PREVENTION OF CROSS-SITE REQUEST FORGERY USING ANTI-CSRF TOKEN

PHYU PHYU WIN

M.C.Sc. **SEPTEMBER 2022**

PREVENTION OF CROSS-SITE REQUEST FORGERY USING ANTI-CSRF TOKEN

BY

Phyu Phyu Win

B.C.Sc.

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Computer Science

(M.C.Sc.)

University of Computer Studies, Yangon

SETEMBER 2022

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude and sincere appreciation to the following persons who have contributed directly or indirectly towards the completion of this thesis and helped me make this dissertation possible.

Secondly, I would like to express my special thanks to **Dr. Mie Mie Khin**, the Rector of the University of Computer Studies, Yangon who gave me the opportunity to develop this thesis and for her general guidance during the period of study.

And I would also like to thank **Dr. Mie Mie Thet Thwin**, the former Rector of the University of Computer Studies, Yangon who gave me the opportunity to develop this thesis and for her general guidance during the period of study.

I would like to express my respectful gratitude to **Dr. Si Si Mar Win,** Processor and **Dr. Tin Zar Thaw**, Processor, Course Coordinators of the Master 24th batch, University of Computer Studies, Yangon, for their excellent guidance.

I would like to thank the former Course Coordinator, **Dr. Mie Mie Su Thwin** who gave me the opportunity to do this thesis on the topic, "Prevention of Cross-Site Request Forgery using Anti-csrf token", and also gave invaluable recommendations regarding to this thesis.

I would like to express my sincere gratitude to my former supervisor, **Dr. Nyein Myint Myint Aung**, Associate Professor of Faculty of Information Science, University of Computer Studies, Yangon for her encouragement and support.

My sincere thanks and respect go to my supervisor, **Dr. Yi Mon Thet**, Associate Professor of Faculty of Information Science, University of Computer Studies, Yangon, for her invaluable advice, giving me detailed guidance, support and comments throughout the thesis process.

I also wish to express my deepest gratitude to **U Aung Myint Thain**, Assistant Lecturer, English Department, the University of Computer Studies, Yangon, for his editing this thesis from the language point of view.

Last but not least, I am extremely thankful to my parents and my family for supporting, inspiring and encouragement to me from the childhood to the present time. Finally, I am extremely grateful to all my teachers, my colleagues and all of my friends for their invaluable and precious help and general guidance.

STATEMENT OF ORIGINALITY

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

------------------------------------	---------------------------------
Date	Phyu Phyu Win

ABSTRACT

Online banking system has created an enormous impact on IT, Individuals, and networking worlds. Online banking systems and its exclusive architecture have numerous features and advantages over traditional banking system. The proposed system detects the csrf-attack with two types of web application, sign in with token and sign in without token. In the system, detection rate illustrates with percentage(%). Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. CSRF attacks specifically target state-changing requests, not theft of data. This attacks target functionality that causes a state change on the server such as changing the victim's email address, password or purchasing something. In the system, the attacker creates a malicious link and sends to the website. The main objectives of the proposed system is to provide the data security of the customer's critical transmission data, to protect for state changing functionalities on critical data processing between the client and server, to illustrate the secure transaction and record transaction history, to prevent the attack using the anti-csrf token when making transactions in banking system. The proposed system illustrates the secure transaction in banking system and provides the data security of the customer's critical transmission data. The proposed system in this thesis is implemented to prevent the CSRF attack. The Blum Blum Shub algorithm is used to generate the Anti-csrf token. The token is a secret, unique and unpredictable value a server-side application generates in order to protect CSRF vulnerable resources. The tokens are generated and submitted by the server-side application and SHA-256 hash is used when sending to the client site. After the request is made, the server aspect utility compares the two tokens found in the user consultation and inside the request. If the token is not match from the received transaction form , the request is rejected.

Keywords: CSRF Prevention, Anti-csrf Token, Blum Blum Shub Algorithm, HMAC (SHA-256) Algorithm

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

		Page
OWASP	Open Web Application Security Project	1
CSRF	Cross-Site Request Forgery	1
CRLF	Carriage Return and Line Feed	5
LDAP	Light Weight Directory Access Protocol	5
IDOR	Insecure Direct Object Reference	5
XSS	Cross-Site Scripting	6
LCG	Linear Congruential Generator	16
HMAC	Keyed-Hashing for Message Authentication Code	17
MD5	Message Digest Hash	18
SHA-1	Secure Hash Algorithm 1	20
BBS	Blum Blum Shub	26
SHA-256	Secure Hash Algorithm 256-bit	27

CHAPTER 1
INTRODUCTION

Banking is popular today and uses to deposit/withdraw the cash. This is the place where customers feel the sense of safety for their property. This is why banking becomes an important role in lives. In the act of using web applications, people give personal information to the organization, and then store sensitive information on them. On the other hand, some attackers who are unethical and selfish exploit the web application to gain unauthorized access and do other things such as identity theft, privacy violation, and other cyber-attacks. These illegal points allow the attackers to make whatever they want through the weaknesses of the web application.

Vulnerability is the weak point of the web application caused by unawareness of the developers who cannot be handled validation the user inputs, appropriate validation methods, and so on. Because of those facts, detection of vulnerability is needed more. There are so many different kinds of vulnerabilities but, it is indicated to OWASP in 2019 that CSRF attack reaches number eight vulnerabilities. Cross-site request forgery (CSRF) is a web security vulnerability that allows an attacker to induce users to perform actions that they do not intend to perform. In a successful CSRF attack, the attacker causes the victim user to carry out an action unintentionally. This might be to change the email address on their account, to change their password, or to make a funds transfer.

1.1 Related Work

In this thesis, the system intends to support the admins who require to obtain secure transaction without vulnerabilities and to prevent CSRF attacks from the attacker. The system uses anti-csrf token and also has generated by a random number generator. This section discusses the previous studies of preventing CSRF vulnerabilities concerning needed to protect against.

The first study describing the implementation of cross site request forgery method using tools is implemented by Sentamilselvan. K Assistant [12]. Their experiment provides suitable solutions for the cross site request forgery attack by means of applying parsing techniques to identify the attacking spots before the attackers attack. It takes a long time and it requires no additional memory.

In 2018, Sami Azam introduced the ‘Preventive Measures for Cross Site Request Forgery Attacks on Web-based Applications’ that identify the available solutions to prevent CSRF attacks [4]. By analyzing the techniques employed in each of the solutions, the optimal tool can be identified. Tests against the exploitation of the vulnerabilities were conducted after implementing the solutions into the web application to check the efficacy of each of the solutions. The research also propose a combined solution that integrates the passing of an unpredictable token through a hidden field and validating it on the server side with the passing of token through URL.

In 2008, Adam Barth and Collin Jackson examined the scope and diversity of CSRF vulnerabilities, studied existing defenses, and described incremental and new defenses based on headers and web application firewall rules [1]. We introduce login cross-site request forgery attacks, which are currently widely possible, damaging, and under-appreciated. There are three widely used techniques for defending against CSRF attacks: validating a secret request token, validating the HTTP Referrer header, and validating custom headers attached to XML Http Requests.

In the other study, Emil Semastin implemented to identify the available solutions to prevent CSRF attacks using tools Pinata, CSRF tester, Burp Suite and OWASP ZAP. Tests against the exploitation of the vulnerabilities were conducted after implementing the solutions into the web application to check the efficacy of each of the solutions. The suggested solution is a combination of the most effective existing technique and the second best option. By implementing this, a double validation takes place at the server side of the web application to ensure the prevention of CSRF attacks.

1.2 Objectives of the Thesis

The main objectives of the thesis are as follows:

- to provide the data security of the customer’s critical transmission data
- to protect for state changing functionalities on critical data processing between the client and server

- to illustrate the secure transaction and record transaction history
- to prevent the attack using the anti-csrf token when making transactions in banking system

1.3 Organization of the Thesis

This thesis consists of five chapters.

Chapter 1 is the introductory section in which the introduction to web application vulnerability. And the related works, the objectives, and the organization of the thesis are presented.

Chapter 2 describes the background theory related to this thesis such as OWASP Top Ten attacks, preventive measure for CSRF attack, and Blum Blum Shub algorithm that are described in details.

Chapter 3 presents the design of the proposed system describing system flow, the detailed explanation with algorithms and the evaluation of the output resulting from the post detection.

Chapter 4 describes the implementation of the proposed system in detail and the experimental result.

Finally, Chapter 5 concludes this thesis which its benefits, limitation and further extension of the proposed system.

CHAPTER 2
BACKGROUND THEORY

In this chapter, the related background theory about the research this is presented. In the first section, the Open Web Application Security Project (OWASP) top ten attacks and the types of the attack are described. The next section describes about the preventive measures of the attacks. In the last section, types of pseudo number generator and HMAC with hash algorithm are presented in details.

2.1 Open Web Application Security Project (OWASP) Top 10 Attacks

The Open Web Application Security Project, or OWASP, is a worldwide no income organization dedicated to web application safety [8]. The OWASP Top ten is a regularly-updated report outlining security concerns for web application security, focusing on the ten most critical risks.

2.1.1 Injection

Injection happens when an attacker exploits insecure code to insert (or inject) their personal code into a software. Because the program is unable to determine code inserted in this way from its own code, attackers are able to use injection attacks to access secure areas and confidential information as though they are trusted users. Examples of injection include SQL injections, command injections, CRLF injections, and LDAP injections.

- **SQL injection:** SQL injection is the position of malicious code in statements, through the web page input. In the case of advanced SQL Injection attacks, the attacker can use SQL commands to write arbitrary files to the server and even execute OS commands. This may lead to full system compromise.
- **Command injection:** The attacker injects operating system commands with the privileges of the user who is running the web application. In advanced cases, the attacker may exploit additional privilege escalation vulnerabilities, which may lead to full system compromise.

- **CRLF infusion:** The attacker injects an unexpected CRLF character collection. This sequence is used to split an HTTP response header and write arbitrary contents to the response body. This attack may be combined with Cross-site Scripting (XSS).
- **LDAP infusion:** The attacker injects LDAP (Lightweight Directory Access Protocol) statements to execute arbitrary LDAP commands. They can gain permissions and modify the contents of the LDAP tree.
- **Header Injection in Email:** CRLF injections are very similar to this attack. The intruder sends IMAP/SMTP commands to a mail server that cannot be accessed directly through a web application.
- **Injection of the host header:** The attacker poisons web caches and password-rest functionality by taking advantage of the HTTP Host header's implicit trust.
- **Injection of OS commands:** With the permissions of the web application's user, the attacker injects operating system commands. In more advanced scenarios, the attackers may take advantage of additional privilege escalation flaws, which could result in the complete system compromise.
- **Injection of XPath:** In order to carry out crafted XPath queries, the intruder injects data into an application. They can use them to get into data that isn't theirs and get around authentication.

2.1.2 Broken Authentication and Session Management

Broken authentication and session management vulnerabilities is an OWASP indexed vulnerability that acknowledges the risk of credentials due to bad identity and access controls implementation. Exploiting a broken authentication, an attack is commonly initiated by means of taking gain of poorly managed credentials and login classes to masquerade as authenticated customers.

