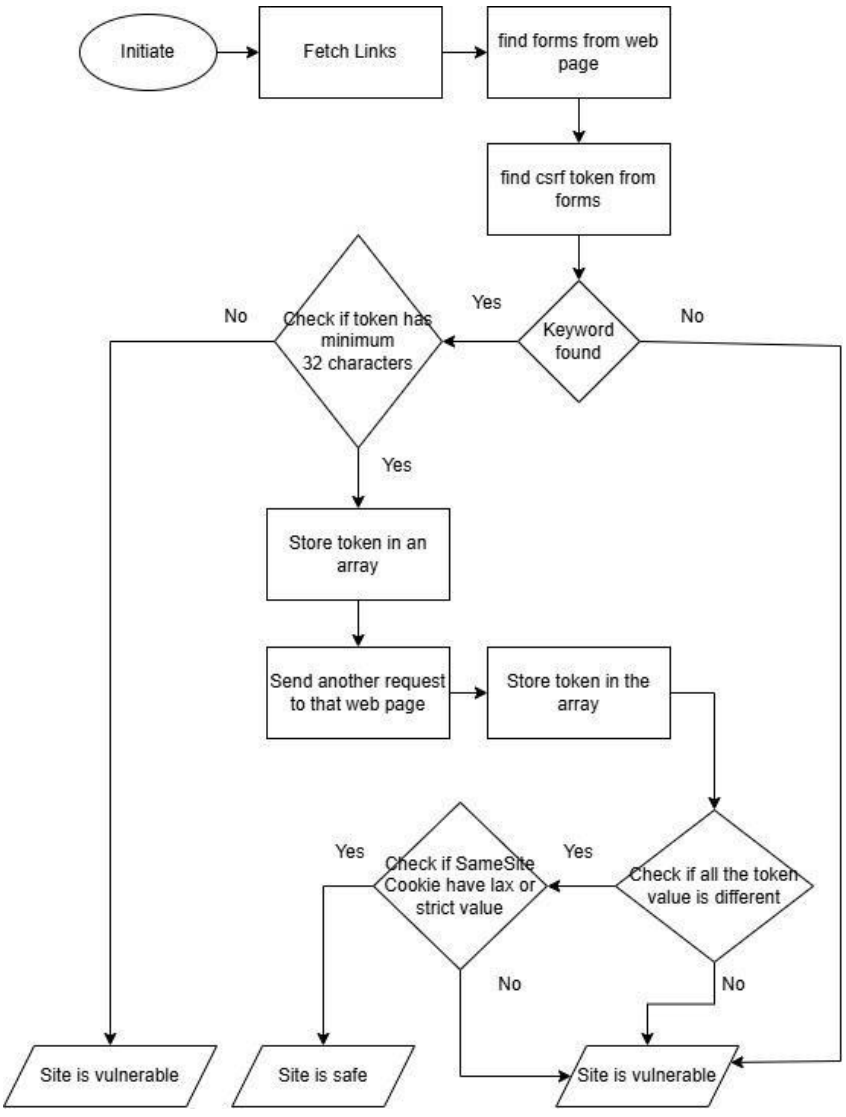


Figure 3.2.1: System Architecture of CSRF Detection System

The system is divided into seven modules:

- Input Module
- Link Extractor
- Form Extractor
- CSRF Token Analysis
- Cookie Security Module
- Security Evaluation Module
- Output Module

3.2.1 Input Module

The Input Module is the entry point of the system, which lets the user input the URL of the website they want to evaluate. This module is the first step, collects the input from the user and passes this input to other modules for continued processing.

3.2.2 Link Extraction Module

The Link Extractor module extract and validate hyperlinks from a URL of the webpage. It uses the requests library for fetching the web page contents as well as BeautifulSoup for the parsing of HTML, so it is able to extract all the <a> tag link efficiently. The module also provides a validation method via the urlparse utility that filters invalid links or malformed URLs, and uses urljoin to convert relative URLs to absolute URLs. This helps only collect working and valid hyperlinks. The design of the module focused on modularization and accuracy, allowing it to be incorporated into larger applications or academic work.

