ENHANCED WEB SECURITY APPLICATION FOR ONLINE FINANCIAL TRANSACTIONS

BY

Olufemi Oriola Sodiq ANIFOWOSE

DEPARTMENT OF MATHEMATICS,
FACULTY OF SCIENCE,
AHMADU BELLO UNIVERSITY, ZARIA, NIGERIA

AUGUST, 2016.

TITLE PAGE

ENHANCED WEB SECURITY APPLICATION FOR ONLINE FINANCIAL TRANSACTIONS

BY

Olufemi Oriola Sodiq ANIFOWOSE

B.Sc Computer Science (Al-Hikmah University), 2010

M.Sc/SCI/11264/2011-12

A DISSERTATION SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES,

AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF A MASTER OF SCIENCE DEGREE IN COMPUTER SCIENCE

DEPARTMENT OF MATHEMATICS,

FACULTY OF SCIENCE

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA

AUGUST, 2016.

DECLARATION

I declare that the work in this dissertation entitled "Enhanced Web Security Application for Online Financial Transactions" has been performed by me in the Department of Mathematics under the supervision of Dr. S. E. Abdullahi and Professor S. B. Junaidu.

The information derived from the literature has been duly acknowledged in the text and list of references provided. No part of this dissertation report was previously presented for another degree or diploma at any university.

Olufemi Oriola Sodiq ANIFOWOSE		
Name of student	Signature	Date

CERTIFICATION

This dissertation entitled "ENHANCED WEB SECURITY APPLICATION FOR ONLINE FINANCIAL TRANSACTIONS" by Olufemi Oriola Sodiq ANIFOWOSE meets the regulation governing the award of the degree of Master of Science in Computer Science of Ahmadu Bello University, Zaria and is approved for its contribution to knowledge and literary presentation.

(Signature) _____

_____ Date _____

Chairman, Supervisory Committee

(Signature) _____

_____ Date _____

Member, Supervisory Committee

(Signature) _____

_____ Date _____

Member, Supervisory Committee

(Signature) _____

_____ Date _____

Head of Department

(Signature) _____

_____ Date _____

Dean, School of Postgraduate Studies

DEDICATION

ALHAMDULILLAHI ROBBIL ‘AL-AMEEN, I dedicate this research work to my parents Engr. M. A. Anifowose and Mrs F. A. Anifowose for their unending love, support, patience and prayers that has kept me going through out this research. God bless you all.

ACKNOWLEDGEMENT

This research work has been full of ups and downs and it would be an injustice on my part not to give honour to whom honour is due.

First of all, my deepest gratitude goes to GOD ALMIGHTY, Most Gracious, Most Merciful who has kept me alive and in good health from day one of this research work till the end. All I can say is…ALHAMDULLILAH.

I would like to also express my sincere gratitude to my supervisors, Dr. S. E. Abdullahi and Professor S. B. Junaidu for the continuous support of my MSc study and related research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this dissertation. I could not have imagined having better supervisors and mentors for my work. Besides my supervisors, my gratitude also goes to other lecturers of the department like Professor O. S. Adewale and Dr. M. B. Hammawa that taught me during my course work period. I am immensely grateful to you all for the knowledge being passed.

I would also like to appreciate my guardians in Zaria Alhaji Bayo Lawal and his wife for taking good care of me in the early days of my educational sojourn. May almighty Allah reward you bountifully. To my family in kaduna, Mr and Mrs Sebiotimo and children, I thank you so much for your love and support, to Alhaja Idiat Folorunsho, thank you so much for your love, support and financial assistance, I'm very grateful and to my wonderful cousins Aisha, Kafayat, Bashir(Fresh Prince of Hajji Guy), Jumoke Adebayo, Aunty Tinu and Moruf, Aunty Kunmi and her family, you are all appreciated.

I thank my fellow MSc classmates like Ishaya, Najim, Yomi, Umar, Vivian, Rosemary, Alhamdu, Solomon, Madam chioma, Madam Sikirat , Hajara, Aisha Shehu, Aisha Amoka, Moruf(el-classico), Uche, Saheed, Mr Pam and all other members I could not mention for deeming it fit to make me their class rep. I tell you, it wasn't easy but it was fun all the way. I will always reminisce on the endless tutorials we had, for the sleepless nights we had working together before deadlines, and for all the fun we have had in the past years.

My sincere thanks specially goes to Mr Laide Oyelade, a dear friend and my course mate as well who has been my greatest assistance on this dissertation work. Without you, I would not have been able to go through this work. Thank you so much, only God can reward you and your efforts. I also want to thank my very good friend and course mate Femi Ooju a.k.a Elder Google for your insightful comments, support and encouragement towards this work, God bless you and your family. To my colleagues ThankGod, Shuaib, Aminu Onimisi, Aminu Umar and Ibrahim Enesi, I thank you for your valuable input towards this work. All I can say is Gracias!!!

I would also like to appreciate my other MSc colleagues and friends Olamide(Ollywise), KB, Aminu, Elisha(Malabo), Tunde Aina, Tajudeen, Abayomi, Kenny, Johnson, Slim YKD, Ehis and body dynamics members for their supports and prayers. I thank God for meeting respected people like you throughout this educational sojourn.

My sincere appreciation goes to my favourite cousin, my brother and my pal, Yinka Anifowose for your constant support and prayers. You show that blood is truly thicker than water and never forget our watch word, we will make it! Special thanks also to my wonderful cousin as well, Hakeem Badmus (Haykay) for your support during hard times. You are well appreciated and not also forgetting my good brother as well Mr. Tayo Odutayo and family for the love and care, I do appreciate.

Last but not the least, I would like to thank my family especially to my parents for supporting me financially and spiritually throughout writing this dissertation and my life in general.

ABSTRACT

Online users now make use of internet banking as a major platform of making payments of products online. Cybercriminals are using newer and more advanced methods to target online users. One of the fastest growing threats and attacks in the world today is Man-in-the-Browser (MITB) attacks. As the advance in technology continues to influence the way society makes payment for goods and services, then more advanced security approach is required for transaction authentication on the internet. This dissertation provides a more secure authentication for online transaction using an enhanced security approach that uses an **Anti-form grabbing technique** to encode user inputs to random characters, **JSON Web Token (JWT)** to provide and secure safe passage of information between two parties, a **One Time Password (OTP) token** for authentication and the **use of Email** as another verification channel from the server to combat MitB attacks.

TABLE OF CONTENTS

TITLE PAGE ii

DECLARATION iii

CERTIFICATION iv

DEDICATION v

ACKNOWLEDGEMENT vi

ABSTRACT ix

TABLE OF CONTENTS x

LIST OF FIGURES xiv

LIST OF TABLES xv

LIST OF ABBREVIATIONS xvi

CHAPTER ONE 1

INTRODUCTION 1

1.1 Background of the Study 1

1.2 Research Motivation 2

1.3 Research Aim and Objectives 3

1.4 Research Methodology 4

1.5 Contribution to Knowledge 4

1.6 Organization of the Dissertation 4

CHAPTER TWO 5

LITERATURE REVIEW 5

2.1 Introduction 5

2.2 History of the Web 5

2.2.1 Online banking 5

2.2.2 History of Online Banking 6

2.2.3 Security 8

2.3 Online Attacks 10

2.3.1 MAN-IN THE-BROWSER ATTACK 11

2.3.2 Other Threats 12

2.4 Security Features 14

2.4.1 SALT 14

2.4.2 HASHING 15

2.4.3 SESSIONS 15

2.4.4 DYNAMIC JAVASCRIPT 16

2.5 Technologies Used 17

2.5.1 Server Side Scripting Language 17

2.5.2 Client Side Scripting Language 17

2.5.3 HTML 18

2.5.4 MySQL 18

2.6 Literature Review 19

CHAPTER THREE 22

MATERIALS AND METHODS 22

3.1 Introduction 22

3.2 The Proposed Enhanced Security System 22

3.2.1 Functionalities of the System 23

3.2.2 System Architecture 24

3.2.3 System Flow Chart 26

3.3 The Security Model Mitigating MITB 28

3.3.1 The Anti-Form Grabbing Technique 28

3.3.2 Token Generation 31

3.3.3 JSON Web Token (JWT) 32

3.3.4 Email Verification Service 32

3.4 Theoretical Evaluation of the Security Model 33

CHAPTER FOUR 38

IMPLEMENTATION AND DISCUSSION 38

4.1 Introduction 38

4.2 Code Implementation 38

4.2.1 Coding the proposed algorithm 38

4.2.3 Email authentication handler 42

4.3 Discussion of Results 42

4.4 Model Comparison Analysis 46

CHAPTER FIVE 50

SUMMARY, CONCLUSION AND RECOMMENDATION 50

5.1 Summary 50

5.2 Conclusion 51

5.3 Recommendation 51

REFERENCES 52

LIST OF FIGURES

Figure 2. 1: Example of a transaction altered during man in the browser attack. 12

Figure 2. 2: Visualisation of a man in the middle attack. 13

Figure 2. 3: Illustration of a server with a separate session for each client where the data is stored. 16

Figure 3. 1: The System Architecture……...24

Figure 3.2: Flow Chart……..26

Figure 4. 1: Pseudo Alphabet System creation……..…39

Figure 4. 2: Token Generation Algorithm 39

Figure 4. 3: Mail Verification Sender 40

Figure 4. 4: Encryption of Transaction Details via JWT 41

Figure 4. 5: The Login Form.....42

Figure 4. 6: Entry Form 43

Figure 4. 7: Entry Form without Anti-form grabbing.....43

Figure 4. 8: Token Verification 44

Figure 4. 9: Entry Form with Anti-form Grabbing 44

Figure 4. 10: Account Verification 45

Figure 4. 11: Security Analysis Chart.....48

LIST OF TABLES

Table 4. 1: Model Comparison Analysis 46

LIST OF ABBREVIATIONS

CERN: European Organization for Nuclear Research.

CSS: Cascading Style Sheet

HTML: Hypertext Markup Language.

JWT: JSON Web Token.

MITB: Man in the browser.

MITM: Man in the middle.

OTP: One Time Password.

PHP: Hypertext Preprocessor.

RDBMS: Relational Database Management System.

SSL: Secured Socket Layer.

TAN: Transaction Authentication Number.

TSL: Transport Layer Security.

XML: Extensible Markup Language.

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

When using services in a web environment, security is of great importance especially for both the user and the provider. The information in use must be handled in a way that does not compromise its security. Passwords are only secured as long as the user keeps them secret. Not everyone is aware of the risk that comes with compromised passwords and other security leaks (Nilsson, 2012).

Lately, client side attacks on online banking and electronic commerce are on the rise due to inadequate security awareness amongst end users. As a result, end user would not be aware if there is vulnerability on their machine or platform that might lead to client side attack such as man-in-the-browser (MitB) attacks. For instance, man-in-the-middle (MitM) attack techniques which are mainly targeting the information flow between a client and a server have now evolved to become man-in-the-browser (MitB) attack. MitM attack occurs when someone manages to eavesdrop on web traffic by fooling the other connections (Web Server and Client Server) to connect to the attacker instead of connecting to each other. One of the common ways to counter these attacks is to use secure channel like SSL(Secured Socket Layer) when sensitive data is transmitted between the client and the server. MitB attack is designed to infiltrate the client software such as the internet browser and manipulate or steal any sensitive information. It takes place on the client side of the connection. The ability of these trojans to perform Man-in-the-Middle-Attacks/ Man-in-the-Browser-Attacks on valid transactions is most worrying since they silently change the information from the client such as the user's bank details or sensitive information to the attacker's account details(Fazli *et al*., 2012).