Attackers use automated tools to retrieve additional data and take control of the application after manually spotting holes in user validation and verification in session management vulnerabilities. Since attackers are always looking for ways to gain access by taking advantage of security implementation flaws, authentication and session management are essential components of modern application security frameworks.

Due to the complex and integrated nature of modern applications, scanning for authentication and session management vulnerabilities can be difficult. Broken authentication and session management vulnerabilities are discussed in this post, along with recommended procedures and tools for implementing them safely.

A session is a series of events and transactions that happen during the same time period for the same user. A unique Session ID (Cookies, URL Parameters, Authentication Tokens, etc.) is given to each user upon logging in to a system. This ID enables communication between the user and the web app during a valid session. It is easier for a hacker to take over the session ID and gain unauthorized system access because many developers fail to develop the appropriate session parameters. Additionally, attackers can impersonate users who are already logged in to the system because some developers fail to set session time limits and rotation plans.

2.1.3 Cross-Site Scripting (XSS)

Cross-Webpage Prearranging (XSS) assaults are a sort of infusion, wherein noxious contents are infused into in any case harmless and confided in sites. XSS assaults happen when an aggressor utilizes a web application to send malevolent code, by and large as program side content, to an alternate end client. Blemishes that permit these assaults to succeed are very inescapable and happen anyplace a web application utilizes input from a client inside the result it produces without approving or encoding it.

An aggressor can utilize sending a malevolent content to a clueless client. The end client's program has no real way to realize that the content ought not be relied upon, and will execute the content. Since it thinks the content came from a believed supply, the vindictive content can get passage to any treats, meeting tokens, or other delicate data held through the program and utilized with that site on the web.

The term "stored" refers to attacks, in which the injected script is permanently stored on the target servers, such as in a message forum, visitor log, comment field, or database. When the victim requests the stored data, the malicious script is retrieved from the server. Persistent or Type-I XSS are other names for Stored XSS.

Persistent XSS includes blind cross-site scripting as one type. It usually happens when the attacker's payload is saved on the server and sent back to the victim by the backend application. An attacker can, for instance, use feedback forms to submit a malicious payload. Once the backend user or administrator of the application opens the submitted form via the backend application, the attacker's payload will be executed. XSS Hunter is one of the best tools for confirming blind cross-site scripting in real-world situations.

When the injected script is reflected off the web server in an error message, search result, or other response that includes some or all of the input sent to the server as part of the request, these attacks are referred to as "reflected attacks." Victims of reflected attacks receive the information via a different means, such as an email or a different website. The injected code travels to the vulnerable website, which reflects the attack back to the user's browser, when the user is tricked into clicking on a malicious link, submitting a specially crafted form, or even just browsing to a malicious website. The code is then executed by the browser due to its origin from a "trusted" server. Non-Persistent or Type-II XSS are other names for Reflected XSS.

2.1.4 Insecure Direct Object Reference

Unreliable direct item references (IDOR) are a network protection issue that happens when a web application designer utilizes an identifier for direct admittance to an internal execution object anyway gives no extra access control as well as approval checks. For instance, IDOR weakness would occur in the event that the URL of an exchange could be changed through client-side client contribution to show unapproved information of another exchange. Shaky direct item happens the designers use reference objects in URL. The aggressor can change the worth in reference protests and can see other data and afterward can do the catalog crossing attack.

2.1.5 Security Misconfiguration

Security misconfigurations are security controls which can be erroneously designed or left unreliable, putting the designs and measurements at possibility. Fundamentally, any ineffectively reported setup changes, default settings, or a specialized issue across any variable on endpoints might need to cause a misconfiguration.

Misconfiguration weaknesses are arrangement shortcomings that could exist in programming subsystems or parts. For example, web server programming could send with default client accounts that a cybercriminal could use to get to the framework, or the product could have a known arrangement of standard setup documents or catalogs, which a cybercriminal could take advantage of.

Assuming weaknesses are the doorway to the local area, it's the misconfigurations that assailants influence to vindictive program their way to the planned targets. Finding them is a needle in the bundle, as they can be situated across any part in an association's frameworks, like its servers, working frameworks, applications, and programs. Absence of deceivability and incorporated means to remediate misconfigurations makes associations succumb to misconfiguration assaults.

Present day local area foundations are particularly muddled and portrayed by utilizing ordinary change; associations can without issues disregard fundamental security settings, which incorporates new organization gadget that could keep up with default setups. Regardless of whether provision secure designs to endpoints, reviewing arrangements and wellbeing controls consistently to see the inescapable setup stream. Frameworks exchange, new framework is brought into the organization; patches are executed all adding to misconfigurations.

2.1.6 Sensitive Data Exposure

Delicate information is any data that is intended to be shielded from unapproved access. Delicate information can incorporate anything from actually recognizable data (PII, for example, Federal retirement aide numbers, to banking data, to login qualifications. At the point when this information is gotten to by an assailant because of information break, clients are in danger for delicate information openness.

Any time an association needs security strategies, information is in danger of openness. To improve methodologies of relief on potential application attacks, advancement and security groups should initially have a solid handle on the manners in which that information is inclined to openness including:

- **Information on the way:** Information on the way is profoundly defenseless, particularly while getting across unprotected channels or to the application programming point of interaction (Programming interface) that permits applications to speak with each other. One assault that objectives information on the way is a man-in-the-center assault, which captures traffic and screens correspondences.
- **Information very still:** is housed in a framework, be it a PC or organization. It is believed to be less powerless without the danger of assaults in passing, yet all at once more important. Aggressors utilize various vectors to get tightly to house information, frequently utilizing malware like diversions or PC worms. Both of these get entrance into frameworks lodging information through direct downloading from a vindictive USB drive or by clicking pernicious connections that are emailed or text. On the off chance that information is housed in a server, assailants could get tightly to data put away in records beyond the typical verified areas of access.

2.1.7 Missing Function Level Access Control

The missing capability level gain passage to influence weakness allows in clients to perform capacities that should be confined, or allows them to get to resources that should be incorporated. Typically, capabilities and assets are straightforwardly safeguarded in the code or by design settings, yet it's not generally simple to accurately do. Assailants who suspect that capabilities or assets are not as expected safeguarded should initially get sufficiently close to the framework they need to assault. To take advantage of this weakness, they should have consent to send genuine Programming interface calls to the endpoint.

OWASP gives an illustration of this weakness of an enrollment interaction set up to permit new clients to join a site. It would presumably utilize a Programming interface GET call, similar to this:

GET/programming interface/welcomes/{invite_guid}

The noxious client would get back a JSON with insights regarding the welcome, including the client's job and email. They could then change GET to POST and furthermore hoist their welcome from a client to an administrator utilizing the accompanying Programming interface call:

POST/programming interface/welcomes/new

{"email":"shadyguy@targetedsystem.com","role":"admin"}

Just administrators ought to have the option to send POST orders, however on the off chance that they are not as expected got, the Programming interface will acknowledge them as authentic and execute anything the aggressor needs.

2.1.8 Cross-Site Request Forgery

Cross-Site Request Forgery (CSRF) is an assault that powers an end client to execute undesirable activities on a web application in which they're presently verified [9]. With a little assistance of social designing, (for example, emailing a connection or talk), an assailant might deceive the clients of a web application into executing activities of the aggressor's picking. In the event that the casualty is an ordinary client, an effective CSRF assault can compel the client to perform state changing solicitations like moving assets, changing their email address, etc.

CSRF attacks target usefulness that causes a state change on the server, for example, changing the casualty's email address or secret phrase, or buying something. Driving the casualty to recover data doesn't acquire an assailant on the grounds that the aggressor doesn't get the reaction, the casualty does. Accordingly, CSRF attacks target state-evolving demands.

There are different methodologies in which an end client might be fooled into stacking data from or submitting data to a web utility. To execute an attack, the initial comprehend how to produce a legitimate malevolent solicitation for our casualty to execute. Allow this to think about the accompanying model: Alice wishes to move $100 to Sway utilizing the bank.com web application that is defenseless against

CSRF. Maria, an assailant, wants to fool Alice into sending the cash to Maria all things considered. The attack will include the accompanying advances.

In the event that the application was intended to basically utilize GET solicitations to move boundaries and execute activities, the cash move activity may be diminished to a solicitation like:

GET http://bank.com/transfer.do?acct=BOB&amount=100 HTTP/1.1

Maria currently settles on a decision to exploit this web application weakness the utilization of Alice on the grounds that the person in question. Maria first develops the accompanying make the most URL which will switch $100,000 from Alice's record to Maria's record. Maria takes the first order URL and replaces the recipient name with herself, raising the exchange sum altogether simultaneously:

http://bank.com/transfer.do?acct=MARIA&amount=100000

The social designing part of the assault fools Alice into stacking this URL when Alice is signed into the bank application. This is generally finished with one of the accompanying procedures:

- sending a spontaneous email with HTML content
- establishing an adventure URL or content on pages that are most likely to be visited through the casualty while they're moreover doing internet banking.

The endeavor URL can be veiled as a normal connection, empowering the casualty to click it:

<a href=http://bank.com/transfer.do?acct=MARIA&amount=100000">VIEW My Photos! </a>

Or on the other hand as a 0x0 phony picture:

<img src="http://bank.com/transfer.do?acct=MARIA & amount=100000" width="0" height="0" border="0">

Assuming this picture tag was remembered for the email, Alice wouldn't see anything. In any case, the program will in any case present the solicitation to bank.com with practically no visual sign that the exchange has occurred.

The bank present utilizes post and the weak solicitation seems this way:

```
POST http://bank.com/transfer.do HTTP/1.1 acct=BOB & amount=100
```

Such a solicitation can't be conveyed utilizing standard an or IMG labels yet can be conveyed utilizing a Structure labels:

```
<form action="http://bank.com/transfer.do method=POST">
<input type="hidden" name=acct" value=MARIA"/>
<input type="hidden" name="amount" value=100000"/>
<input type="submit" value="View My Photos"/>
</form>
```

This structure will require the client to tap on the submit button, yet this can be additionally executed naturally utilizing JavaScript:

```
<body onload="document.forms [0].submit ()">
<structure…..>
```

2.1.9 Using Components With Know Vulnerabilities

This specific weakness can carry enormous gamble to the business particularly due to its simplicity of exploitability. On the off chance that the aggressor can figure out the weak parts which a specific application is utilizing, it tends to be handily taken advantage of since the endeavor techniques are now out there in the web and the aggressor basically needs to utilize it and can cause a negligible effect, or serious or even total information split the difference, or lead to server/have takeover for associations.

This weakness can undoubtedly sidestep the application security safeguards and can likewise go about as a turning point to empower different assaults for instance programmers might summon a web administration with full consent without giving an approval token or direct a remote code execution. The shortcoming while at the same time utilizing weak parts incorporate infusion, XSS and broken admittance control.

Developers must consider the consequences of using dependencies and be fully aware of all of the dependencies they use. Additionally, all dependencies ought to be entered into an inventory system that can provide a straightforward overview of all the dependencies being utilized. Although developers should keep in mind which automatic actions are carried out, it is best to perform all of these actions automatically.