3.2.3 Form Extraction Module

It fetches out all HTML forms from the pipeline given by users. This module utilizes the requests library to request the raw HTML content of the page and uses BeautifulSoup to parse and search for elements in the page structure. These extracted <form> tags and forward to the next stage for parsing. The outcome of this module's requirement for all data relevant to a form to be passed to it lays the groundwork for CSRF token processing.

3.2.4 CSRF Token Analysis Module

The following module studies the forms which were fetched in the previous phase to check and verify the CSRF tokens. It checks the <input> fields within the forms for names or attributes that suggest the presence of CSRF tokens such as "csrf" or "token." When a token has been detected, it verifies the token's validity by confirming its satisfies minimum length constraints (the token is at least 32 characters long). The module also determines whether the tokens are unique across requests to identify potential token reuse. If this module performs these checks, it is going to tell you if the application actually implemented its basic CSRF protections appropriately.

3.2.5 Cookie Security Module

This Cookie Security Module, as its name suggests, is designed to analyze the security of the cookies being used by the website. It uses Selenium WebDriver to emulate an actual browser experience by loading cookies from the website and examining their properties. In particular, it examines whether the cookies have the SameSite attribute set to Lax or Strict, both key tools in blocking CSRF attacks Cookies lacking these attributes are identified as insecure, and the system incorporates cookie-level vulnerabilities into the evaluation process in addition to form and token vulnerabilities.

3.2.6 Security Evaluation Module

The Security Evaluation Module combines the findings from the CSRF Token Analysis and Cookie Security modules to provide a comprehensive evaluation of the website's overall security against CSRF attacks. By integrating the results of token checks and cookie analyses, this module determines whether the application is adequately protected or if vulnerabilities exist. Its decision logic considers missing or invalid tokens, token reuse, and insecure cookie attributes to provide a holistic security assessment.

3.2.7 Output Module

At last, the output module presents to the user easily understandable assessment findings. It offers a synopsis of the security situation of the website together with indications of whether or not it is vulnerable to CSRF attacks. Should vulnerabilities be found, the module offers thorough comments pointing out particular flaws as inadequate cookie settings, missing tokens, or poor token strength. This guarantees users' ability to properly grasp and solve the found problems.

3.3 Algorithm and Implementation

This section delineates the primary algorithm of our CSRF detection system. Figure 3.3.1 illustrates the algorithm of our approach.

```
Input URL
Fetch webpage content using requests
Parse HTML content using BeautifulSoup
Extract all forms from the webpage

Initialize an empty list for tokens

For each form in forms:
    Search for input fields with names containing "csrf" or "token"
    If token found:
        Add token to the list
    Else:
        Mark site as vulnerable and exit
    Check if token length is at least 32 characters
    If not:
        Mark site as vulnerable and exit

Fetch the webpage content again
Extract forms and tokens again
For each token in the new extraction:
    Add to the token list
If token list contains duplicates:
    Mark site as vulnerable and exit

Launch headless browser with Selenium
Navigate to the URL
Retrieve cookies
For each cookie:
    If SameSite attribute is missing or not Lax/Strict:
        Mark site as vulnerable and exit

Output "Site is safe" or detailed vulnerability report
```

Figure 3.3.1: Algorithm

This method guarantees a methodical and comprehensive assessment of the security policies of the website by using automation to effectively find any CSRF vulnerabilities and therefore provide the user with unambiguous, practical findings. The program starts by getting a URL as input and locating its webpage content. It examines every form for

CSRF tokens in input fields and breaks down the HTML to extract all forms. Should no token be located or if the token is too short—less than 32 characters—the site is considered vulnerable. The method then does a second pass looking for token reusing by pulling tokens from the forms once more and comparing them for originality. Should any duplicate tokens surface, the site is declared as vulnerable. The method then gets Selenium to visit the page and gather cookies. It looks at if the SameSite property is present and appropriately set to Lax or Strict. Should any cookies lack this quality, the website becomes insecure. At last, the program produces either a thorough report of the vulnerabilities found or indicates if the site is safe. This stage guarantees an all-around assessment of the CSRF defenses of the website.