The password remains the most popular authentication mechanism in use today. In order to complete any web-based transaction exchange, the online user will be required to enter his/her password into an online system.

As technological advances continue to influence the way society makes payment for goods and services, the requirement for more advanced security approaches for transaction verification in the online environment increases.

In order to mitigate these security issues, this proposed dissertation proffers a solution to the problem by integrating different authentications and methods to provide an improved and secure online transaction between the client and the server. The thesis introduces an anti-form grabbing technique which disallows the attacker from "grabbing" sensitive information and modifying it when they are being sent to the server by the client and also protects the web contents through JSON Web Token (JWT) which is a safe means of transferring information between two parties. The system also minimizes the risk of man-in-the-browser (MitB) by using One Time Password (OTP), a password that is valid for only one login session or transaction within a limited time along with the use of Email as a different verification channel.

1.2 Research Motivation

Cyber criminals are using newer and more advanced methods to target online users and one of the fastest growing threats in the world today is man-in-the-browser (MitB) Trojan attacks (RSA, 2011). What makes MitB attacks difficult to detect from the client side is that any activity performed seems as though it is originating from the legitimate user's web browser and with this, it silently changes the information of the user's account details to the attacker's account details which is most worrying.

The losses attributed to financial fraud are alarming. The financial services industry has become a primary target of cyber-attacks on a global scale and, in 2009 alone, suffered losses totalling $54 billion - an increase from $48 billion in 2008 (SafeNet, 2010).

In 2010, there has been an exponential increase in the number of these attacks against financial institutions including the European consumer banking and U.S. corporate banking markets (RSA, 2011).

The hackers target the most sensitive information such as the account number and the amount and alter it for their own benefit. One must be able to trust the data that is transmitted to the bank server which is why an enhanced web security application will be developed to tackle the online security threat.

1.3 Research Aim and Objectives

The aim of this dissertation is to develop mechanisms for preventing Man-in-the-Browser (MitB) attacks on online financial transaction. The research objectives of this proposed dissertation are to:

a) Develop anti-form grabbing technique to encode the user inputs as they are being entered.

b) Implement an authentication mechanism using One Time Password (OTP).

c) Develop a medium that make use of Email from the server for identity verification.

1.4 Research Methodology

The following are methods that were adopted for this research:

a) Develop the anti-form grabbing algorithm to encode user inputs.

b) Develop the OTP algorithm to authenticate the user.

c) Develop a medium that make use of Email from the server for identity verification.

d) Design the proposed system architecture to mitigate MitB attack.

e) Implement the proposed system.

f) Assess performance of the proposed system.

1.5 Contribution to Knowledge

The Enhanced Web Security Application was developed to tackle MitB attacks and in doing that, the following contributions were made to this dissertation:

a) The Anti-form grabbing algorithm was developed to tackle form grabbing which is a technique of MitB attack.

b) The web contents were encrypted with JWT to protect the information exchange between two parties.

c) The use of Email for verification channel.

1.6 Organization of the Dissertation

The organization of the rest of the dissertation with a brief outline of the chapters is as follows. In chapter 2, history of online banking will be discussed and also related works on MitB will be carried out. In chapter 3, the proposed design of the enhanced security application will be discussed, especially the security components; Anti-form grabbing, JWT, OTP and the use of Email which makes up the system architecture. Chapter 4 will involve the implementation of the design proposed in chapter 3. Chapter 5 will summarize the dissertation and outlining of the future work.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter presents an overview of internet banking and the threats associated with it. The encryption being employed in this work and the technologies used will be discussed with some of the related works done in the past.

2.2 History of the Web

The World Wide Web was officially introduced to the world on August 6, 1991 by Sir Tim Berners-Lee. In the late 1980's, a CERN (European Organization for Nuclear Research) scientist named Tim Berners-Lee came up with the idea of **hypertext**, information that was "linked" to another set of information. His idea was for the researchers at CERN to be able to communicate more easily via a single informational network, instead of many smaller networks that were not linked with one another in any sort of universal way. This hypertext technology included hyperlinks, which enabled users to peruse information from any linked network merely by clicking on a link. Although invented many years earlier, Mr Berners-Lee's invention married hypertext with the Internet and also made available all of the files necessary for people to replicate his invention. (Boswell, 2014)

2.2.1 Online banking

Online banking is a system allowing individuals to perform banking activities at home, via the Internet. Some online banks are traditional banks which also offer online banking, while others are online only and have no physical presence. Online banking through traditional banks enable customers to perform all routine transactions, such as account transfers, balance inquiries, bill payments, and stop-payment requests, and some even offer online loan and

credit card applications. Account information can be accessed anytime, day or night, and can be done from anywhere. A few online banks update information in real-time, while others do it daily. Once information has been entered, it doesn't need to be re-entered for similar subsequent checks, and future payments can be scheduled to occur automatically. Many banks allow for file transfer between their program and popular accounting software packages, to simplify record keeping. Despite the advantages, there are a few drawbacks. It does take some time to set up and get used to an online account. Also, some banks only offer online banking in a limited area. Online-only banks have a few additional drawbacks: an account holder has to mail in deposits (other than direct deposits), and some services that traditional banks offer are difficult or impossible for online-only banks to offer, such as traveller's cheques and cashier's cheques (Batchelor, 2014).

Online banking is also known as "Internet banking" or "Web banking." A good online bank will offer customers just about every service traditionally available through a local branch, including accepting deposits (which is done online or through the mail), paying interest on savings and providing an online bill payment system (Nilsson, 2012).

2.2.2 History of Online Banking

The concept of Internet banking has been simultaneously evolving with the development of the World Wide Web. Programmers working on banking data bases came up with ideas for online banking transactions, sometime during the 1980s. The creative process of development of these services was probably sparked off after many companies started the concept of online shopping. The online shopping promoted the use of credit cards through Internet. Many banking organizations had already started creating data ware housing facilities to ease their working staffs. The developments of these databases were widely used during the development of ATM's(Scholasticus, 2009).

Sometime in 1980s, banking and finance organizations in Europe and United States started suggestive researches and programming experiments on the concept of 'home banking'. Initially in the 80's when computers and Internet were not so well-developed, 'home banking' basically made use of fax machines and telephones to facilitate their customers. The widespread of Internet and programming facilities created further opportunities for development of home banking(Scholasticus, 2009).

In 1983, the Nottingham Building Society, commonly abbreviated and referred to as the NBS, launched the first Internet banking service in United Kingdom. This service formed the basis for most of the Internet banking facilities that followed. This facility was not very well-developed and restricted the number of transactions and functions that account holders could execute. The facility introduced by Nottingham Building Society is said to have been derived from a system known as Prestel, which is deployed by the postal service department of United Kingdom(Scholasticus, 2009).

The first online banking service in United States was introduced, in October 1994. The service was developed by Stanford Federal Credit Union, which is a financial institution. The online banking services are becoming more and more prevalent due to the well-developed systems. Though there are pros and cons of electronic cash, it has become a revolution that is enhancing the banking sector (Scholasticus, 2009).

While financial institutions took steps to implement e-banking services in the mid-1990s, many consumers were hesitant to conduct monetary transactions over the web. It took widespread adoption of electronic commerce, based on trailblazing companies such as America Online, Amazon.com and eBay, to make the idea of paying for items online widespread. By 2000, 80 percent of U.S. banks offered e-banking and customer use grew slowly. At Bank of America, for example, it took 10 years to acquire 2 million e-banking customers. However, a significant cultural change took place after the Y2K scare ended. In 2001, Bank of America became the first bank to top 3 million online banking customers,

more than 20 percent of its customer base. In comparison, larger national institutions, such as Citigroup claimed 2.2 million online relationships globally, while J.P. Morgan Chase estimated it had more than 750,000 online banking customers. Wells Fargo had 2.5 million online banking customers, including small businesses. Online customers proved more loyal and profitable than regular customers. In October 2001, Bank of America customers executed a record 3.1 million electronic bill payments, totalling more than $1 billion. In 2009, a report by Gartner Group estimated that 47 percent of U.S. adults and 30 percent in the United Kingdom bank online. Meanwhile, in the UK e-banking grew its reach from 63% to 70% of Internet users between 2011 and 2012 (Batchelor, 2014).

Today, many banks are internet only banks. Unlike their predecessors, these internet only banks do not maintain brick and mortar bank branches. Instead, they typically differentiate themselves by offering better interest rates and online banking features (Jjchai, 2010).

2.2.3 Security

While the internet offers enormous advantages and opportunities, it also presents security risks. With this in mind, banks take extensive step to protect the information transmitted and processed when banking online. This includes, for example, ensuring that confidential data sent over the internet cannot be accessed or modified by unauthorised parties.

But the banks normally have no influence over the systems used by their customers. The choice is entirely up to them. The systems used by online banking customers are therefore exposed to risks beyond the banks' control. For this reason, the bank cannot assume liability for them (Association of German Banks, 2007).

As a result of these security issues, layered security is provided as an approach that combines risk-based transaction monitoring, Trojan detection, shutdown, and intelligence services, and out-of-band capabilities provides a solid defence to mitigate the threat of man-in-the-browser

attacks. The following RSA solutions help financial institutions address the challenge posed by man-in-the-browser attacks (RSA Lab, 2011):

a) **Transaction Monitoring**: RSA Transaction Monitoring looks at activities conducted post-login to detect unusual behaviour that may indicate a fraud attempt or Trojan activity. It works with any existing strong authentication solution and can be deployed so that it is completely invisible to the end user. Additionally, RSA Transaction Monitoring offers advanced features to identify Trojan behaviour such as the ability to detect manual session hijacking, mule accounts and HTML injection.

b) **Trojan Detection, Shutdown and Intelligence**: The RSA FraudAction™ Anti-Trojan Service works to mitigate the impact of Trojans through the identification and shutdown of known infection points and blocking the resources that Trojans use to communicate (i.e., drop servers, Command & Control servers). In addition, the service attempts to extract stolen credentials and information on mule accounts that are set up to receive fraudulent money transfers (RSA Lab, 2011).

c) **RSA eFraudNetwork**: RSA Adaptive Authentication and RSA Transaction Monitoring leverage information on fraud patterns that is contained within the RSA eFraudNetwork™ data repository. The eFraudNetwork receives feeds of fraud data supplied by a vast network of customers, end users, ISPs, and other third parties. Frequent contributions on cybercrime intelligence are also provided by analysts at RSA's Anti-Fraud Command Center on a regular basis (RSA Lab, 2011).

d) **Out-of-band Authentication:** RSA offers out-of-band phone authentication that allow users to enter a one-time password into the keypad on their telephone. Out-of-band authentication offers a powerful defence against man-in-the-browser attacks because it separates the authentication process from the Web channel making it more difficult to compromise (RSA Lab, 2011).

2.3 Online Attacks

Online attacks also known as web attacks are nowadays one of the major threats on the Internet, and several studies have analysed them, providing details on how they are performed and how they spread (Canali and Balzarotti, 2012). Web attacks are one of the most important sources of loss of financial and intellectual property. In the last years, such attacks have been evolving in number and sophistication, targeting governments and high profile companies, stealing valuable personal user information and causing financial losses of millions of euros. Moreover, the number of people browsing the web through computers, tablets and smartphones is constantly increasing, making web-related attacks a very appealing target for criminals (Canali and Balzarotti, 2012).