In order to avoid being overlooked, these scans should be performed on a regular basis, preferably automatically. Because of how the customer interacts with the web application or program, it is best to perform these scans using an external system. Additionally, this will guarantee that no other servers will be slowed down by the resources required for these scans.

2.1.10 Un-validated Redirects and Forward

Nullified Diverts and forward Weakness, likewise occasionally known as URL Redirection Weakness, is a kind of pernicious program found inside the web application. In this kind of weakness, the aggressor uses to control the URL and sends it to the person in question. When the casualty opens the URL, the site diverts it to a pernicious site or site to which the aggressor believes that the client should get diverted.

The aggressor regularly uses to exploit this type of Weakness with the help of manual control in the URL or with the assistance of a few devices like Burp suite, which provides an assailant with a few sorts of approaches in light of which he can control the URL to get Diverted.

2.2 Preventive Measure for CSRF

The system describes many CSRF prevention mechanisms. This includes Using a Secret Cookie, Only accepting POST request, Multi step transaction, Checking Referrer Header, Anti csrf-token. In this system, the anti-csrf token is used to prevent the csrf attack.

2.2.1 Using a Secret Cookie

The severe worth will keep the treat from being dispatched via the program to the objective site in all pass-site riding setting, regardless of whether following an ordinary hyperlink. A monetary organization site yet would have no desire to permit any conditional pages to be connected from outside destinations, so the severe banner would be generally reasonable.

The default remiss worth gives a sensible harmony among security and ease of use for sites that need to keep up with client's signed in meeting after the client shows up from an outside connect. The meeting treat would be permitted while following an ordinary connection from an outer site while obstructing it in CSRF-inclined demand techniques like POST. Just cross-site-demands that are permitted in careless mode are the ones that have high level routes and are likewise protected HTTP techniques.

Illustration of treats utilizing this characteristic:

Set-Treat: CookieName=CookieValue; SameSite=Lax;

Set-Treat: CookieName=CookieValue; SameSite=Strict;

Assuming the worth is going to Severe, it moves toward that any solicitation beginning from an outsider site to your site might have all treats wiped out by means of the program. It ‘smiles the most solid putting and empowers in forestalling untrusted lawful solicitations from being delivered.

Setting the worth to Remiss doesn't eliminate the treats for any GET demands. This gives a consistent encounter to your client when they follow joins from different destinations to your site.

2.2.2 Only accepting POST request

Applications can be created to just acknowledge POST demands for the execution of business rationale. The misinterpretation is that since the aggressor can't develop a malevolent connection, a CSRF attack can't be executed. There are various procedures where in an assailant can fool a victim into recording a manufactured distribute demand, alongside a simple shape facilitated in an assailant's site with stowed away qualities. This structure can be set off naturally by JavaScript or can be set off by the casualty who figures the structure will accomplish something different.

2.2.3 Multi step transaction

Multi-Step exchanges are certainly not a sufficient counteraction of CSRF. Inasmuch as an aggressor can are expecting or derive each step of the finished exchange, then CSRF is conceivable.

2.2.4 Checking Referrer Header

The Referrer header is an old header that contains the URL the client came from. In the event that you click on a connection, the URL of the ongoing page is sent in the Referrer header to the mentioned connects. At the end of the day, this could be utilized to figure out where the client came from, which can assist us with hindering cross-site demands. Notwithstanding, there are two issues with the Referrer header.

In the first place, the Referrer header is ineffectively determined. It isn't determined on which demands the header ought to be sent, or regardless of whether it ought to be sent by any means. Despite the fact that most programs in all actuality do send this header, there is no particular that says they ought to.

Also, the Referrer header releases the entire URL to different areas. Assuming the URL contains delicate information, for example, the meeting token or some other identifier that is spilled when the URL is sent in the Referrer header when the client clicks a connection. This is the explanation that numerous enemy of infection arrangements take the Referrer header from all HTTP demands, to try not to release delicate information in the URL. Since so many enemy of infection arrangements strip the header, we can't depend on the Referrer header to be available.

2.2.5 Anti csrf-token

A CSRF Token is confidential, one of a kind and unusual worth a server-side application produces to safeguard CSRF weak assets [2]. The tokens are created and presented by utilizing the server-side application. After the solicitation is made, the server side application analyzes the two tokens situated inside the individual meeting and inside the solicitation. On the off chance that the token is missing or doesn't match the worth inside the client meeting, the solicitation is dismissed, the client meeting ended and the occasion logged as a potential CSRF assault.

Tokens is used to prevent attackers from sending requests through a victim are anti-CSRF.A pair of cryptographically related anti-CSRF tokens that a user receives to validate his requests. For instance, when a user sends a request to the webserver for a form-filled page, the server calculates two cryptographically related tokens and sends them to the user as a response. The Set-Cookie header of the response contains the other token, which is sent as a hidden field in the form. These two tokens are sent back to the server when the user submits the form: one in a cookie and one as a GET/POST parameter (which is sent to the user as a hidden form field).Following that, the server checks these two tokens for forgery or malformation. The server validates the request and performs the appropriate function if the tokens match the cryptographic mechanism; otherwise, the server returns an error.

CSRF token is produced utilizing a cryptographic strength pseudo-irregular number generator (PRNG), cultivated with the timestamp when it was made and a static mystery. The token submits to the client inside a secret field of the client submit structure. The symbolic will then be incorporated as a solicitation boundary when the structure is submitted:

<input type="hidden" name="csrf-token" value="CIwNZNlR4XbisJF39I8 yWnWX9wX4WFoz"/>

2.3 Types of Pseudorandom Number Generator

A pseudorandom number generator (PRNG) also known as a deterministic random bit generator, is an algorithm for generating a sequence of numbers whose properties approximate the properties of sequences of random numbers.

2.3.1 Tausworth Generator

Tausworth Generator is a sort of pseudorandom number generator, which produces irregular pieces.

The following equation-

$$X_{n+1} = (A_1x_n+A_2x_{n-1}+\text{---}+A_kx_{n-k+1}) \bmod 2 \qquad (2.1)$$

Where,

Xi, Ai {0, 1} for all i.

Since TG just delivers bits, it is too delayed to possibly be helpful. A strategy to accelerate is to utilize a unique structure called three fold based TG.

Using XOR operation

$$I_i=I_{i-250} \text{ XOR } I_{i-147} \qquad (2.2)$$

Where XOR shows a bitwise selective or activity. It likewise has an extremely lengthy inside express (the last 250 whole numbers). Subsequently the cycle length is extremely lengthy.

2.3.2 Linear Congruential Generator (LCG)

Direct Congruential Generator (LCG) creates long irregular line of number with the grouping rehashing eventually. The irregular line of significant worth not entirely settled by a proper number called a seed.

$$X_{n+1} = (aX_n+b) \bmod m \qquad (2.3)$$

Where X is the sequence of pseudo-random values, and

m =modulus and $m>0$

a =the mutiplier and $0 < a < m$

c =the increment and $0 < b < m$

X_0 =the starting seed value and $0 \leq x0 < m$

The degree arbitrary numbers produced is not exactly the scope of the number utilized in the computation. The produced arbitrary numbers xi are supposed to be occasional where the period is in every case less $\leq m$ and all xi are in the stretch $0 \leq xi < m$

A LCG with enormous enough state can finish even tough factual assessments; a modulo-2 LCG which returns the high 32 pieces passes. An ideal arbitrary number generator with 32 pieces of result is supposed to start copying before yields. Any PRNG whose result is its full, shortened state won't create copies until its full period passes, an effectively perceptible factual defect.

2.4 HMAC using Hash Function

HMAC stands for (keyed-hash message authentication code or hash-based message authentication code) is a specific type of message authentication code (MAC) involving a cryptographic hash function and a secret cryptographic key[5]. As with any MAC, it can be used to simultaneously verify both the information integrity and authenticity of a message. Any cryptographic hash function, such as SHA-1 or SHA-3, may be used in the calculation of an HMAC; the resulting MAC algorithm is termed HMAC-X, where X is the hash function used (e.g. HMAC-SHA256). The cryptographic strength of the HMAC depends upon the cryptographic strength of the underlying hash function, the size of its hash output, and the size and quality of the key. HMAC does not encrypt the message. Instead, the message (encrypted or not) must be sent alongside the HMAC hash. Parties with the secret key will hash the message again themselves, and if it is authentic, the received and computed hashes will match.

$$\mathrm{HMAC}(K, m) = H((K' \oplus \text{opad}) \parallel H((K' \oplus \text{ipad}) \parallel m)) \qquad (2.4)$$

$K' = H(K)$ K is a larger than block size, K otherwise

Where

H is a cryptographic hash function

m is the message to be authenticated

K is the secret key K' is a block-sized key derived from the secret key,

$\parallel$ denotes concatenation

$\oplus$ denotes bitwise exclusive or (XOR)

opad is the block-sized outer padding

ipad is the block-sized inner padding

2.4.1 MD5 Hash

MD5 is a cryptographic hash capability calculation that accepts the message as contribution of any length and changes it into a fixed-length message of 16 bytes. MD5 calculation represents the message-digest set of rules. The result of MD5 (Condensation length) is consistently 128 pieces. MD5 became advanced in 1991 through Ronald Rivest. MD5 creates a similar hash capability for various data sources. MD5 gives unfortunate security over SHA1. MD5 has been viewed as an uncertain calculation. So presently utilizing of SHA256 rather than MD5.

```
Algorithm: MD5 Hashing Algorithm
Input = 5bit array variable
Output = character
Begin
    Initialize variables:
    Append 1 bit to message
    Append 0 bit until message length in bits = 448 (mod 512)
    Append original length in bits mod $2^{64}$ to message
    For each 512-bit chunk of padded message do
         Break chunk into sixteen 32-bit words M[j], $0 \le j \le 15$
    For i from 0 to 63 do
        var int F, g
           If $0\le i \le 15$ then
              F: = (B and C) or ((not B) and D)
              g: = i
           Else if $16 \le i \le 31$ then
              F: = (D and B) or ((not D) and C)
              g: = (5×I + 1) mod 16
           Else if $32 \le i \le 47$ then
               F: = B xor C xor D
               g: = (3×i + 5) mod 16
           Else if $48 \le i \le 63$ then
               F: = C xor (B or (not D))
               g: = ( 7×i ) mod 16
           F: = F + A + K[i] +M[g], A: = D, D: = C, C: = B, B:= B + leftrotate ( F, s[i] )
  End for
           Adding hash, a0:= a0 + A, b0:= b0 + B, c0:= c0 + C, d0:= d0 + D
  End for
           Var char digest [16]:= a0 append b0 append c0 append d0

End
```