3.4 System Limitations

The CSRF Detection System has some limitations that may affect its applicability in certain scenarios.

Here, three major conditions have encountered.

1. The system extensively uses static analysis of HTML forms; hence it is not always able to detect dynamically produced forms or tokens being handled entirely by JavaScript.
2. CSRF tokens is limited to forms and fails to recognize other mechanisms, such as header-based token validation.
3. The system can’t not identify header origin or referrer.

Additionally, the reliance on Selenium for cookie inspection may not be suitable for websites with advanced anti-bot measures. These constraints highlight areas for potential improvement in future iterations of the system.

3.5 Test suite setup

I chose 20 online apps to examine. I have compiled a list of some of the most popular online apps utilized by consumers, as well as my own site for work. Table 3.5.1 shows the names and kinds of the applications. While developing our program, we utilized a standard machine running 64-bit Windows 10. The specification of system is 3.6 GHz, Ryzen 5, 16GB RAM.

Table 3.5.1: Specifications of Test Applications

Application	**Type**
ShoeShop	My Own Project
JUC	Educational Institute
ESJI	Journal
YTS	Entertainment
Jotform	Development
Wow Express	Courier
PeerTube	Video Sharing Platform
BD Specialized Hospital	Hospital
VS Hospital	Hospital
Medicare Spots	Clinic
Le Reve	Clothing Store
Vaidam	Hospital
Businessinspection	Business
Paperfly	Courier
EB medicine	Medical help
ooredoo	E-Commerce
Telenor	Telecommunication
Infomaniak	Website builder
Gaming Beasts	Entertainment
Thepcgames	Entertainment

CHAPTER 4

RESULTS AND DISCUSSION

We addressed the results in this section. The experiment on the built CSRF Detection system will be evaluated and discussed.

4.1 Discussion

The outcomes of our experiment are shown in Table 4.1.1. The table has two primary columns: column one denotes the application name, while column two indicates the outcomes of our experiment. Column 2 is partitioned into three segments.

The first segment is titled “Successful.” In this section, we will evaluate if our automated technique correctly detects the intended outcomes. The second portion, under “False Positive,” will examine if our program inaccurately identifies any non-susceptible applications as vulnerable. The third portion, under “False Negative,” examines whether our method inaccurately identifies any sensitive sites as non-vulnerable.

Table 4.1.1: Efficiency of CSRF detection system

Application	Detection		
	Successful	**False Positive**	**False Negative**
ShoeShop	✓		
JUC	✓		
ESJI	✓		
YTS			✓
Jotform	✓		
Wow Express	✓		
PeerTube	✓		
BD Specialized Hospital	✓		
VS Hospital	✓		
Medicare Spots	✓		
Le Reve	✓		
Vaidam	✓		
Businessinspection	✓		
Paperfly	✓		
EB medicine	✓		
ooredoo		✓	
Telenor	✓		
Infomaniak	✓		
Gaming Beasts	✓		
Thepcgames	✓		

4.2 Analysis

The operations and possibilities of web apps are often limited according to the kind of user and their authentication. For instance, there may be a situation in which a user of a web application is unable to access some services without being signed in. Therefore, due to appropriate authentication, we are unable to conduct experiments on all types of web apps in our system. Consequently, the objective is to analyze the registration or login forms. Additionally, look through the contact-us and other forms that are available. The idea is justified by the fact that tokens and SameSite cookies/attribute analysis are the foundation of our system. These tokens must be included in every form when a request is made.