Most of the attacks on online banking used today are based on deceiving the user in order to steal login data and valid TANs. Two well-known examples for those attacks are phishing and pharming. Cross-site scripting and keylogger/Trojan horses can also be used to steal login information. A method to attack signature-based online banking methods is to manipulate the used software in a way so that correct transactions are shown on the screen and faked transactions are signed in the background.

A Federal Deposit Insurance Corporation (FDIC) Technology Incident Report, compiled from suspicious activity reports banks file quarterly, lists 536 cases of computer intrusion, with an average loss per incident of $30,000. That adds up to a nearly $16-million loss in the second quarter of 2007. Computer intrusions increased by 150 percent between the first and second quarters of 2007. In 80 percent of the cases, the source of the intrusion is unknown but it occurred during online banking, the report states (SOLIDPASS SECURITY REBORN, 2014).

Nowadays, the most recent kind of attack is the Man-in the Browser attack, where the Trojan horse manipulates the client’s account and amount details to its own destination account detail and also the amount.

2.3.1 MAN-IN THE-BROWSER ATTACK

A man-in-the-browser attack is designed to intercept data as it passes over a secure communication between a user and an online application. A Trojan embeds in a user's browser application and can be programmed to trigger when a user accesses specific online sites, such as an online banking site. Once activated, a man-in-the-browser Trojan can intercept and manipulate any information a user submits online in real-time. A number of Trojan families are used to conduct MitB attacks including Zeus/SpyEye, URLzone, Silent Banker, Sinowal, and Gozi. Some MitB Trojans are so advanced that they have streamlined the process for committing fraud, programmed with functionality to fully automate the process from infection to cash out. Additional capabilities offered by MitB Trojan developers include (RSA Lab, 2011):

a) HTML injection to display socially engineered pages (i.e. injecting a field into a page asking for the user's ATM card and PIN number in addition to their username and password) (RSA Lab, 2011).
b) HTML or JavaScript pop-ups to communicate with the victim in real-time (i.e., requests that the victim enters a valid one-time password or divulge answers to their secret questions) (RSA Lab, 2011).
c) Real-time interaction of Trojans with mule account databases to aid in the transfer of funds (RSA Lab, 2011).

The basic flow of a MitB attack is as follows (RSA Lab, 2011):

i. A user gets infected with an MitB Trojan
ii. Upon initiating an online banking session, the Trojan is triggered into action and launches its MitB functionalities
iii. The user passes all authentication stages, including two-factor authentication when needed. The Trojan waits silently for successful login and/or for the user to initiate a money transfer.

iv. The Trojan manipulates the transaction details such as the payee and the amount of the transfer. The legitimate payee is replaced with a mule account.

v. The Trojan maintains an apparently legitimate face of the transaction by using social engineering techniques. It displays bogus HTML pages to the user, which show the details of the legitimate transaction. If additional authentication is necessary to complete the transaction, the user proceeds to approve the transaction using whatever authentication method is required by the financial institution (RSA Lab, 2011).



Figure 2.1: Example of a transaction altered during man in the browser attack (Nilsson, 2012).

What makes MitB attacks difficult to detect from the bank's server side is that any activity performed seems as though it is originating from the legitimate user's web browser. Characteristics such as the Windows language and the IP address will appear the same as the user's real data. This creates a challenge in distinguishing between genuine and malicious transactions (RSA Lab, 2011).

2.3.2 Other Threats

There are also other threats that should be taken into consideration. This section will cover some of them. A "Man in the middle attack" occurs when someone manages to eavesdrop on web traffic by fooling the other part to connect to the attacker instead of connecting to each other as shown in Figure 2.2.

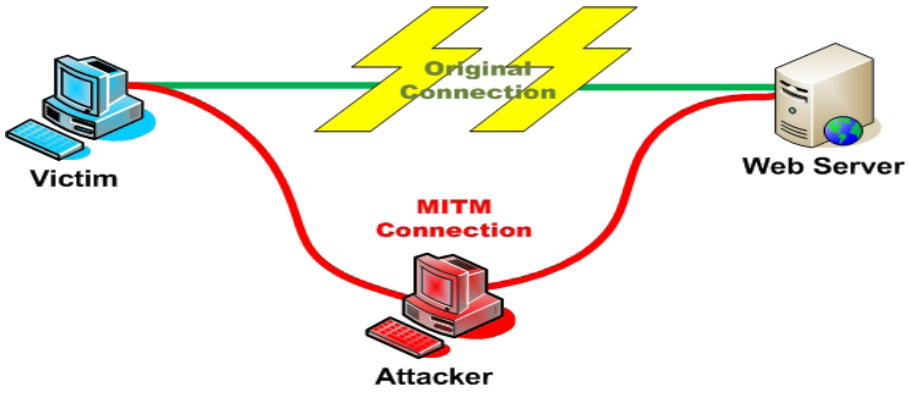


Figure 2.2: Visualization of a man in the middle attack (Nilsson, 2012).

One of the most common ways to counter MITM-attacks is to use a secure channel like Secured Socket Layer (SSL) when sensitive data is transmitted between the client and server. SSL verifies servers by using certificate authorities to guarantee that they are who they claim to be (Nilsson, 2012).

"Brute force attacks" implies trying every possible alternative until you find the right one. In this case it would mean to alter the XML document piece by piece until a match has been found. Precautions against these kinds of attacks would normally be to only allow certain amount of login attempts or to cause a delay after a failed login attempt or simply block a user after too many failed attempts (Nilsson, 2012).

"One click attacks" or "cross-site request forgery" is an attack where an unknowing person that has been authenticated by a web page is used to send unwanted requests to the server.

"Cross site scripting" implies injecting harmful code into the server with the user input.

While one click attacks focuses on the trust the server has for an authenticated user, cross site scripting focuses on the trust that a user has for a specific website (Nilsson, 2012).

Normal ways to counter one click attacks are to add unique form-keys to either every request or to each session or checking the HTTP referrer to find out the origin of the request. Cross site scripting is usually countered by sanitising user inputs and having rules and clear definitions of what user input should look like (Nilsson, 2012).

A “web replay attack” is a method of exploiting a captured packet or packets, and resends that traffic to cause unexpected or unwanted behaviour from the server. If the server does not

detect the reused data and accepts the repeatedly transmitted packets, the attack is successful. If an attacker would come across the data from a victim that is generated by the JavaScript, it would be possible to log in as the victim without the server noticing any difference. The data is usually gathered either by listening to the traffic or by installing malicious software on the victims computer (Nilsson, 2012).

Usually web replay attacks are prevented by using some kind of session token to make each connection different, for instance hashing with different keys or using timestamps.

Using secure channels like Secured Socket Layer (SSL) or Transport Layer Security (TLS) is also recommended as it makes it a lot harder to access the information needed to perform these kinds of attacks (Nilsson, 2012).

2.4 Security Features

2.4.1 SALT

A salt is a random string or value that is combined with the data that is going to be hashed. It is random data that are used as an additional input to a one-way function that hashes a password or passphrase. In a normal case, a salt is used to increase security with hashed passwords (Nilsson, 2012).

Below is a simple example of how hashing with a salt with the length 10 is performed.

```
salt = rand(10);
hashed_value = hash_function(salt + original_value) (Nilsson, 2012).
```

A new salt is randomly generated for each password. In a typical setting, the salt and the password are concatenated and processed with a cryptographic hash function, and the resulting output (but not the original password) is stored with the salt in a database. Hashing allows for later authentication while defending against compromise of the plaintext password in the event that the database is somehow compromised. The original intent of salting was primarily to defeat pre-computed rainbow table attacks that could otherwise be used to greatly improve the efficiency of cracking the hashed password database. A greater benefit

now is to slow down parallel operations that compare the hash of a password guess against many password hashes at once (Nilsson, 2012).

Cryptographic salts are broadly used in many modern computer systems, from Unix system credentials to Internet security (Nilsson, 2012).

It is common for a web application to store in a database the hash value of a user's password. Without a salt, a successful SQL injection attack may yield easily crackable passwords. Because many users re-use passwords for multiple sites, the use of a salt is an important component of overall web application security (Nilsson, 2012).

2.4.2 HASHING

Hashing is a method for creating a mathematical representation of a set of data.

The result of the hash function is called **hash** or **message digest**. For this type of problem, when one would have to verify that the data has not been altered, a cryptographic hash function is recommended. One of the reasons that these are used is that it is hard to reverse the hash process to acquire the original data (Nilsson, 2012).

To verify that the data is unchanged, a hash has to be generated on the server side and compared with the hash that was transmitted from the client. The client side hash would be submitted with the rest of the data to the server and compared with the server side hash. The idea is covered in where a cryptographic hash function is suggested to be used on both the transmitting and receiving end of the communication in order to verify the authenticity of a piece of data (Nilsson, 2012).

2.4.3 SESSIONS

A session provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user. Unlike communication without sessions, it is possible to implement states in the communication between the server and client. This means that the communication can be dependent of past events in the communication

between the server and client in contrary to stateless communication where the only thing that matters is the present request (Nilsson, 2012).

Sessions are ideal for storing sensitive data on the server side where it is safe from attacks. It also makes it possible to associate information with individual users making it possible to adapt the data and functionality to the user that is connected and grant users different permissions like in Figure 2.3 (Nilsson, 2012).

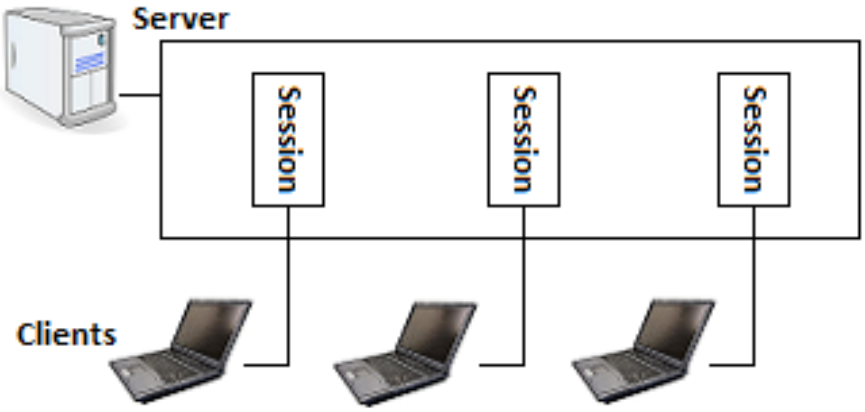


Figure 2.3: Illustration of a server with a separate session for each client where the data is stored (Nilsson, 2012).

2.4.4 DYNAMIC JAVASCRIPT

Sometimes one can benefit from using a dynamic JavaScript that looks different depending on who is using it. This could be useful when the script has to meet different requirements depending on who is using it and what permissions they have been given on the server. For instance, if two different users require different functionality from the script depending on what type of user they are. When using this technique the script is generated programmatically and can be submitted to the client or stored on the server side (Nilsson, 2012).

Session variables make it possible to make dynamic JavaScript that are unique for each client that connects to the server. This would make it more difficult for an attacker to use the data from an old session in a web replay attack since he would have to adapt to a different JavaScript than the session where the data was collected (Nilsson, 2012).

In this application one can change the JavaScript from each session so that the format of the XML document where the data is gathered is different each time someone uses the application. Changing the JavaScript every time would also make it more difficult for an attacker to analyze the script in order to be able to manipulate the data (Nilsson, 2012).

2.5 Technologies Used

2.5.1 Server Side Scripting Language

Server-side scripting is a web server technology in which a user's request is verified by running a script directly on the web server to generate dynamic web pages. It is usually used to provide interactive web sites that interface to databases or other data stores. The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores. From a security point of view, server-side scripts are never visible to the browser as these scripts are executed on the server and emit HTML corresponding to user's input to the page (Nilsson, 2012).