Figure 2.1 MD5 Hashing Algorithm

2.4.2 SHA-1 Hash Algorithm

SHA-1 is the abbreviation for Secure Hash Algorithm 1, utilized for hashing information and declaration documents. Each piece of information delivers a novel hash that is completely non-duplicable by some other piece of information. SHA-1 works by taking care of a message as a piece string delivering a 160-piece hash esteem known as a message digest. SHA-1 (Secure Hash Calculation 1) is a cryptographically broken yet at the same time broadly utilized hash capability which takes info and produces a 160-bit (20-byte) hash esteem known as a message digest. It was planned by the US Public safety Office, and is a U.S. Government Data Handling Standard.

```
Algorithm: SHA-1 Hashing Algorithm
Input = an array 5 items long
Output = hash code
Begin
   Initialize variables:
   Append the bit '1' to the message.
   Append 0 ≤ k < 512 bits '0', such that the resulting message length in bits.
   Append ml, the original message length in bits, as a 64-bit big-endian integer.
   Break message into 512-bit chunks
        for each chunk
           break chunk into sixteen 32-bit big-endian words w[i], 0 ≤ i ≤ 15
         for i from 16 to 79
            w[i] = (w [i-3] xor w [i-8] xor w[i-14] xor w[i-16]) leftrotate 1
          for i from 0 to 79
             if 0 ≤ i ≤ 19 then
                f = (b and c) or ((not b) and d)
                k = 0x5A827999
          else if 20 ≤ i ≤ 39
                f = b xor c xor d
                k = 0x6ED9EBA1
          else if 40 ≤ i ≤ 59
                 f = (b and c) or (b and d) or (c and d)
                 k = 0x8F1BBCDC
          else if 60 ≤ i ≤ 79
                 f = b xor c xor d
                 k = 0xCA62C1D6
           temp = (a leftrotate 5) + f + e + k + w[i]
           e = d
           d = c
           c = b leftrotate 30
           b = a
           a = temp
End for
End for
  output hash = (h0 leftshift 128) or (h1 leftshift 96) or (h2 leftshift 64) or (h3 leftshift 32) or h4
End for
End
```

Figure 2.2 SHA-1 Hash Algorithm

CHAPTER 3
DESIGN OF THE PROPOSED SYSTEM

The main goal of this thesis is to prevent Cross-Site Request Forgery (CSRF) attack using Anti-csrf Token. The Blum Blum Shub algorithm is used to generate the token in the server site. Firstly, the overview of the proposed system of system architecture is described. And each of the algorithms that take part in the main program is described in a detailed explanation. This chapter mainly focuses on the design of the system.

3.1 Overview of the Proposed System

The proposed system consists of the algorithm of proposed system, system flow and CSRF attack flow. The expected architecture of the system is shown in Figure (3.1).

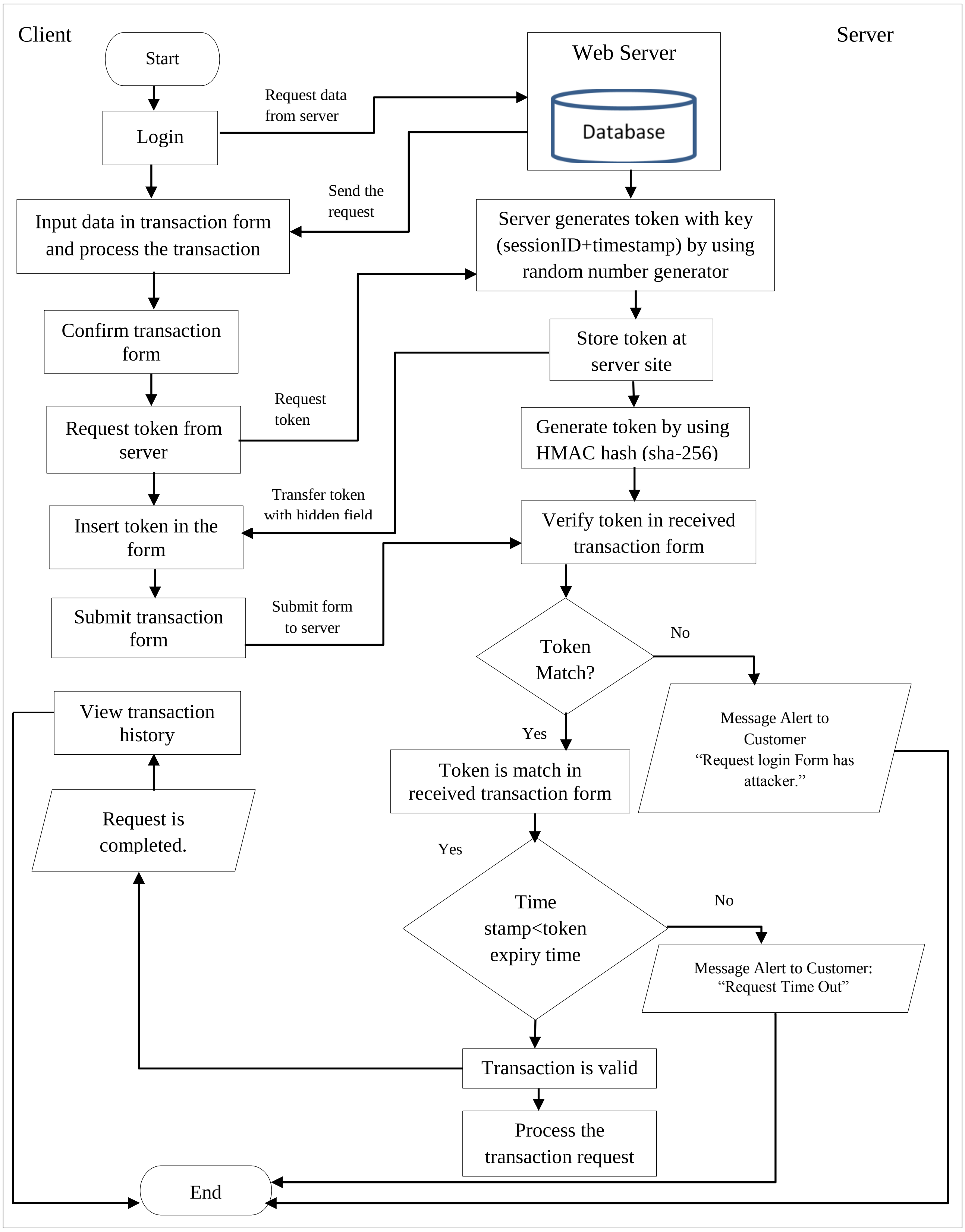

Client
Server
Start
Login
Request data from server
Web Server
Database
Send the request
Input data in transaction form and process the transaction
Server generates token with key (sessionID+timestamp) by using random number generator
Confirm transaction form
Store token at server site
Request token
Request token from server
Generate token by using HMAC hash (sha-256)
Transfer token with hidden field
Insert token in the form
Verify token in received transaction form
Submit transaction form
Submit form to server
Token Match?
No
Message Alert to Customer "Request login Form has attacker."
View transaction history
Yes
Token is match in received transaction form
Request is completed.
Yes
Time stamp<token expiry time
No
Message Alert to Customer: "Request Time Out"
Transaction is valid
Process the transaction request
End

Figure 3.1 System Flow Diagram of the Proposed System

In figure (3.2), the system shows the creation of the csrf attack. The attacker creates the malicious request sending to web application for a fund transfer. The attacker embeds the request into hyperlink and sending to the website. Website validates request and transfers funds from the client’s account to the attacker.

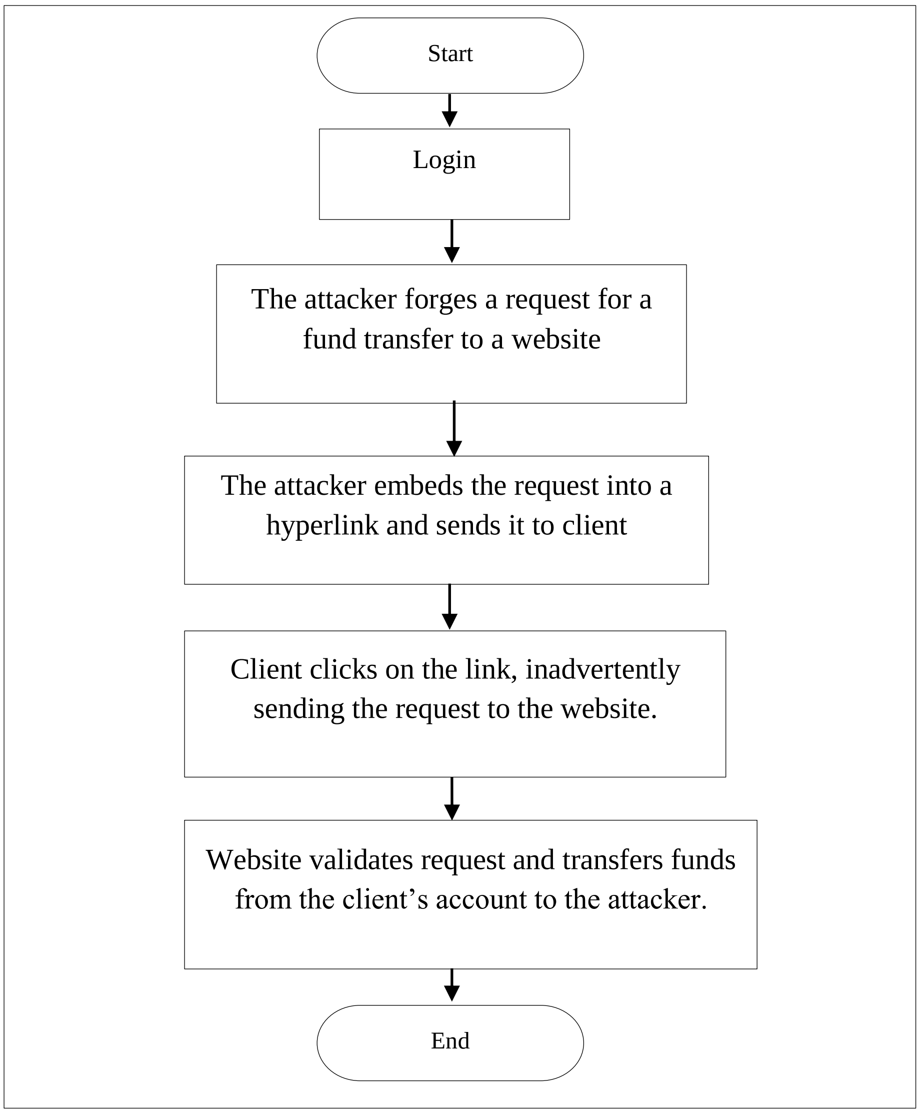

Start
Login
The attacker forges a request for a fund transfer to a website
The attacker embeds the request into a hyperlink and sends it to client
Client clicks on the link, inadvertently sending the request to the website.
Website validates request and transfers funds from the client’s account to the attacker.
End

Figure 3.2 CSRF Attack System Flow

```
Algorithm of Proposed System

Input = banking URL, attack link

Output= secure transaction or attack

Begin

  Step 1: Create a CSRF Token
        1.1 Start a session on the server.
        1.2 Generate a user session ID(using a random number generator)
        1.3 Keep Token Expire Time
        1.4 Generate a CSRF Token using key k
              1.4.1    Generate HMAC (user session ID+timestamp)
  Step 2: Include the token in the form (i.e. HMAC+timestamp)
        2.1 Inject the token into the hidden field of the user submit form
  Step 3: Validate the token
        3.1 Regenerated the token with the same key k (parameter are session ID from the
         request and timestamp in the received token.
        3.2 If ("If the HMAC in the token and the one generated in this step match") {
              If (Timestamp received is less than token expire time) {
               Request is treated as legitimate and can be allowed ;}
               Else {Request Time Out ;}
         End If
            Else if ("If the HMAC in the token and the one generated in this step not
match")
            {Reject the Process ;}
End If
End If
End
```

Figure 3.3 Algorithm of Proposed System

3.2 Blum Blum Shub Generator

The Blum Blum Shub (BBS) generator is perhaps the earliest and most popular cryptographically secure pseudo-arbitrary piece generators. The Blum Shub is a pseudorandom number generator proposed in 1986 by LenoreBlum, Manuel Blum and Michael Shub. Blum Shub takes the structure-

$$x_{i+1} = x_n{}^2 \mathrm{mod}\ M \qquad (3.1)$$

Where M = pq is the result of two enormous primes p and q. At each step of the calculation, some result is gotten from xn+1; the result is normally the piece equality of xn+1 or at least one of the most un-critical pieces of xn+1. The seed x0 ought to be a number that is co-prime to M (for example p and q are not elements of x0) and not 1 or 0. The two primes, p and q, ought to both be compatible to 3 (mod 4).