We conducted our experiment based on this idea. Twenty online apps in all have been used, and the outcome is good. Table 2 shows that our technology has effectively identified vulnerabilities in 18 of these 20 web apps. Additionally, the false positive rate is low. Both the total number and the number of false positives is 1. The experiment outcomes suggest that the web application is expected to be secure if our system indicates it is not vulnerable. Furthermore, it is highly likely to be susceptible if it displays a response indicating that the web application is vulnerable; nevertheless, we must be mindful that the answer might also be a false positive or false negative in other situations.

4.3 Performance

Our suggested CSRF detection method correctly identifies 18 out of 20 web apps. Figure 4.3.1 illustrates the efficacy of our investigation.

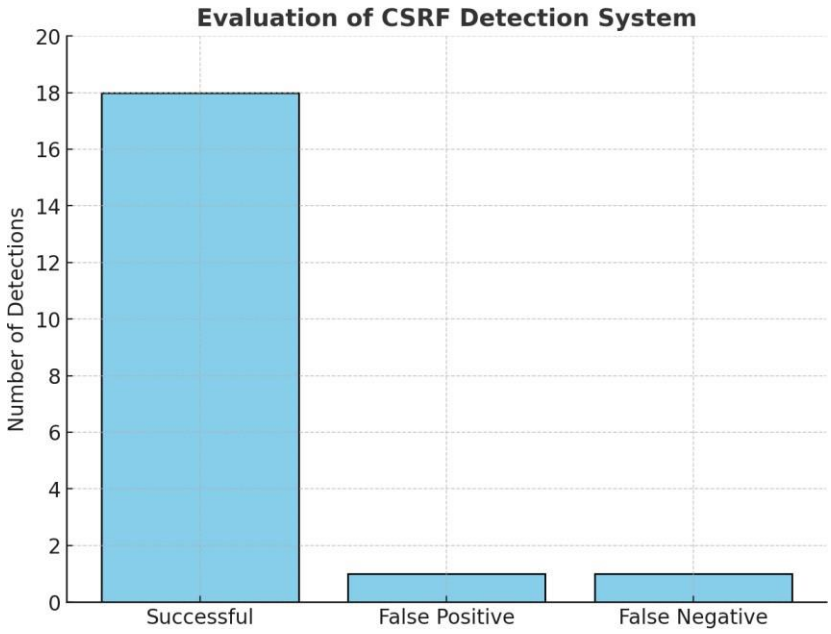


Figure 4.3.1: Potency of CSRF Detection System

With a success rate of 90% (i.e., 18 out of 20 detections), the bar chart demonstrates that the CSRF detection system was evaluated. Two detections, one of which was a false positive and the other a false negative, each contributed 5% and total is 10%. This proves that the system is quite precise, with almost no detection mistakes.

CHAPTER 5

CONCLUSION

5.1 Findings and Contribution

A tool has been built to detect the Cross-site Request Forgery vulnerability. The devised method utilizes a web scraper to collect data and detect flaws in the online form. This tool is crucial for identifying Cross-site Request Forgery vulnerabilities. This instrument has an efficacy rate of 90%. The tool's adequacy is good for evaluating various options.

5.2 Recommendations for Future Works

The future aim is to build a model to improve the CSRF vulnerability detection system's detecting capacity. Furthermore, the wish is to include machine learning methods into this model to raise its flexibility and accuracy. Combining machine learning with this project aims to provide a more intelligent and effective solution competent of managing challenging online environments. More robust, accurate, and user-friendly integration of this will help the system to be adopted more widely and used practically in online security.