There are a lot of different server side languages. The languages that come up in almost every comparison are ASP.net, ColdFusion, JavaServer Pages (JSP), PHP, Perl, Python and Ruby (Nilsson, 2012).

2.5.2 Client Side Scripting Language

Client-side scripting generally refers to the class of computer programs on the web that are executed client-side, by the user's web browser, instead of server-side (on the web server). Client side script is a special script that is downloaded and executed at the users end. The response to interaction is faster once the program code has been downloaded. Services are secure as there is no access to sensitive files or databases and it is completely browser dependent and is affected by the speed of the user computer. Examples are VBScript and Javascript (Nilsson, 2012).

2.5.3 HTML

HTML is written in the form of HTML elements consisting of *tags* enclosed in angle brackets (like `<html>`), within the web page content. HTML tags most commonly come in pairs like `<h1>` and `</h1>`, although some tags, known as *empty elements*, are unpaired, for example `<img>`. The first tag in a pair is the *start tag*, and the second tag is the *end tag* (they are also called *opening tags* and *closing tags*). In between these tags web designers can add text, tags, comments and other types of text-based content.The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page. HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behaviour of HTML web pages. Web browsers can also refer to Cascading Style Sheets (CSS) to define the appearance and layout of text and other material. The W3C, maintainer of both the HTML and the CSS standards, encourages the use of CSS over explicit presentational HTML mark-up. HTML will be part of the language that will be used for this thesis (Nilsson, 2012).

2.5.4 MySQL

MySQL is a RDBMS (Relational Database Management System), which works based on the relational database model, to make database performance more quickly and flexible.

The SQL standard used in MySQL commonly means the current version of the SQL Standard at any time. MySQL software has dual-license with Open Source License. This means it is free for private use with commercial licenses that allows MySQL to be purchased by enterprises or people who want to embed the code into commercial applications. After all, with its rich functionalities, MySQL kept to be considered as one of the most famous and

reliable DBMS which occupies a large portion of database customer market (Canali and Balzarotti, 2012).

2.6 Literature Review

In the past, several protection mechanisms have been proposed to prevent man-in-the-browser attacks.

The IBM research team Weigold, (2008) introduced external device approach to prevent man-in-the-middle (MitM) variant attacks called **Zone Trusted Information Channel (ZTIC)**. The ZTIC is a USB device containing components such as simple verification display and other authentication components in order to provide secure communication between a client and a server. In this approach, ZTIC acts as a main medium between the client and the server without relying on any client's application or browser. ZTIC will scan and intercept any sensitive information and will only permit to exchange the information once the client has verified the information within its display component. However, the external devices requirement in the above solutions may become barrier for some users.

Jason *et al* (2008) proposed a design which adds another layer of security to existing methods to either prevent a MitM attack or to make the procedure of capturing and reassembling customer log on and transaction details more computationally and time intensive than what it is worth to an attacker. The model is based on a graphical authentication application previously developed called **Authentigraph** for preventing MitM attacks in authentication, data entry and transaction verification. When a client requests an authentication session with the server, an image is generated based on the associated process parameters which are a collection of characters randomly placed within the image. Each character is positioned randomly within its designated quadrant each time an image is generated.

Abbasi *et al* (2010) proposed a secure web contents to mitigate the MitM and the

MitB attacks. In their solutions, the web contents from a server will be encrypted by a secured web server and a secured proxy on the client side will decrypt the web content. Thus, this solution provides better protection of web content stored in the server. However, this solution concentrates only on the protection of a web contents on the server side and is lacking of protection against the malicious software attack on the client machine.

Sidheeq *et al* (2010) integrated the biometrics USB with the Trust Platform Module (TPM) in order to mitigate the risk of malware attacks on the client's machine. In detail, this solution requires the user to provide biometrics evidence for authentication. Then it compares the authenticity of the provided evidence with the user's biometrics stored in the USB which is protected by the TPM. In addition, this solution uses the encryption feature provided by the TPM in order to secure the data exchanged between a client and a server.

However, Fazli *et al* (2012) proposed a hardware-based authentication scheme to mitigate MitM and MitB attacks. Fazli *et al* provided trust communication between a client and a server as well as preserving the secrecy of the user's sensitive information. He incorporated **Trust Platform Module (TPM)** based remote attestation in order to provide the platform integrity verification. In addition, He adopted the **Secure Remote Password (SRP)** as the secure key exchange protocol in order to provide zero knowledge proof that allows one party to prove themselves to another without revealing any authentication information such as password.

Nilsson (2012) described the security research for a web application designed by **BehavioSec**. The application uses JavaScript to record keystrokes to generate data that is sent back to the server for verification. On the client side the data is gathered from the users typing pattern and finally the data is transmitted to the server where it is validated, extracted and processed. The time spent is also evaluated and stored in the database which shows the behavioural techniques that he incorporated in order to make it difficult for the attacker.

Rania *et al* (2012) presented a Mobile User Authentication System (MUAS) that used QR code technology to authenticate on-line users, through a challenge/response protocol.

The system proposed the usage of mobile devices, along with its cellular network as an Out-Of-Band (OOB) communication channel.

CHAPTER THREE

MATERIALS AND METHODS

3.1 Introduction

This chapter looks into the main research especially the proposed security system, the implementation with the functionalities of the system. The breakdown of the system architecture and all its components will be discussed. Furthermore, this chapter will explain the system architecture and its component together with the flow chart and as well demonstrate the working technique of the Anti-Form grabbing method of securing the web, how the JWT is securing the information exchange, how the OTP authenticates the client side for adequate security and also show how the email verification service will be employed. Finally, the design of the database shall be considered and also the enumeration of required tools and technologies during implementation.

3.2 The Proposed Enhanced Security System

In this section, the modelling strategy employed in achieving the proposed enhanced security system is considered. First, the system architecture and the flow diagram chart or the workflow shall be considered. Predominantly, during the implementation of the work, the workability of the security features will be actualised, but in this section, we seek to consider the theoretical functionalities of the various components of the enhanced security system. This contains two separate solutions which are discussed later on. One will be showing how the data would be gathered and transmitted in a login session and one showing how the transaction could be verified. PHP will be used as server side scripting language and JavaScript will be used for the client side scripting. Apache will be used as web server software. A MySQL server will also be used on the server machine.

Each of these technologies listed earlier has its respective part it played in realizing the proposed security measure for online financial web applications. Section 3.3 will discuss in detail the theoretical approach taken in archiving the system.

3.2.1 Functionalities of the System

The key features of the implementation are:

a) The client/user will initiate a transaction by login in his details.

b) During the login phase, a token generating algorithm generates a token and relays it to the user email.

c) When the client logs in successfully, a dynamic JavaScript will be generated from the server side which will now be sent to the client side.

d) Since a two-layered encryption is being proposed, item (c) does the first encoding while JSON Web Token (JWT) is used for the second layer encoding.

e) The client will send the data to the server for it to verify the information.

f) While the data is on transit, the data input will be automatically encoded using encryption algorithm.

g) When the client submits to the server side, the server interprets the data input based on the dynamic JavaScript sent to the client.

h) If the forms submitted are correct, the server sends a verification email through another third party mail server.

i) The client opens his email and authenticates the transaction by accepting the transaction details sent to the mail or declining the transaction.

3.2.2 System Architecture

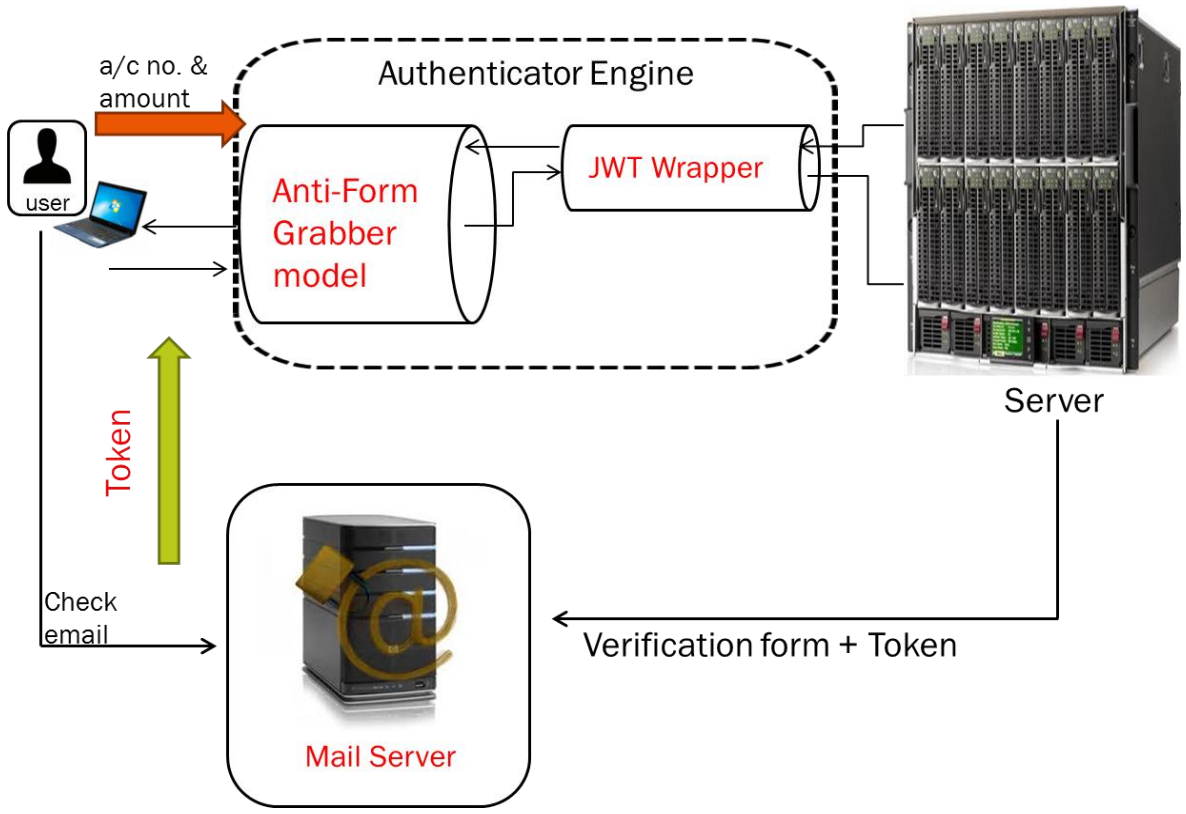


Figure 3.1: The System Architecture.

a) **User:** The user is the person that makes use of the client side and initiates the transaction. The user logs in his details and sends it to the server for authentication. It provides the interface for the user to be able to communicate with the server side like data inputs, submission of forms and so on.

b) **Authenticator Engine:** It is the security layer that consists of Anti-Form grabber model and JWT Wrapper.

   i. **Anti-Form Grabber Model:** What this model does is that all the data input typed in by the user is automatically encoded into unknown characters making what is displayed on the screen different from what is being typed.

   ii. **JWT Wrapper:** JSON Web Token is used to send information from the user which will be verified and trusted by means of a digital signature. It breaks down the information into three parts which are

header, payload and signature which generates their own unique token at the background before being sent to the server.

c) **Server Side:** The server side comprises of three components, which consists of the PHP script engine, JavaScript creator and the email handler. This is the power house of the system which connects all the three. It is the driving force of all the three components.

i) JavaScript Creator: This generates the dynamic JavaScript that will be sent to the client side which manages the user input. It creates a JavaScript code based on the session value of the user who has logged in. After creating the JavaScript, it will be sent to the client side along with the form. When it is sent to the client side, the user fills the form and sends it back.

ii) PHP Script: This server side scripting is used to generate the web pages which analyse the data when transmitted to the server and also communicates with MySQL Server. It automatically engineers the creation of the JavaScript.

iii) Email Handler: This is the aspect that handles email. Both the mail that is sent to the user and the confirmation mail from the user are handled by this component.

iv) Database: All the information is being stored on the database.

d) **Mail Server:** This is like a middle-man between the server side and the client side. It's the one that receives the mail from the server side and delivers the mail to the client side and vice versa. The server side sends the authentication mail to the user's email and he checks it, confirms it and sends it back to the server to confirm authentication. This is actually known as the mail server. The verification is rerouted through the email instead of the same browsing session as part of preventive measures for MitB.

e) **Token (OTP):** OTP which is the short form for One Time Password also known as token is a kind of password that is valid for only one login session or transaction. It

is generated from the server from the time the user successfully logs in and is being sent to the user's email for the user to be authenticated. The user checks his email for the token and he fills it in together with the transaction details and sends it to the server.