The generator BBS fills in as follow:

```
Algorithm: Blum Blum Shub Algorithm
Input = two prime number
Output = random sequence
Begin
  Compute n = pq .
  Select a random integer 0 <S < n (the seed) such that gcd (S, n) =1
  Compute $y = S^2 \bmod n$
  For i from 1 to N do the following:
      $y_i = y_{i-1}{}^2 \bmod n$
      $x_i = y_i \bmod 2$ the least significant bit of $y_i$
      The output sequence is $x_1, x_2, \ldots, x_i$ .
End
```

Figure 3.4 Blum Blum Shub Algorithm

3.3 SHA-256 Hash Algorithm

SHA-256 represents Secure Hash Algorithm 256-digit and it's utilized for cryptographic security. A hash isn't 'encryption' - it can't be decoded back to the first text. It is remarkably difficult to reproduce the underlying information from the hash esteem. To break a hash to want every one of the 64 of the digits to coordinate. It would require a long investment to break a SHA-256 hash utilizing all the whole organization.

```
Algorithm: SHA-256 hashing Algorithm
Input = an array 8 items long
Output = hash values
  Begin
    Initialize hash values- The compression function uses 8 working variables, a through h
    Initialize array of round constants- k [0..63] := [428a2f98, 71374491, b5c0fbcf, e9b5dba5,
3956c25b, 59f111f1,     923f82a4,…]
    Pre-processing (Padding) - begin with the original message of length L bits
                             - append a single '1' bit
                             - append K '0' bits, where K is the minimum number
    Expanded message blocks   $w_0$, w1,…, w63
      First 16 words w [0...15] of the message schedule
          $w_i=m^{(j)}_i$   for i=0,1,…,15, and
      For i=16 to 63
          σ0 := (w[i-15] rightrotate 7) xor (w[i-15] rightrotate 18) xor (w[i-15] rightshift 3)
          σ1 := (w[i- 2] rightrotate 17) xor (w[i- 2] rightrotate 19) xor (w[i- 2] rightshift 10)
          w[i] := w[i-16] + σ0 + w[i-7] + σ1
    Initialize working variables to current hash value
      for i from 0 to 63
           s1 := (e rightrotate 6) xor (e rightrotate 11) xor (e rightrotate 25)
           ch: = (e and f) xor ((not e) and g)
           temp1:= h + s1 + ch + k[i] + w[i]
           s0:= (a rightrotate 2) xor (a rightrotate 13) xor (a rightrotate 22)
           maj := (a and b) xor (a and c) xor (b and c)
           temp2:= s0 + maj
    Produce the final hash value
      hash: = h0 append h1 append h2 append h3 append h4 append h5 append h6 append h7
End
```

Figure 3.5 SHA-256 hashing Algorithm

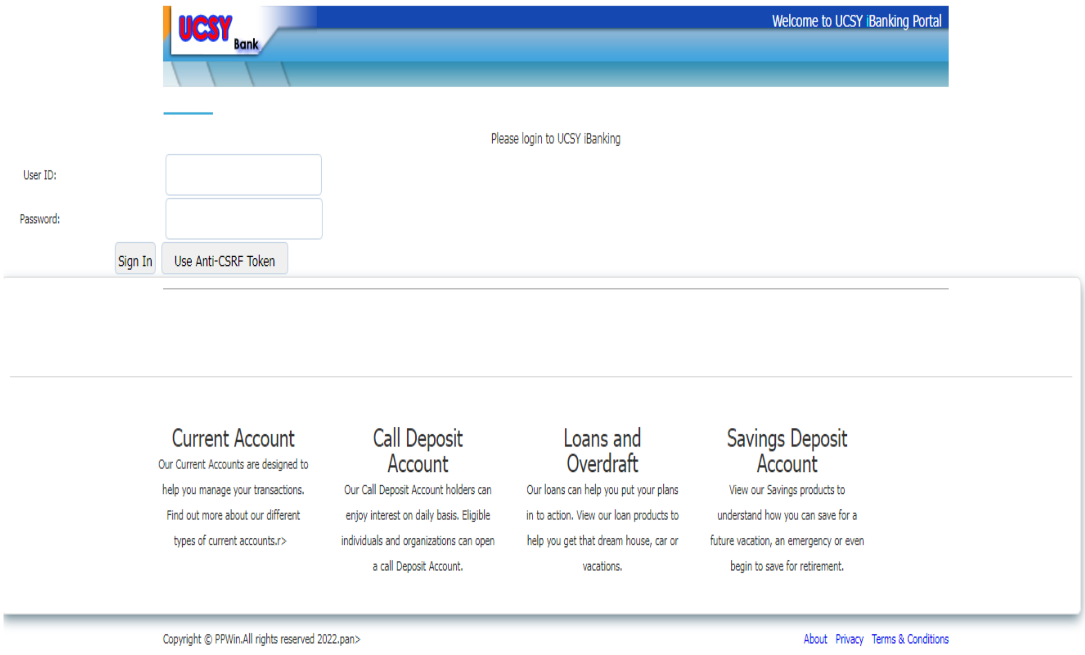
CHAPTER 4
IMPLEMENTATION OF THE PROPOSED SYSTEM

4.1 Experimental Setup

The purpose of this chapter is to present the implementation, design, and performance evaluation of the proposed system. The banking application testing system uses the anti-csrf token to prevent the csrf attack the web application.

4.2 Implementation of the System

When the system starts, the user can see the main form of the system as shown in Figure (4.1). The main form consists of the normal banking form and preventing the attack using anti-csrf token form.



Please login to UCSY iBanking

Current Account
Our Current Accounts are designed to help you manage your transactions. Find out more about our different types of current accounts.r>

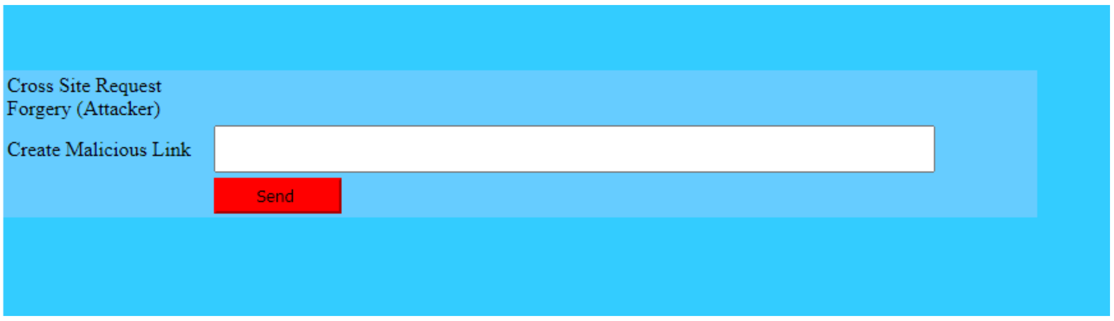
Call Deposit Account
Our Call Deposit Account holders can enjoy interest on daily basis. Eligible individuals and organizations can open a call Deposit Account.

Loans and Overdraft
Our loans can help you put your plans in to action. View our loan products to help you get that dream house, car or vacations.

Savings Deposit Account
View our Savings products to understand how you can save for a future vacation, an emergency or even begin to save for retirement.

Copyright © PPWin.All rights reserved 2022.pan>

Figure 4.1 Main Form of the Proposed System

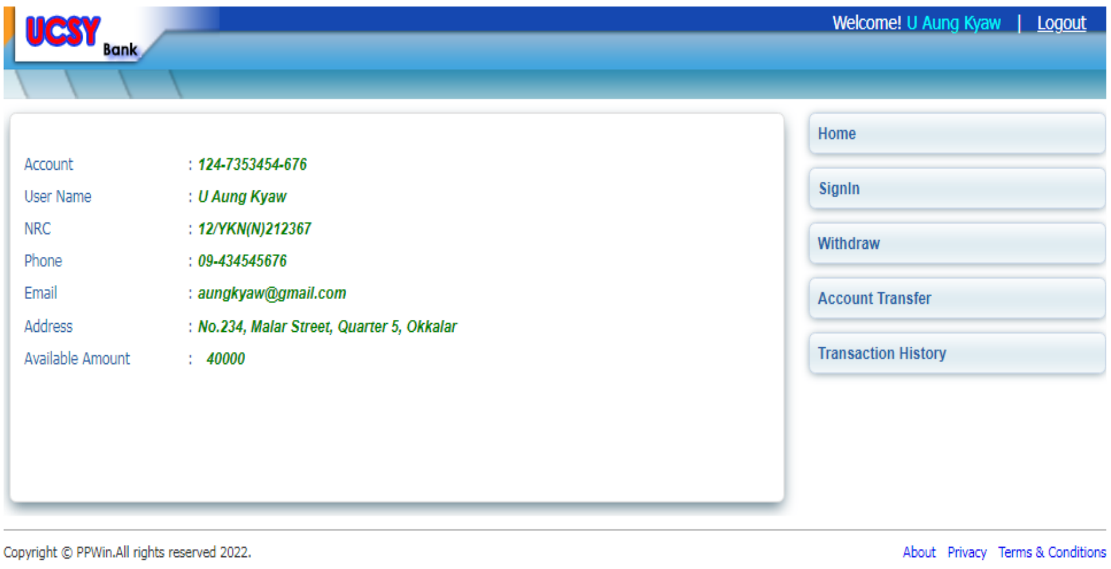

Cross Site Request Forgery (Attacker)

Create Malicious Link	

Figure 4.2 Creating a Malicious Link Form

In Figure (4.2), The attacker creates a malicious link to trick the user. The link sends within a hyperlink. Send button is used to make the process.