REFERENCES

Beltrano, G., Greco, C., Ianni, M., & Fortino, G. (2023). Deep Learning-Based Detection of CSRF Vulnerabilities in Web Applications. *2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, 0916–0920. https://doi.org/10.1109/DASC/PiCom/CBDCom/Cy59711.2023.10361414

Calzavara, S., Conti, M., Focardi, R., Rabitti, A., & Tolomei, G. (2019). Mitch: A machine learning approach to the black-box detection of CSRF vulnerabilities. *Proceedings - 4th IEEE European Symposium on Security and Privacy, EURO S and P 2019*, 528–543. https://doi.org/10.1109/EUROSP.2019.00045

Compagna, L., Jonker, H., Krochewski, J., Krumnow, B., & Sahin, M. (2021). A preliminary study on the adoption and effectiveness of SameSite cookies as a CSRF defence. *Proceedings - 2021 IEEE European Symposium on Security and Privacy Workshops, Euro S and PW 2021*, 49–59. https://doi.org/10.1109/EUROSPW54576.2021.00012

Cross Site Request Forgery (CSRF) | OWASP Foundation. (n.d.). Retrieved January 2, 2025, from https://owasp.org/www-community/attacks/csrf

Digital Around the World — DataReportal – Global Digital Insights. (n.d.). Retrieved January 2, 2025, from https://datareportal.com/global-digital-overview

How Many Websites Are There In The World (2024). (n.d.). Retrieved January 2, 2025, from https://curatelabs.co/how-many-websites-are-there/

Khodayari, S., & Pellegrino, G. (n.d.). *The State of the SameSite: Studying the Usage, Effectiveness, and Adequacy of SameSite Cookies*.

Lalia, S., & Moustafa, K. (2019). Implementation of Web Browser Extension for Mitigating CSRF Attack. *Advances in Intelligent Systems and Computing*, *931*, 867–880. https://doi.org/10.1007/978-3-030-16184-2_82

Likaj, X., Khodayari, S., & Pellegrino, G. (n.d.). *Where We Stand (or Fall): An Analysis of CSRF Defenses in Web Frameworks; Where We Stand (or Fall): An Analysis of CSRF Defenses in Web Frameworks*. *21*. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Liu, C., Shen, X., Gao, M., & Dai, W. (2020). CSRF Detection Based on Graph Data Mining. *Proceedings of 2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education, ICISCAE 2020*, 475–480. https://doi.org/10.1109/ICISCAE51034.2020.9236806

Md Afzal Ismail, B. (n.d.). *An Automated Detection System of Cross Site Request Forgery (CSRF) Vulnerability in Web Applications*.

Nadar, V. M., Chatterjee, M., & Jacob, L. (2018). A defensive approach for CSRF and broken authentication and session management attack. *Advances in Intelligent Systems and Computing*, *696*, 577–588. https://doi.org/10.1007/978-981-10-7386-1_49

Pellegrino, G., Johns, M., Koch, S., Backes, M., & Rossow, C. (2017). Deemon: Detecting CSRF with Dynamic Analysis and Property Graphs. *Proceedings of the ACM Conference on Computer and Communications Security*, 1757–1771. https://doi.org/10.1145/3133956.3133959

Saleh, T., Malkawi, M., Elgammal, Z., Calayır, A. K., & Alhajj, R. (2024). Scenario-Based Cross-Site Request Forgery (CSRF) Attack Simulation. *2024 6th International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, 1–5. https://doi.org/10.1109/ISAECT64333.2024.10799863

Wang, S., Ni, C., Wang, J., & Nie, C. (2022). Detecting and Defending CSRF at API-Level. *Proceedings - 2022 IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW 2022*, 75–80. https://doi.org/10.1109/ISSREW55968.2022.00043

What is CSRF (Cross-site request forgery)? Tutorial & Examples | Web Security Academy. (n.d.). Retrieved January 2, 2025, from https://portswigger.net/web-security/csrf

Yadav, P., & Parekh, C. D. (2017). A report on CSRF security challenges & prevention techniques. *Proceedings of 2017 International Conference on Innovations in Information, Embedded and Communication Systems, ICIIECS 2017*, *2018-January*, 1–4. https://doi.org/10.1109/ICIIECS.2017.8275852