3.2.3 System Flow Chart

The flow chart captures the flow of information in the system. As soon as the user gets to the system, he must login with his login details and wait for authentication. The server will check whether the authentication is valid and if it is, it will now generate JavaScript and after generating it, it will now attach the JavaScript generated to the form that will be sent to the user. During the time the form is sent to the user, a token is generated and sent to the user's email for authentication process. When the form is sent to the user, it is in that form that the user will fill all the transaction details together with the token details and send it back to the server. The server checks for session variable in order to decode what has been filled in the form, then process the transaction and sends email to the user for verification. After sending the email for verification, the user opens his email box and confirms it because the transaction won't be complete unless the user confirms the transaction. If the user doesn't confirm the transaction either by not taking any action in time or he clicked on "decline", the transaction will be cancelled and it will return back to the user filling the form and submitting it again. But if the user confirms the transaction by clicking on "accept", then the transaction has completed successfully. So between the generation of the JavaScript and the user filling and submitting the form is where the Anti-Form Grabbing technique comes in. The generated JavaScript that is attached to the form is what makes the data that the user types in the form be encoded in such a way that is not readable before the form is sent to the server. So if the attacker "grabs" the form, he ends up stealing meaningless information.

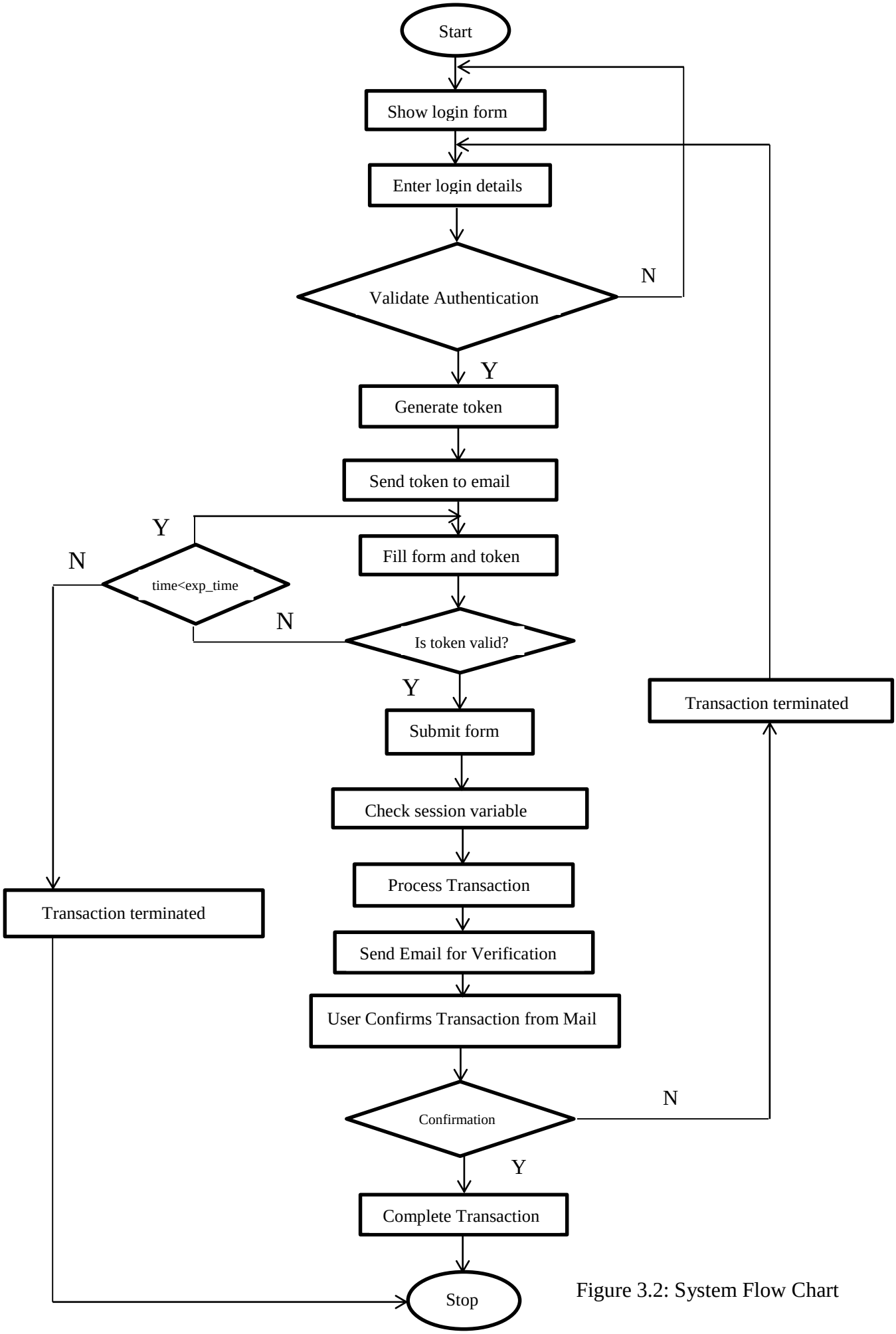


Figure 3.2: System Flow Chart

3.3 The Security Model Mitigating MITB

In this session, the technique employed in mitigating Man-In-The-Browser attack through the Anti-Form Grabbing Technique and the use of Token will be considered and also how the Email Verification Service is going to be used.

3.3.1 The Anti-Form Grabbing Technique

Sensitive information which contains data fields that are labelled Password, Username, Account Number and the Amount being entered will be encrypted. Whatever the user fills in the form will be sent to the user's email and when the user opens his email, the user confirms the transaction and sends it back to the server which now verifies the transaction.

In this technique, a random generation number system is needed with level randomization scheme, an encryption algorithm, PHP session variable and a pool of multilingual characters to make the system work. Once the user logs in, his login details in transit to and from the server will be encrypted automatically to thicken the security measure. After the user has logged in successfully and the user is authenticated, the PHP session variable value for that specific user is attached to that user which will now be stored at the server side. The specialize random number generation algorithm will now be used to generate 62 characters (A-Z, a-z, 0-9) from a pool of 100 characters for each user. The random numbers will be generated from 1to 62. For example, assuming a character pool of 19 is generated from the multilingual characters that are there, it will pick a character at position 19 in the pool of our overall multilingual. So if a user logs in, the 62 characters generated randomly will be attached to that user and for another user that logs in, also another set of 62 characters will be generated for its transactional purpose. The idea behind this different alphabets system among different users of the same system is to make it more difficult for hackers who may want to exploit security weakness through MITB to compromise the transaction. Here, since every user has a continual changing alphabetical system, it will be difficult for a sniffer to observe the alphabetical system with a notion of compromising the system in subsequent

attack. Also, even if the attacker successfully hijacks transaction data in transit, what it would have captured will be a sequence of multilingual character that may not make any reasonable meaning except it is decoded. The encoding for the alphabet system of a given user is done using the random number scheme described above and embedded in dynamically generated JavaScript. So this shows if two or more users are accessing the website at the same time, their number systems are different. So even if the attacker grabs the users' forms, he will notice that they are different number systems, so he can't make meanings out of their details filled in the form which is why it is called Anti-Form Grabbing Technique. So the attacker is confused over a range of users since each user has a different number system. Even if the attacker tries the attack another period for the same user, the number system will be different from the other time he earlier attacked. So with this format, the multi-lingual alphabet system will be generated and sent to the user along with the form but it will be encoded in form of JavaScript. The mathematical model used for the salt is shown below in equation 3.1.

x=rand(): 1-1000

$$fRand = \begin{cases} X \leq 100 \\ X \geq truncate\,hundredth\,leaving\,tens\,and\,units \end{cases} \quad 3.1$$

In the implementation of this mathematical model, a number between 1 to 1000 was picked and the fRand function returns x if x is less than or equal to 100 but if it's greater than 100, it will return x but truncate the hundredth leaving only tens and units, e.g. with 399, it will truncate the '3' which is the hundredth leaving only '99' which consists of tens and units.

Algorithm 1: Algorithm for Anti-Form Grabbing Technique

1 Show login form
2 Generate token
3 Send token to user's email
4 **if** token is correct and on time,
then
5 Read character pool $P = \int_{(alp)} + C_{(alp)}^{c} + G_{(alp)}^{c}$
6 Get session ID
7 let Web App alphabet system be $\mathcal{E}_{(Alp)} \equiv$ [A…Z, a…z, 0…9]
8 **for** *i* range from 1 to 62
$\mu(p)_i = P$[fRand(.)]

where $\mu(p)_1 \ldots \mu(p)_{62} \in \varepsilon_{(Alp)}$

9 **foreach** $\mu(p)_i$ mapped into $\varepsilon_{(Alp)i}$

10 Create dynamic JavaScript.

- Encode all $\mu(p)_i$ into the JavaScript.
- Denote each $\mu(p)_i$ with its corresponding $\varepsilon_{(Alp)i}$ on the client side.
- Tag dynamic JavaScript with user session ID.

end loop

11 Send transaction form with dynamic JavaScript.

12 **if** user sends form back

then

i. Decode user inputs by mapping all $\mu(p)_i$ back to $\varepsilon_{(Alp)i}$

ii. Send verification email

iii. **if** email is confirmed, **go to** step 13

else wait

13 Complete transaction based on user **agree/disagree** option through mail

14 **end** transaction.

else

15 go to step 1

Algorithm 1 is the one that generates the anti-form grabbing technique that encodes the characters that are entered. After successful login from the user, the form from the server is sent to the client where the user can enter the destination account and the amount the user wants to send. During this time, a one-time password otherwise known as token is being generated from the server and sent to the user's email. The user checks the email for the token and types it correctly and on time together with the destination account and the amount. All this happen from step 1 to 4 then the anti-form grabbing technique begins its operation by reading the character pool in step 5. The character pool denoted by *P* consists of three operands which contains three categories of symbols used for mapping. Then move to step 6 in which a session ID is given the user online. Step 7 contains the alphabet system ($\varepsilon_{(Alp)}$) that will be used which consists of upper case and lower case of the English alphabet A to Z then number digits 0 to 9. All the characters in $\varepsilon_{(Alp)}$ added together gives 62 in which a pseudo character is used for it and this pseudo character is going to be drawn from step 5 which will now be mapped to $\varepsilon_{(Alp)}$. In step 10, a dynamic JavaScript is created where each of the pseudo characters are encoded and mapped to corresponding real alphabet system ($\varepsilon_{(Alp)}$) which is

tagged to the user session ID to make it unique. In step 11, transaction form is sent to the user and when he fills it and sends it back to the server in step 12, then, the user inputs will be decoded by mapping the pseudo characters back to their real alphabet form for it to be readable then sends the verification to the user's email. When the transaction is verified, go to step 13. In step 13, transaction is completed based on user agree/disagree option provided in the email.