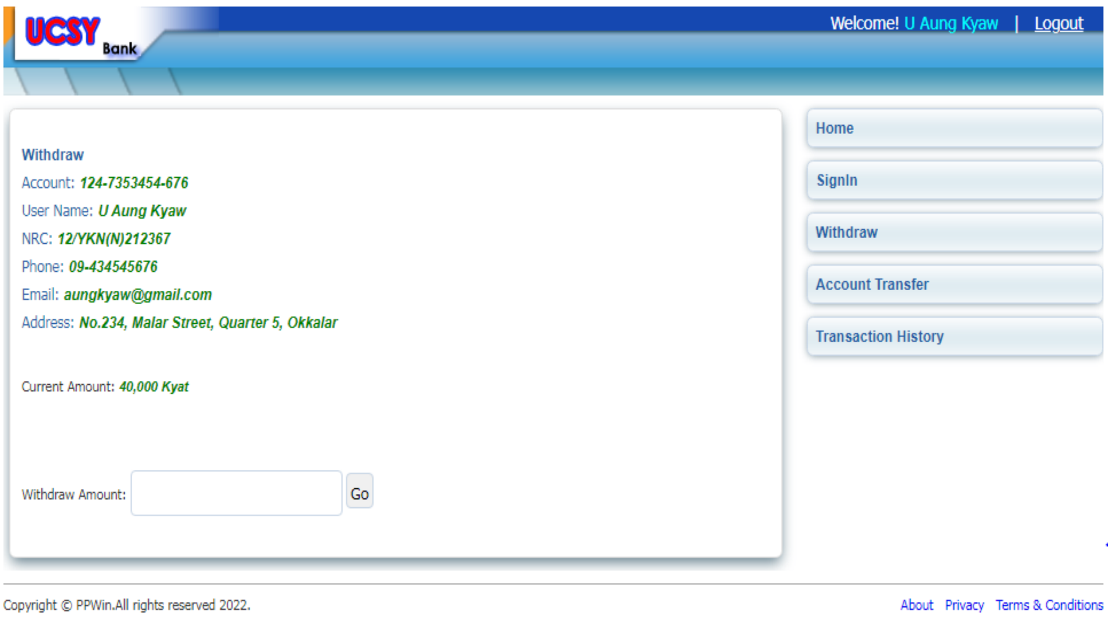
The two options are used to enter the banking website. User reaches home page in Figure (4.3). In the home page, the user can view user account, user name, NRC number, phone number, email, user address and available amount. And then, the user can make withdraw funds and transfer funds to another user. After the user makes the transaction,the results show in the transaction history. And the user can logout from the banking application.



Account	: 124-7353454-676
User Name	: U Aung Kyaw
NRC	: 12/YKN(N)212367
Phone	: 09-434545676
Email	: aungkyaw@gmail.com
Address	: No.234, Malar Street, Quarter 5, Okkalar
Available Amount	: 40000

Copyright © PPWin.All rights reserved 2022.

Figure 4.3 Home Page Form



UCSY Bank

Welcome! U Aung Kyaw | Logout

Withdraw
Account: 124-7353454-676
User Name: U Aung Kyaw
NRC: 12/YKN(N)212367
Phone: 09-434545676
Email: aungkyaw@gmail.com
Address: No.234, Malar Street, Quarter 5, Okkalar

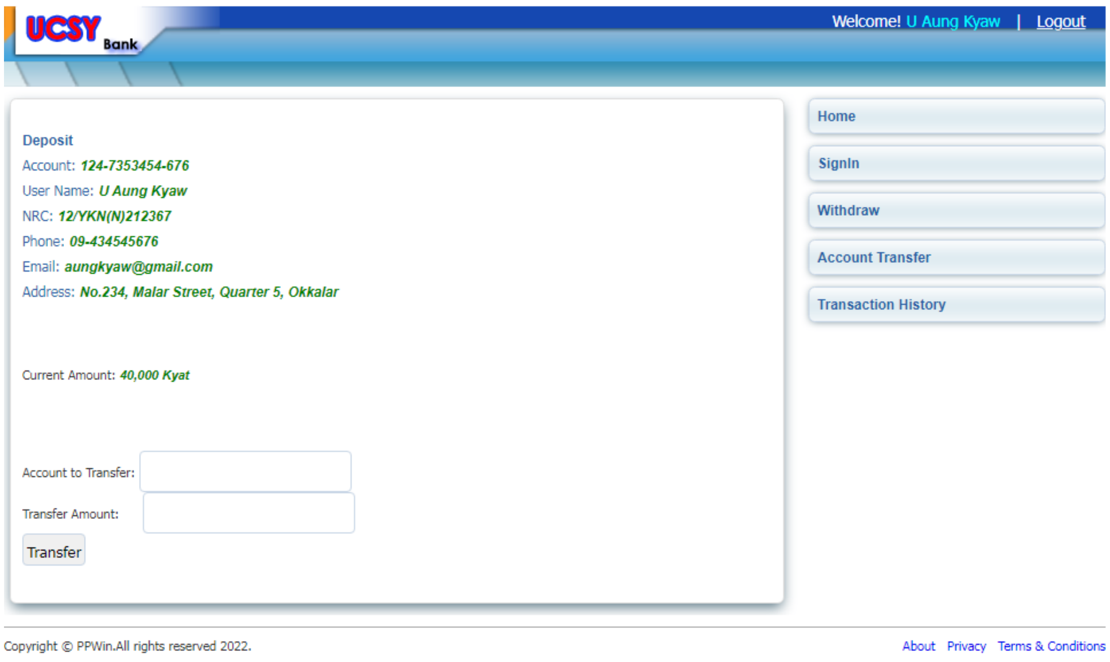
Current Amount: 40,000 Kyat

Withdraw Amount: Go

Copyright © PPWin.All rights reserved 2022.

Figure 4.4 Withdraw Form

In withdraw form, the user can make withdraw process from the account. In the form consists of account number, user name, NRC number, phone number, email, address and the current amount. The user can enter the withdraw amount in the entry box. Shown in Figure (4.4). Go button is used to make the process.



UCSY Bank

Welcome! U Aung Kyaw | Logout

Deposit
Account: 124-7353454-676
User Name: U Aung Kyaw
NRC: 12/YKN(N)212367
Phone: 09-434545676
Email: aungkyaw@gmail.com
Address: No.234, Malar Street, Quarter 5, Okkalar

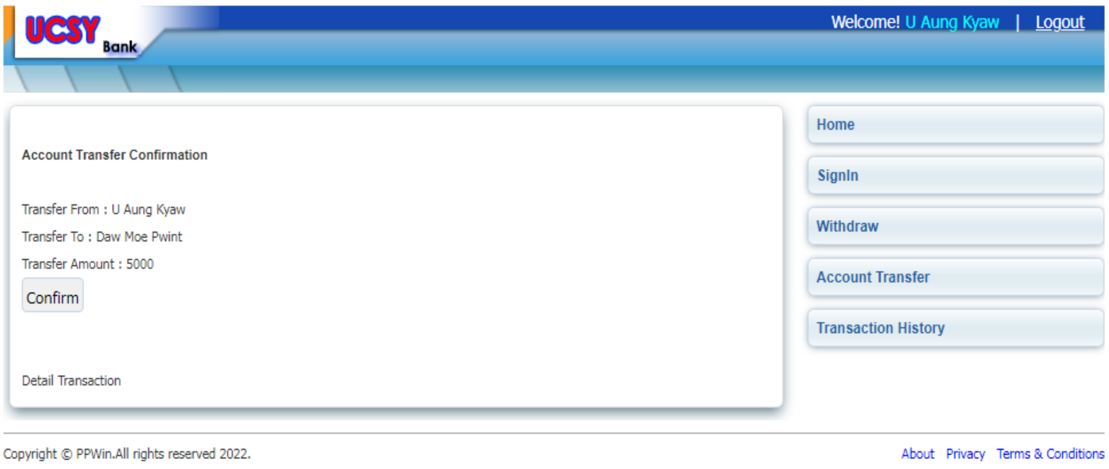
Current Amount: 40,000 Kyat

Account to Transfer:
Transfer Amount:
Transfer

Copyright © PPWin.All rights reserved 2022.

Figure 4.5 Transfer Form

In the transfer form, the user can transfer amount to another account. The user enters the transfer account and the amount in the entry box. The user uses the transfer button to make the process in Figure (4.5). And then, in the confirm page consists of transfer account name and the transfer amount. The confirm button is used to process the transaction.



Account Transfer Confirmation

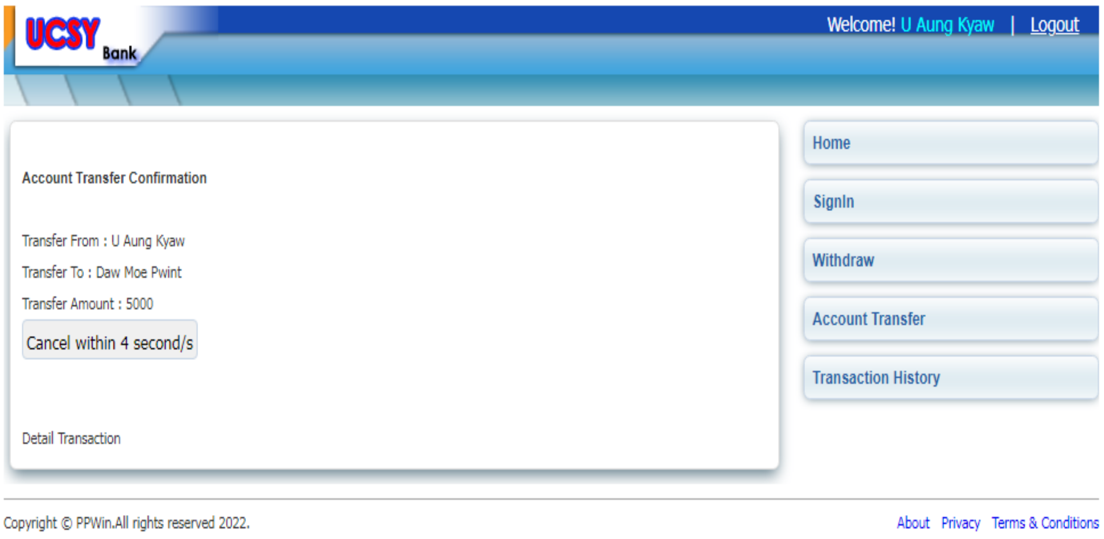
Transfer From : U Aung Kyaw
Transfer To : Daw Moe Pwint
Transfer Amount : 5000

Detail Transaction

Copyright © PPWin.All rights reserved 2022.

Figure 4.6 Account Transfer Confirmation Form

In Figure (4.6), the user uses confirm button to transfer amount. In Figure (4.7), the user can make the cancel transaction in within 5 seconds. The detail transaction shows in Figure (4.8) when the user doesn't make the cancel transaction.