Design and Implementation of an Automated System for Detecting CSRF Vulnerabilities in Web Applications .V2

by Ashiqur Rahman

Submission date: 15-Jan-2025 01:08AM (UTC+0600)
Submission ID: 2564179810
File name: Md_Ashiqur_Rahman-211-35-3162_A-34_Thesis.pdf (552.99K)
Word count: 6025
Character count: 33829

Design and Implementation of an Automated System for Detecting CSRF Vulnerabilities in Web Applications .V2

ORIGINALITY REPORT

22%	19%	10%	15%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dspace.daffodilvarsity.edu.bd:8080 Internet Source	6%
2	Submitted to Midlands State University Student Paper	2%
3	Submitted to University of Bedfordshire Student Paper	1%
4	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	1%
5	Submitted to University of Glasgow Student Paper	1%
6	Submitted to Charles Sturt University Student Paper	1%
7	Submitted to National College of Ireland Student Paper	1%
8	Submitted to Hellenic Open University Student Paper	1%
	umpir.ump.edu.my	

9	Internet Source	1%
10	Submitted to Colorado Technical University Student Paper	1%
11	Submitted to University of Gloucestershire Student Paper	<1%
12	h4ck3r5.code.blog Internet Source	<1%
13	Submitted to Ghana Technology University College Student Paper	<1%
14	Submitted to Manchester Metropolitan University Student Paper	<1%
15	Submitted to University of Warwick Student Paper	<1%
16	docslib.org Internet Source	<1%
17	www.open.ou.nl Internet Source	<1%
18	ijrpr.com Internet Source	<1%
19	www.fastly.com Internet Source	<1%

20	Rodriguez, Martin. "Computer Self-Efficacy: A Critical Mediator Between Task-Specific Self-Efficacy and Task Performance in Advanced Use of Common It Applications.", Trident University International, 2024 Publication	<1%
21	Ronghao Pan, Antonio Ruiz-Martínez. "Evolution of web tracking protection in Chrome", Journal of Information Security and Applications, 2023 Publication	<1%
22	datareportal.com Internet Source	<1%
23	www.lnwebworks.com Internet Source	<1%
24	Nivedita Singh, Yejin Do, Yongsang Yu, Imane Fouad, Jungrae Kim, Hyoungshick Kim. "Crumbled Cookies: Exploring E-commerce Websites? Cookie Policies with Data Protection Regulations", ACM Transactions on the Web, 2024 Publication	<1%
25	ouci.dntb.gov.ua Internet Source	<1%
26	portswigger.net Internet Source	<1%

27	wave1994.tistory.com Internet Source	<1%
28	Submitted to Liberty University Student Paper	<1%
29	Mano Paul. "Official (ISC) Guide to the ", CRC Press, 2013 Publication	<1%
30	Submitted to Sheffield Hallam University Student Paper	<1%
31	9lib.co Internet Source	<1%
32	ijisrt.com Internet Source	<1%
33	mbfagun.blogspot.com Internet Source	<1%
34	Submitted to Coventry University Student Paper	<1%
35	ntnuopen.ntnu.no Internet Source	<1%
36	online-journals.org Internet Source	<1%
37	repository.aust.edu.ng Internet Source	<1%
38	www.geeksforgeeks.org	

	Internet Source	<1%
39	www.microsoft.com Internet Source	<1%
40	Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Artificial Intelligence, Blockchain, Computing and Security", CRC Press, 2023 Publication	<1%
41	Chang, Yu-Hao. "Design and Implementation of IoTtalk AA Subsystem and Account Subsystem.", National Yang Ming Chiao Tung University, 2024 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

Design and Implementation of an Automated System for Detecting CSRF Vulnerabilities in Web Applications .V2

GRADEMARK REPORT

FINAL GRADE

/0

GENERAL COMMENTS

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39