3.3.2 Token Generation

This session will give the explanation on how the token is generated to create the one time password for the authentication of the user.

Algorithm 2: Algorithm for Token Generation

```
BEGIN SUBPROGRAM
    OTP (salt="", n_d)
    if salt=""
        TS= randomly select integer value
        TC=(unixtime(now)-unixtime)/TS
        salt=hex(TC+rand(5))
    end if
    int i=0;
    encrypted=sha512(salt)
    token=" ";
    while i < n_d
        token += encrypted[i]
        i++
    end while
    return token;
END SUBPROGRAM
```

OTP: One Time Password

n_d: number of digits

TS: Time Step

TC: Time Counter

hex: hexadecimal

Algorithm 2 is the one that generates the one time password which takes a variable which is salt. If the salt value is parsed into it already, it's not going to be empty but in the algorithm above, the salt value is empty because it's not parsed into it which makes us create our own salt value. Therefore we use TC as a process of creating the salt and we get TC by subtracting the original unixtime from the present unixtime and divide it by TS. Unixtime is known to be the number of seconds elapsed since January 1 1970. After this, we get the hexadecimal value of adding TC and the random value between 0 and 5 then store it in salt. We now get the encrypted value of the salt by hashing it with SHA 512. So in order to get the digits that make up the token, n_d which is number of digits will be picked from the encrypted value randomly one after the other and it will be added to the token to make up the digits that will be used.

3.3.3 JSON Web Token (JWT)

JSON Web Token is a compact URL-safe means of representing claims to be transferred between two parties. These set of claims are encoded in JWS enabling the claims to be digitally signed or Message Authentication Coded (MACed). It is used to send information that can be verified and trusted by means of a digital signature. Because of its compact structure, JWT is usually used in HTTP Authorization headers or URL query parameters.

The output of the encoded value of the anti-form grabber model is sent into the JWT for a further encoding and encryption. This guarantees that a thicker security layer is wrapped around the transaction information to ensure that it is not tampered with.

3.3.4 Email Verification Service

After the Anti-Form Grabbing Technique has done its job, an email is sent from the email handler in the server to the user's email box to verify the transaction whether it is valid or not. This is as indicated in Algorithm 1.

This is relatively out of the reach of the attacker since he cannot monitor both the transaction and the email verification option at the same time. Moreover, note that user is expected to respond to email alert within a given time, else the transaction will be rendered invalid. Two Universal Resource Locator (URL) links will be sent to the clients email in order to confirm whether the transaction is valid or not. If it is valid and correct, the user will click the link that agrees with the transaction details sent to the mail. Else the user can decline the transaction by clicking on the "**disagree**" URL link.

3.4 Theoretical Evaluation of the Security Model

The goal in this section is to make some analysis that may look theoretical and prove that the proposed security model for this research is tenable in mitigating MITB. First, the token generated is further used in authenticating the use through email and to as well ensure that the proposed transaction is genuinely from him. Once the user is authenticated through the token sent to the email, then the next phase of the model is the Algorithm 1. The token value is passed along with the transaction details. Looking at Algorithm 1, we could categorise the approach to tackling the online transaction compromising trojan into the pseudo alphabet system generation, dynamic JavaScript creation, client side encapsulation of keystrokes and lastly the email verification code.

Firstly, the pseudo alphabet system generation is considered. The word pseudo in this contest connotes a logical facade of the conventional English language characters or the United States and United Kingdom based keyboard characters. The focus of this research was to abstract user inputs in such a way that even if the sniffer successfully sniffs on the data as it is been carried between the server and the client, or if the Man in the Brower seeks to manipulate the user inputs, the information these prospective attackers will eventually capture will be a junk of characters that does not provide any information. Our major concept is to create a heterogeneous pool of character set – selected and distinct characters drawn from

three different categories of symbols - and then a well-coordinated random number generation system is used to compose a pseudo alphabet system generation per user. PHP is used in the implementation in other to realise this.

Looking at line 5 in the Algorithm 1, the specification is seen for the pseudo alphabet system that will be generated for each user based on the session ID. The three operands on the right hand of equation 3.2 are the characters selected from the three categories of symbols mentioned above.

$$P = \mathcal{F}(\text{alp}) + \mathcal{C}(\text{alp}) + \mathcal{G}(\text{alp}) \tag{3.2}$$

The motive behind equation 3.2 is to narrow the possibility of breaking through the solution being proffered by this research. Drawing characters from these three categories, makes our pool of alphabets a different symbol and character and as well difficult to be compromised. In line 6 ("Get session ID") is the session of the user unto which unique automatically generated pseudo alphabet system will be stored and identified with the user session ID. Maintaining the fact that we already know the conventional English characters, we proposed to represent them with equation 3.3. Now this is concluded that the automatically generated pseudo alphabet system will be made up of 62 characters. And as these 62 characters are going to be drawn from the characters of the three categories indicated above, a random number generation scheme is employed in choosing the specific pattern in which these characters are going to be drawn and assembled into the automatically generated pseudo alphabet system. This random number generation scheme, denoted by fRand(.) is included in what is in equation 3.3.

$$\mathcal{E}(\text{Alp}) \equiv [\text{A} \ldots \text{Z}, \text{a} \ldots \text{z}, 0 \ldots 9] \tag{3.3}$$

$$\mu(p)i = P[\text{fRand}(.)] \tag{3.4}$$

The first phase of the solution to mitigating MITB was the method described in previous paragraphs, the creation of pseudo alphabet system generation. The second concept employed

is the use of dynamic JavaScript for encoding the pseudo alphabet system created. This idea entails that for each user that logs unto this site, the unique pseudo alphabet system created for the user is stored in a PHP session variable. Then a JavaScript file was created to abstract user's keystrokes. Since in the pseudo alphabet system a character have been mapped to a corresponding character in the conventional English language based on their index. For example, if ***ă*** were to be a character at position 3 in our pseudo alphabet system, then, the corresponding alphabet that is mapped to it in the English language alphabet will be ***c***. Once this JavaScript file is created, it is attached to the server's response to the client side. Now, this where the client side keystrokes encapsulation concept comes into play. On the client side, when the user strikes ***c***, following the example above, the JavaScript will replace the character that will be displayed to be ***ă***. Since MITB hijacks user inputs using the form grabbing technique, then we conclude that grabbing such encoded input as indicated by this research will be meaningless to the attacker. This is to make even the JavaScript code quite secure so as to frustrate the attacker from breaking the algorithm suggested in this research.

The last functionality used to curtail MITB as proposed by this research is the use of email verification scheme. Here, the user will be required to respond to an email sent to his email. This is aimed at using a completely new channel for the transaction authentication.

All these are captured in the line 12 of algorithm 1. This is shown as follows.

if user sends form back
then
i. Decode user inputs by mapping all $\mu(p)_i$ back to $\varepsilon_{(Alp)i}$
ii. Send verification email
iii. **if** email is confirmed, **go to** step 13
else wait

3.5 The Database Design of the Proposed System

The proposed web application that the security model mitigating MITB is deployed into will work with an underlying database. This database was designed using MySQL database system. And the entity relational (ER) diagram was drawn using MySQL

workbench. Majorly, the database consists of five tables, this include *userinfo, account_type, transaction_type, account_info, login_details* and *transactions*.

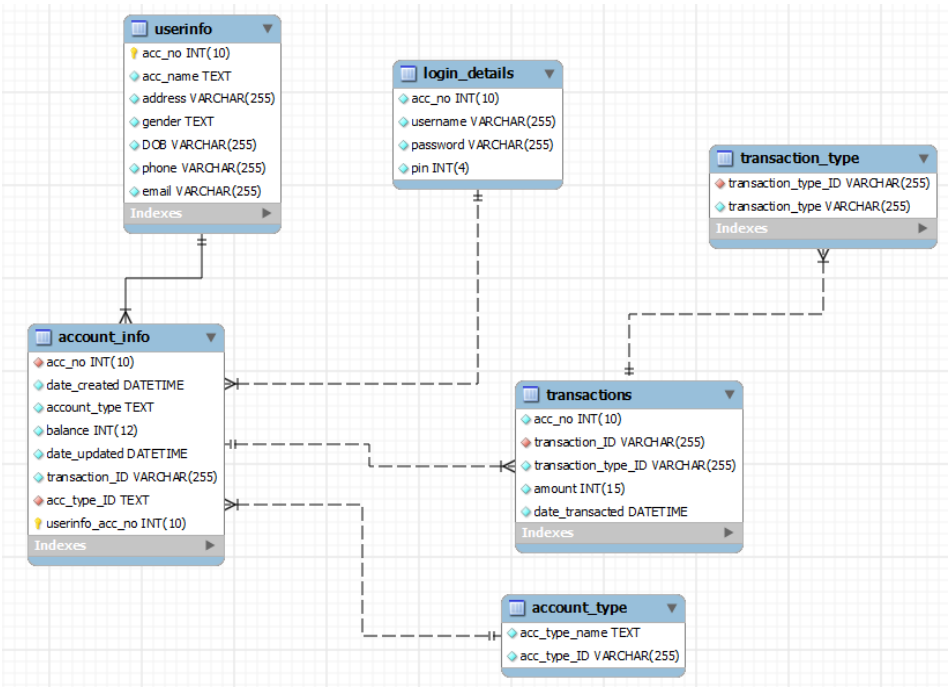


Figure 3.3: Entity Relational Diagram.

The *userinfo* consists of the information of the user. This contains the ***acc_no*** which serves as the account number of the customer and also the foreign key referencing ***acc_no*** on table *account_info*, and the ***acc_name*** which serves as the name of the customer who owns the account. Other important fields of the user information are the ***address*** which can be the office or house address of the customer, ***gender*** which indicates whether male or female, ***DOB*** (Date of Birth) which specifies when the customer was born, ***phone*** which serves as the phone number of the user and ***email*** which serves as the email account of the user and also the primary key. The *account_info* contains the information about the account which consists of ***acc_no*** which serves as the primary key, ***date_created***, that is the date the account was created, ***account_type*** which specifies whether its current account or savings account, ***balance*** which shows the amount that remains in customer account, ***date_updated*** which shows the date that a transaction was carried out against an account, ***transaction_ID*** which serves as the identification for proof of payment to the user**,** and ***acc_type_ID*** which serves

as a unique identification of the type of account and the foreign key. The *account_type* consists of ***acc_type_name,*** that is the name of the account type and ***acc_type_ID*** which serves as the primary key.

The *login_details* also contains ***acc_no*** which is the foreign key with other three personal information like ***username*** which serves as a unique name of the user used to gain access to the user's financial account , ***password*** which is a string of characters used for user authentication and ***pin*** which serves as combination of numbers used as an additional password to access the user account and the primary key as well. The *transaction_type* contains information about the type of transaction being handled which are ***transaction_type_ID*** which specifies the unique identification of the type of transaction and also serves as the primary key and ***transaction_type*** which also specifies the type of transaction. The *transactions* consist of **acc_no, transaction_ID** which serves as the foreign key that references **transaction_ID*, transaction_type_ID, amount*** which serves as the amount of money used for the transaction and ***date_transacted*** which serves as the date of the transaction**.**

CHAPTER FOUR

IMPLEMENTATION AND DISCUSSION

4.1 Introduction

First of all, the code written for this work especially the algorithm will be implemented. The information flow between the client and the server will also be encrypted and also encoded in JWT. The session variable will be discussed as well as the function of the Email Authentication Handler. How the dynamic java script was created will be shown and screen snapshots of our intended program will be provided with the results being discussed.