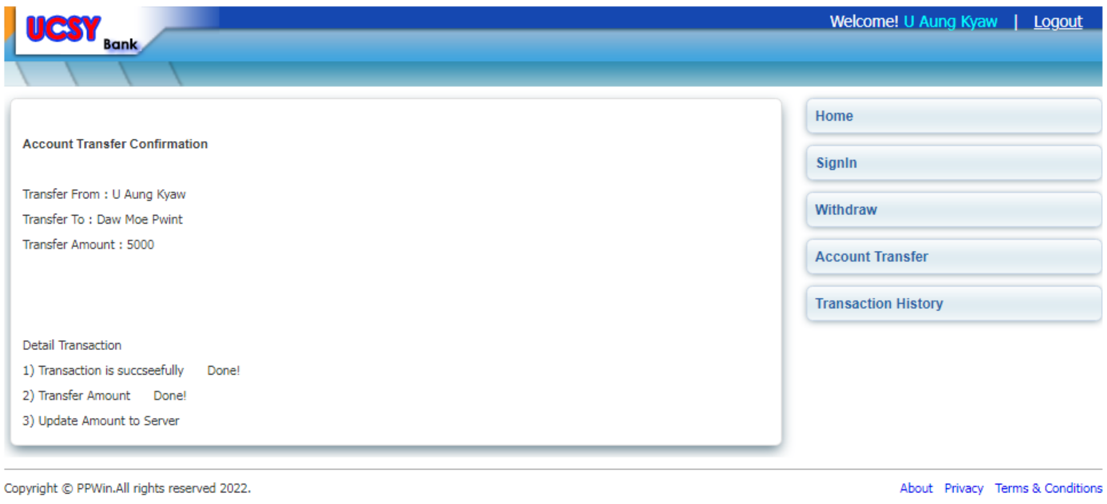
Account Transfer Confirmation

Transfer From : U Aung Kyaw
Transfer To : Daw Moe Pwint
Transfer Amount : 5000

Detail Transaction

Copyright © PPWin.All rights reserved 2022.

Figure 4.7 Cancel Transaction Form



UCSY Bank

Welcome! U Aung Kyaw | Logout

Account Transfer Confirmation

Transfer From : U Aung Kyaw
Transfer To : Daw Moe Pwint
Transfer Amount : 5000

Detail Transaction
1) Transaction is succseefully Done!
2) Transfer Amount Done!
3) Update Amount to Server

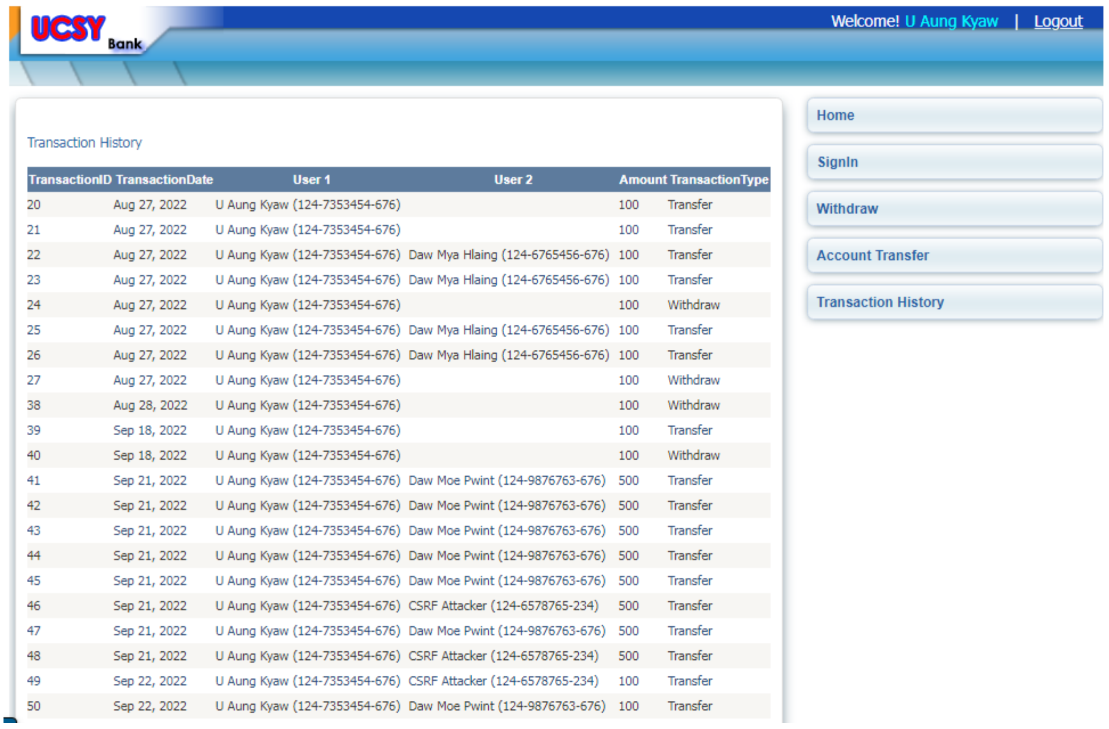
Home
SignIn
Withdraw
Account Transfer
Transaction History

Copyright © PPWin.All rights reserved 2022.

About Privacy Terms & Conditions

Figure 4.8 Detail Transaction Form

In the transaction history form, the user can view the history. The form consists of transaction ID, transaction Date, user1, user2, amount and transaction Type in Figure (4.9).



UCSY Bank

Welcome! U Aung Kyaw | Logout

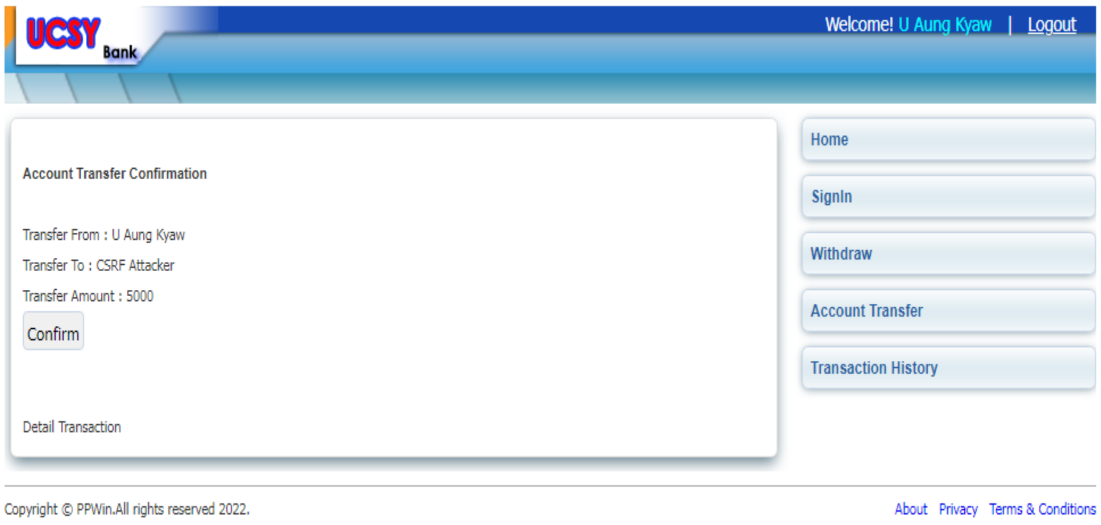
Home
SignIn
Withdraw
Account Transfer
Transaction History

Transaction History

TransactionID	TransactionDate	User 1	User 2	Amount	TransactionType
20	Aug 27, 2022	U Aung Kyaw (124-7353454-676)		100	Transfer
21	Aug 27, 2022	U Aung Kyaw (124-7353454-676)		100	Transfer
22	Aug 27, 2022	U Aung Kyaw (124-7353454-676)	Daw Mya Hlaing (124-6765456-676)	100	Transfer
23	Aug 27, 2022	U Aung Kyaw (124-7353454-676)	Daw Mya Hlaing (124-6765456-676)	100	Transfer
24	Aug 27, 2022	U Aung Kyaw (124-7353454-676)		100	Withdraw
25	Aug 27, 2022	U Aung Kyaw (124-7353454-676)	Daw Mya Hlaing (124-6765456-676)	100	Transfer
26	Aug 27, 2022	U Aung Kyaw (124-7353454-676)	Daw Mya Hlaing (124-6765456-676)	100	Transfer
27	Aug 27, 2022	U Aung Kyaw (124-7353454-676)		100	Withdraw
38	Aug 28, 2022	U Aung Kyaw (124-7353454-676)		100	Withdraw
39	Sep 18, 2022	U Aung Kyaw (124-7353454-676)		100	Transfer
40	Sep 18, 2022	U Aung Kyaw (124-7353454-676)		100	Withdraw
41	Sep 21, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	500	Transfer
42	Sep 21, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	500	Transfer
43	Sep 21, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	500	Transfer
44	Sep 21, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	500	Transfer
45	Sep 21, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	500	Transfer
46	Sep 21, 2022	U Aung Kyaw (124-7353454-676)	CSRF Attacker (124-6578765-234)	500	Transfer
47	Sep 21, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	500	Transfer
48	Sep 21, 2022	U Aung Kyaw (124-7353454-676)	CSRF Attacker (124-6578765-234)	500	Transfer
49	Sep 22, 2022	U Aung Kyaw (124-7353454-676)	CSRF Attacker (124-6578765-234)	100	Transfer
50	Sep 22, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	100	Transfer

Figure 4.9 Transaction History Form

The attacker builds an exploit URL tricking the user into executing the action with social engineering. In Figure (4.10), the attack sends a malicious link and the user clicks the link. The attacker injects to the bank application and changes name and the transfer amount at the same time. The amount reaches to the attacker account.