4.2 Code Implementation

The codes for the anti-form grabbing, JWT and the OTP will be implemented in this section. It involves converting the theoretical design specifications outlined in chapter three into executable programs. PHP cum JavaScript/jquery programming languages are used to write the codes.

4.2.1 Coding the proposed algorithm

The proposed algorithm is based on the implementation of the anti-form grabbing which is used to encode the inputs being typed, into characters chosen from the pseudo alphabetic system that was formulated in chapter three. There is also the algorithm of the One Time Password (OTP), and that of the Email Verification were combined together to further strengthen the security of the online transaction, thereby producing an enhanced security framework for the online banking system. These are being implemented which is used as a strong security for verification process of the transactions. All the algorithms were coded with PHP because it’s a server side scripting language.

```
//Now, begin to generate pseudo alph
$muin=array();
$real_muin=array();
$unzip_pseudocalp=" var unzip_pseudocalp=new Array();";
$unzip_realalp="  var unzip_realalp=new Array();";

for($i=0; $i < 62; ++$i)
   {
     //how to generate a random number from number 1 to 1000 in php
     $ran_btw_1_1000=rand(1,1000);
     $p=$this->getFRand($ran_btw_1_1000);

     do
       $temp_char=$this->getMultiLingualAlphbeth($p);
     while(in_array($temp_char, $muin));

     $muin[$i]=$temp_char;
     $real_muin[$i]=$all_char[$p];

     $unzip_pseudocalp.="  unzip_pseudocalp[".$i."]= ".$muin[$i].';  ';
     $unzip_realalp.=" unzip_realalp[".$i."]= ".$real_muin[$i].';  ';
   }

 $_SESSION['m']=implode(',', $muin);
 $_SESSION['rm']=implode(',', $real_muin);

 $this->encode2JavaScript($unzip_pseudocalp, $unzip_realalp);
}
```

Figure 4.1: Pseudo Alphabet System Creation.

Figure 4.1 is an encoding for the pseudo alphabet system designed in this thesis. Particularly the first two lines are creating empty arrays that will hold it in PHP while the next two lines are also creating empty arrays that will hold it in JavaScript. The ***for*** loop that follows is the creation of all the alphabets and assigning them to their various arrays for storage which will later be retrieved in the program for use.

```
function getTokenGenerator($salt="")
{
   if($salt == "")
      {
        //Converting it to a unix timestamp
        // Create timestamp for today at 00:00:00
        $unixtimenow = mktime('0', '0', '0', date('n'), date('j'), date('Y'));
        $unixto = strtotime('1970-01-25 14:35:08' ); //mktime('0', '0', '0', da
        $unixstep=4;

        $tc=($unixtimenow - $unixto)/$unixstep;
        $salt=hexdec($tc+rand(5, 1000));
      }
   $hash=hash('sha512', $salt);
   return substr($hash, 0, 8); //.rand();
}
```

Figure 4.2: Token Generation Algorithm.

Figure 4.2 describes the implementation of the token generation algorithm. The function "***getTokenGenerator***" is called in order to create the salt value since the value wasn't sent

automatically during creation. It will be the value that will be used in hashing the characters in the hash function. In order to get the actual time of the token generation to its expiration, the unix value of the current time “unixtimenow” and the step “unixstep” is gotten. So it will be calculated as shown in figure 4.2 which will be the time counter (tc). A random value between 5 and 1000 is gotten and added to the time counter which will now be converted to an hexadecimal value. The hexadecimal value becomes the salt which is what is passed to the hashing algorithm to be able to get the token being sent. The last line returns 8 characters of token which will be sent to the user. In the reason behind the eight-character-token is to make in tandem with normal usage of token.

```
include 'PasswordHash.php';
$password=uniqid();
$raw=create_hash($password);
$hash=str_replace('+', '_',str_replace(':', '_',
    str_replace('/', '_', $raw)));
$hash=substr($hash, 7);
$token=$hash;

include './PHPMailer.php';                                    //Create a
$mail = new PHPMailer;                                                //
$mail->isSMTP();
$mail->SMTPDebug = 2;
$mail->Debugoutput = 'html';
$mail->Host = 'smtp.gmail.com';
$mail->Port = 587;                                            //Set the
$mail->SMTPSecure = 'tls';
$mail->SMTPAuth = true;
$email=$acc_info[email];
$mail->Username = "capitalbank2015@gmail.com";
$receiver_info=mysql_query("select * from  userinfo where
        acc_no='{$acc_num}'") or die(mysql_error());
$receiver=mysql_fetch_array($receiver_info);
file_put_contents('contents.html',
     formatEmailMessage($sender_name, $receiver[acc_name],
     $bank_name, $receiver[acc_no], $required_amount, $token), LOCK_EX);
$mail->Password = "endowd247";
$mail->setFrom('capitalbank2015@gmail.com', '');
$mail->addReplyTo('info@citibanks.com', '');
$mail->addAddress($stemail, '');
$mail->Subject = 'Account Verification';
$mail->msgHTML(file_get_contents('contents.html'), dirname(__FILE__));
$mail->AltBody = "You have requested to transfer {$required_amount}
   to acount number {$acc_num} in bank {$bank_name} ";
if (!$mail->send()) {
```

Figure 4.3: Mail Verification Sender.

In Figure 4.3, a class was instantiated known as “***PHPMailer***” so all the functions called are in this class. So all these functions are correspondingly being called and used, these are done during the process of sending the confirmatory email to the user.

4.2.2 Encrypting data in transit

The main data that is encrypted in transit are the “Account Number” destination and the “amount”. This sensitive information is not only be encrypted but also show unknown character symbols during the encoding process when it is being typed in. The JSON Web Token will take care of the web content as well which is the “Account Number” and the “Amount” in the entry fields. The information is sent through JWT as part of the headers to the server.

```
            $cnt005 = "";
            $user=$_SESSION['username'];
            $selectquery="select * from userinfo WHERE username='$user'; ";
            $qexe=mysql_query($selectquery)or die(mysql_error());
            $cnt005 = "";
  while($datapool=mysql_fetch_array($qexe)) {
  $cnt005 = $datapool['skey'];
  $stemail = $datapool['email'];
  $acc_num = $datapool['acc_no'];
  break;
  }

if($key2==""){
    $_SESSION['error']='Provide a Valid Secrete Code 22';
                header('Location: ScreenShot2Valid.php');
                exit;
}

if($cnt005 != "" && $cnt005 == $key2){
        $expire = time()-3600;
 // 7 days; 24 hours; 60 mins; 60 secs
$token = array(
    "iss" => "http://example.org",
    "aud" => "http://example.com",
    "iat" => 1356999524,
    "nbf" => "$expire"
);
$jwt = JWT::encode($token, $key2);
$val = 0;
$decoded = JWT::decode($jwt, $key2, array('HS256'));
            include 'PasswordHash.php';
```

Figure 4.4: Encryption of Transaction Details via JWT.

Figure 4.4 encodes the encryption process of JWT on the transaction details. The required parameters are:

iss: The issuer of the token

sub: The subject of the token

aud: The audience of the token

nbf: Defines the time before which the JWT MUST NOT be accepted for processing

4.2.3 Email authentication handler

This is the aspect that handles email. This verification is done for both the authentication value stored on the database and the authentication value sent through the mail that is sent to the user, which is now being sent back to the website, hidden under the link that will redirect the user from its email to the bank's website.

4.3 Discussion of Results

In this section, how the transaction was carried out with the screenshots of this work will be discussed. Also the summary of the work of the literatures that were reviewed will be analyzed.

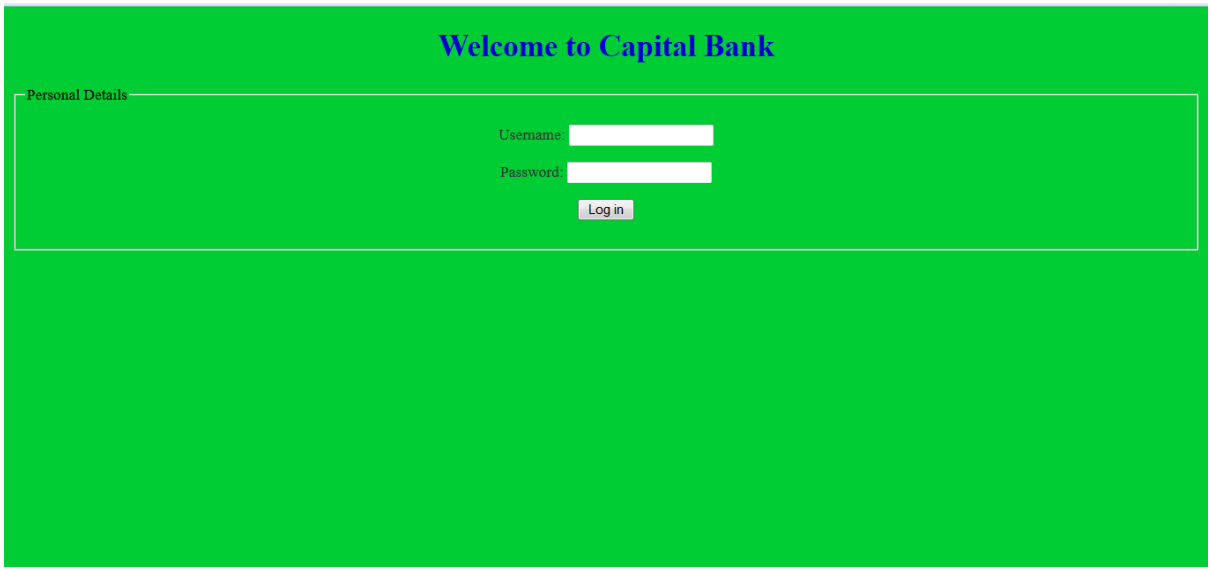


Figure 4.5: The Login Form.

Figure 4.5 shows the login form when the user is about to login to the financial website. Here, the user inputs his username and password in order to gain access.

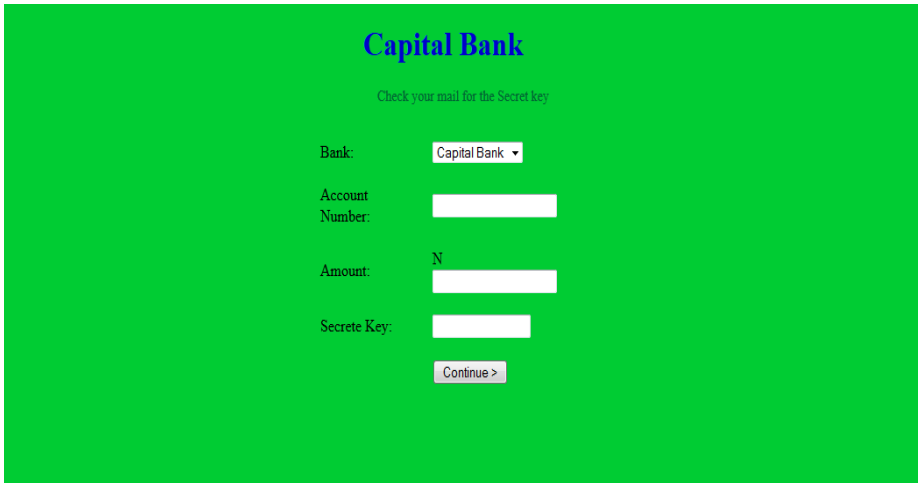


Figure 4.6: Entry Form.