Account Transfer Confirmation

Transfer From : U Aung Kyaw
Transfer To : CSRF Attacker
Transfer Amount : 5000

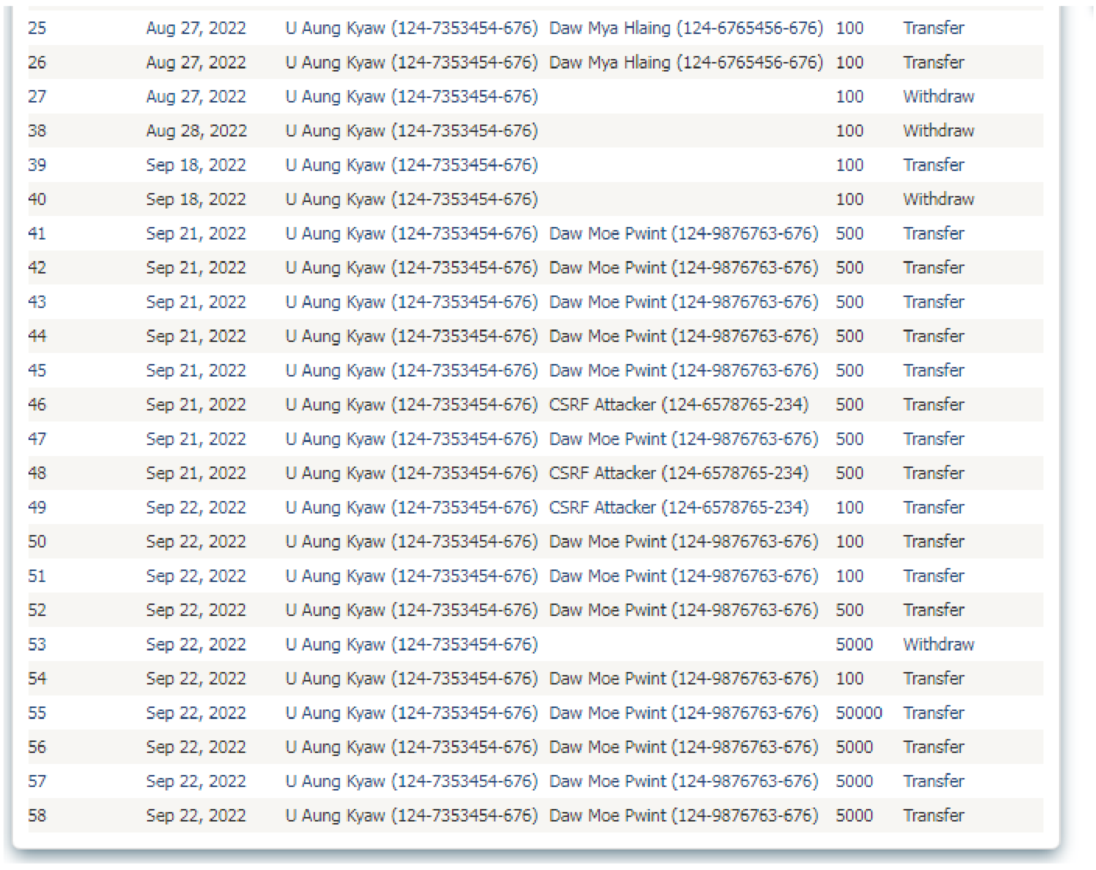
Detail Transaction

Copyright © PPWin.All rights reserved 2022.

Figure 4.10 Attack Form

4.3 Experimental Results

The proposed system prevents the csrf attack using anti-csrf token. Blum Blum Shub is used to generate token and hash function uses HMAC-sha 256. The attack can inject to the bank application when the user enters with the normal bank application. The attacker sends the malicious link to the user and the user clicks the link when logged to bank application. The attacker gets the user authorization and changes name, amount to his account. The user uses anti-csrf token in the application, the attack cannot inject to bank application. The user gets the secure transaction when making the transfer amount.



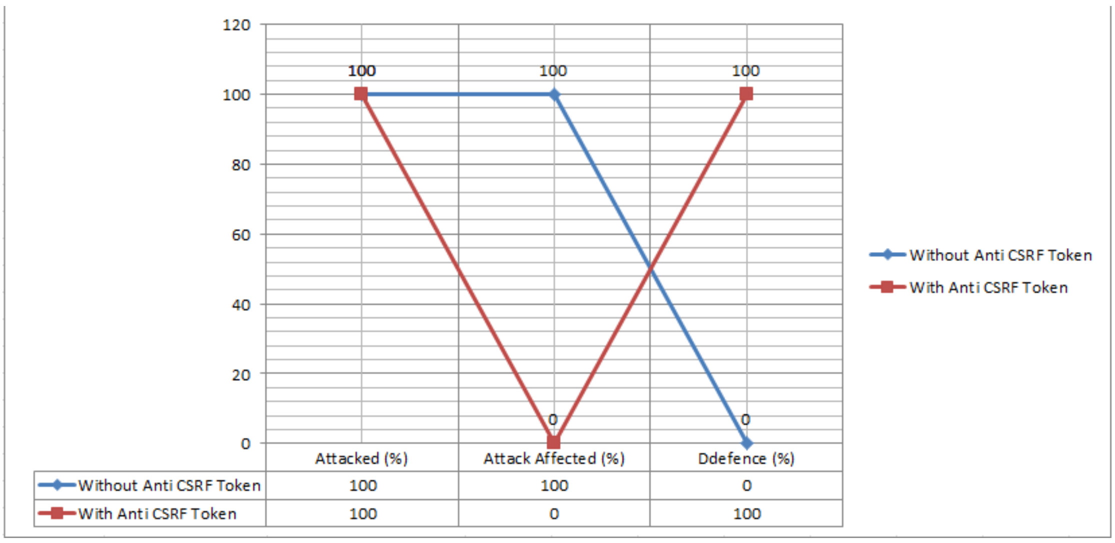
25	Aug 27, 2022	U Aung Kyaw (124-7353454-676)	Daw Mya Hlaing (124-6765456-676)	100	Transfer
26	Aug 27, 2022	U Aung Kyaw (124-7353454-676)	Daw Mya Hlaing (124-6765456-676)	100	Transfer
27	Aug 27, 2022	U Aung Kyaw (124-7353454-676)		100	Withdraw
38	Aug 28, 2022	U Aung Kyaw (124-7353454-676)		100	Withdraw
39	Sep 18, 2022	U Aung Kyaw (124-7353454-676)		100	Transfer
40	Sep 18, 2022	U Aung Kyaw (124-7353454-676)		100	Withdraw
41	Sep 21, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	500	Transfer
42	Sep 21, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	500	Transfer
43	Sep 21, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	500	Transfer
44	Sep 21, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	500	Transfer
45	Sep 21, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	500	Transfer
46	Sep 21, 2022	U Aung Kyaw (124-7353454-676)	CSRF Attacker (124-6578765-234)	500	Transfer
47	Sep 21, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	500	Transfer
48	Sep 21, 2022	U Aung Kyaw (124-7353454-676)	CSRF Attacker (124-6578765-234)	500	Transfer
49	Sep 22, 2022	U Aung Kyaw (124-7353454-676)	CSRF Attacker (124-6578765-234)	100	Transfer
50	Sep 22, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	100	Transfer
51	Sep 22, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	100	Transfer
52	Sep 22, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	500	Transfer
53	Sep 22, 2022	U Aung Kyaw (124-7353454-676)		5000	Withdraw
54	Sep 22, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	100	Transfer
55	Sep 22, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	50000	Transfer
56	Sep 22, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	5000	Transfer
57	Sep 22, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	5000	Transfer
58	Sep 22, 2022	U Aung Kyaw (124-7353454-676)	Daw Moe Pwint (124-9876763-676)	5000	Transfer

Figure 4.11 Non-secure Transaction and Secure Transaction Form

The proposed system was evaluated in terms of in percentage. In detection attack, without anti-csrf token in 100 times, the attack affected 100 % and defense in 0%. In detection attack, with anti-csrf token in 100 times, the attack affected 0% and defense in 100%. In the system, the detection of the attack is showed with the percentage in table (4.1).

Table (4.1) Detection of the Attack

Detection	Attacked (%)	Attack Affected (%)	Defense (%)
Without Anti CSRF Token	100	100	0
With Anti CSRF Token	100	0	100



	Attacked (%)	Attack Affected (%)	Ddefence (%)
Without Anti CSRF Token	100	100	0
With Anti CSRF Token	100	0	100

Figure 4.12 Attack Detection Rate with Graph

In figure (4.12), shows attack detection rate with graph of the proposed system. in the system shows attack affected rate and defense rate with percentage.

CHAPTER 5
CONCLUSION

In this thesis, the prevention of the cross-site request forgery (CSRF) attack is used the anti-csrf token. This thesis presents the proposed algorithm, Blum Blum Shub Algorithm to generate anti-csrf token and HMAC sha-256 uses to generate hash and to transfer the token within the hidden field. The experimental results are produced the secure transaction. The system is implemented using C#.Net programming language on the web platform. In this chapter, the summary of the main conclusion and advantages, limitations, and further extensions are suggested.

The proposed system uses URLs as input for banking web application. The attack sends a malicious link to the target web application. The system prevents the attack vulnerability using the token checks at the server side. This system is to show the secure transaction, the attack transaction, record transaction history and prevents the csrf attack using the anti-csrf token.

The experimental results of the proposed algorithm produce percentage result with the graph. The attack detects in 100 times, the user uses two types of web application, sign in without token and sign in without token application. The attack detects in 100 times, the user uses sign in with token application that the attack affected in 0 % and defense in 100%. The attack detects in 100 times, the user uses sign in without token application that the attack affected in 100 % and defense in 0%.

In conclusion, the proposed Blum Blum Shub algorithm is a secure, randomness, unique. And it helps web developers to secure their web applications from being attacked.

5.1 Limitation and Further Extension

In this study, the proposed system does not consider other web application vulnerabilities such as buffer overflow, XSS, command injection, and so on. The future work will be dedicated to prevent all vulnerabilities in websites and to get secure transaction when the transferring funds in web application.

AUTHOR'S PUBLICATION

[1] Phyu Phyu Win, Yi Mon Thet, University of Computer Studies, Yangon, Myanmar, *"Prevention of Cross-Site Request Forgery Using Anti-CSRF Token"*, to be published in the Proceedings Journal Organizing Committee PSC 2022, Yangon, Myanmar, 2022.

REFERENCES

[1] A.Barth, C.Jackson, and J.C.Mitchell. “Robust defenses for cross site request forgery”. In Proc. ACM Conference on Computer and Communications Security (CCS), Oct, 2008.

[2] Anti-csrf token, https://blog.insiderattack.net/anti-csrf-tokens-to-prevent-cross-site-request-forgery-csrf.

[3] Banking System prevent from the Attack, https://blog.nettitude.com/how-can-banks-protect-themselves-from-cyber-attacks.

[4] Emil Semastin, Sami Azam,Preventive Measures for Cross Site Request Forgery Attacks on Web-based Applications, College of Engineering, IT and Environment, Charles Darwin University, Australia, 2018.

[5] HMAC, hash-based message authentication code, https://en.wikipedia.org/wiki/HMAC

[6] Lenore Blum, Manuel Blum and Michael Shub, Blum Blum Shub Algorithm, https://en.wikipedia.org/wiki/Blum_Blum_Shub, 1986.

[7] Nenad Jovanovic, Engin Kirda, and Christopher Kruegel. “Preventing cross site request forgery attacks”.In IEEE International Conference on Security and Privacy in Communication Networks (SecureComm), 2006.

[8] OWASP. Top ten most critical web applications securityvulnerabilities.https://www.owasp.org/index.php/Top_10_2013Top_10.Forgeries.www.securityfocus.com/archive/1/19S90,2001.

[9] OWASP. (2017), CSRF prevention cheat sheet. https://www.owasp.org/index.php/CrossSite_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet.

[10] Ozgur Yildirim, Daniel Gerep, SHA-256 Algorithm, 2020. https://blog.boot.dev/cryptography/how-sha-2-works-step-by-step-sha-256/

[11] Sooel Son, "Prevent Cross site Request Forgery Attack PCRF", Global science journals (GSJ) userweb.cs.utexas.edu/ Samuel/PCRF/Final_PCRF_paper.pdf.

[12] Sentamilselvan. K, Lakshmana Pandian. S ": Preventive Measures", Assistant Professor Kongu Engineering College Perundurai, Tamilnadu, 2014.

[13] W. Zeller and E. W. Felten, "Cross-Site Request Forgeries: Exploitation and Prevention," Technical Report, Princeton University, 2008.