Figure 4.6 is the entry form in which the user enters the necessary details such as the "Account number", "Amount" and the "Secret Key"

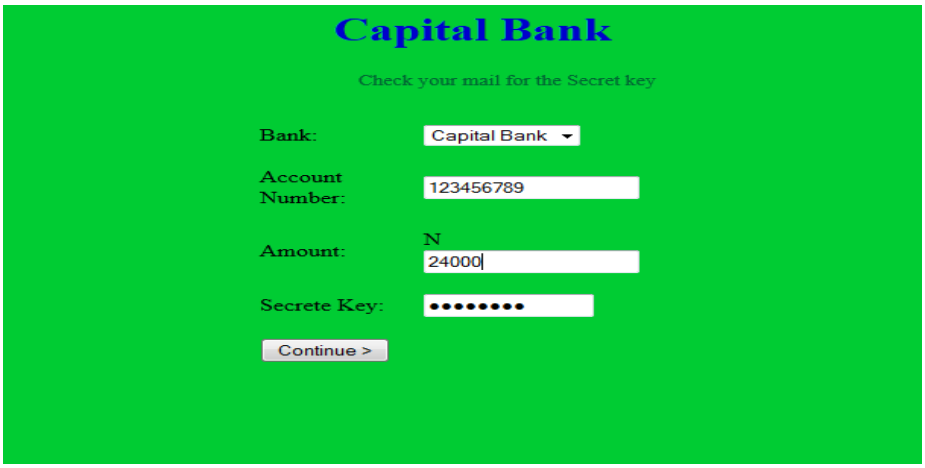


Figure 4.7: Entry Form without Anti-form grabbing.

Figure 4.7 shows the entry form without the anti-form grabbing which leaves the transaction details exposed and risks the information being grabbed by the attacker.

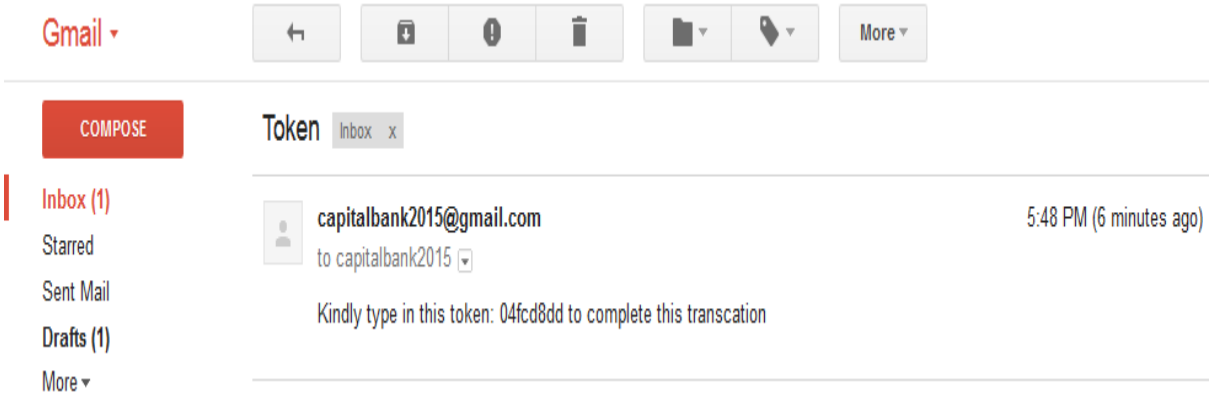


Figure 4.8: Token Verification.

Figure 4.8 shows the Token verification which is being sent to the user's email immediately the entry form is generated and shown to the user for input.

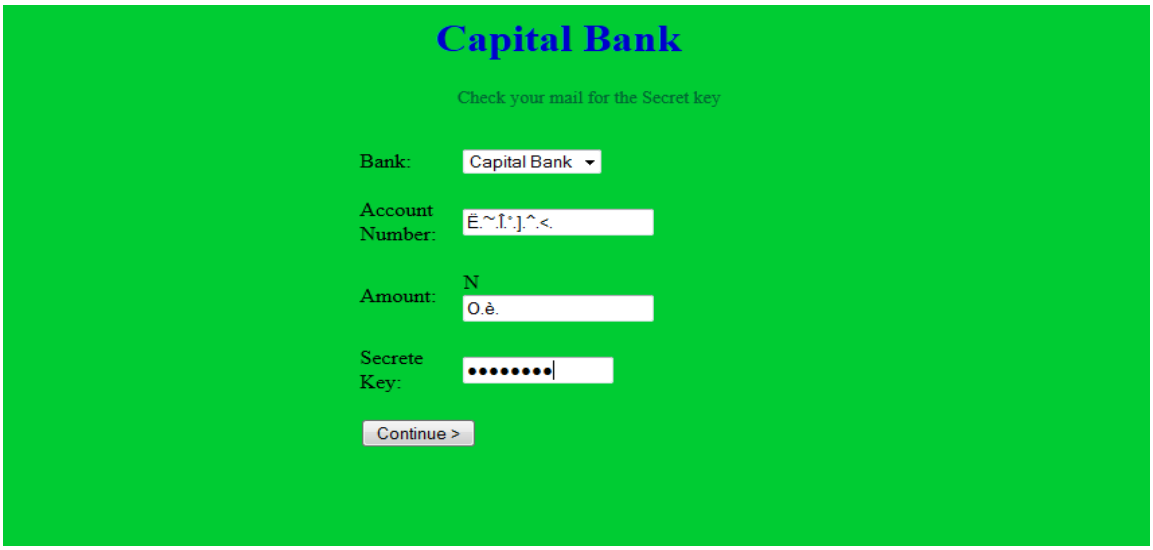


Figure 4.9: Entry Form with Anti-form Grabbing.

Figure 4.9 shows the entry form with the anti-form grabbing which encodes the transaction details into unknown characters which will disallow the attacker from grabbing the information. The secret key gotten from the email will be entered here as well after that the "Continue" button will be clicked in order to continue with the transaction.

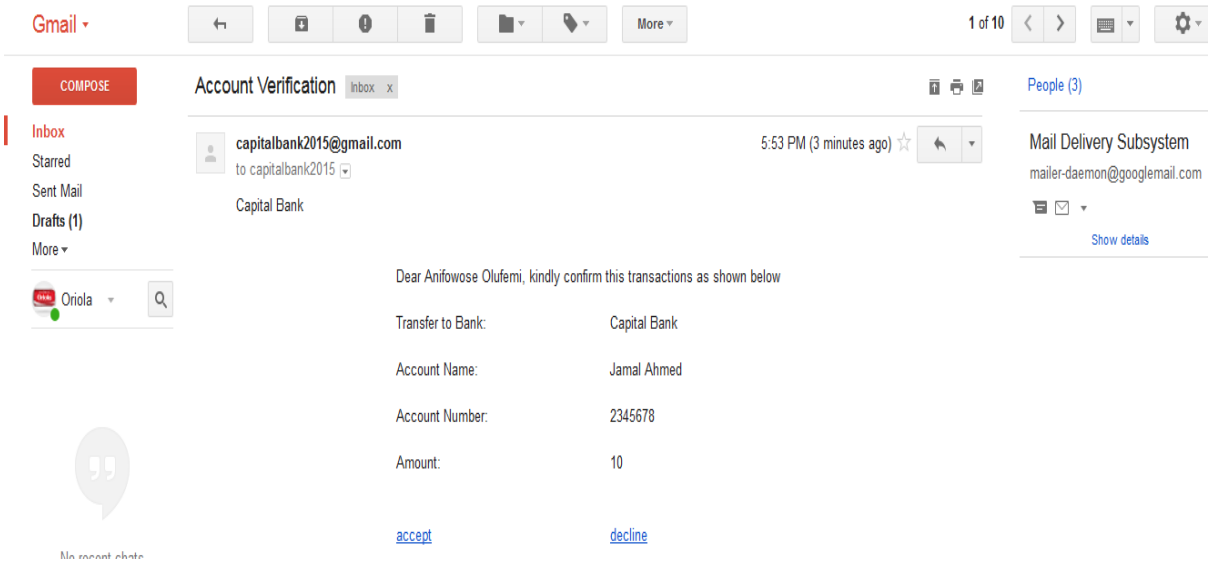


Figure 4.10: Account Verification.

Figure 4.10 shows another email being sent to the user but this time verifying the transaction details whether there is any error or it has been tampered with. If the transactions are in line with what the user typed, two links have been provided in order to accept the transaction or decline the transaction.

4.4 Model Comparison Analysis

Table 4.1 shows the various comparative model analysis of some of the literature reviews done with our own work.

Table 4. 1: Model Comparison Analysis

S/N	Review	Web Content Protection	Form grabbing	Transaction verification	Client protection
1	ZTIC(2008)	YES	NO	YES	YES
2	WSAD(2010)	YES	NO	YES	NO
3	MUAS(2012)	YES	NO	YES	NO
4	SLOTP(2013)	YES	NO	NO	YES
5	EWSA(2015)	YES	YES	YES	YES

The parameters chosen to compare them with are Web Content Protection, Form Grabbing, Transaction Verification and Client Protection.

Web Content Protection is the content of the transaction (important information) transferred from client to server and vice versa.

Form Grabbing is the hijacking of important information in a form by the attacker before its being delivered to the server.

Transaction Verification is verifying the transaction that the user initiated from the client side whether it is the same or different client.

Client protection is protecting the information from the client side like username, password, account destination and amount and other confidential data.

In the first review which is **ZTIC (2008)**, web content protection was provided due to the fact that the USB device consists of components such as verification display and other authentication components in order to provide secure communication between the client and the server. These facilities provide for both the transaction verification and client protection as well but they fail to mention how form grabbing will be tackled since the main information of the transaction will be exposed for the attacker to grab.

In the second review which is **WSAD (2010)**, which stands for Web Contents Protection, Secure Execution and Authorized Distribution provides protection for the web contents by encrypting it with a secured web server and as the contents passes through the internet to the client, a secured proxy on the client side will decrypt it which provides better protection of web contents stored in the server and at the same time provides verification for the transaction. However, since the solution only concentrates on the protection of web contents on the server side and lacking protection against software attack on the client side, it shows that it lacks client protection. The work also didn't address any solution on form grabbing issues which is a major parameter.

The third review, Mobile User Authentication System for e-Commerce Applications which we abbreviated as **MUAS (2012)** also provided web content protection in their work by using Quick Response (QR) code technology to authenticate users through challenge/response protocol and also provide transaction verification by making use of mobile device as an out of band communication channel. But it didn't offer client protection since the details will still be exposed which might be "grabbed" due to form grabbing.

The fourth review **Secure Login Using Encrypted One Time Password (Otp) and Mobile Based Login Methodology**, which is abbreviated as **SLOTP (2013)** also provided for web

content protection by encrypting the messages being sent from the server to client and vice versa and provided for client protection by the usage of encrypted OTP sent through mobile device which the user will use to log in which provides stronger authentication but the work didn't address transaction verification and form grabbing since the information being exchanged even though being encrypted is still exposed and risks being grabbed.

Compared to the fifth work **EWSA (2015)** which is **Enhanced Web Security Application for Online Financial Transactions**, this takes care of all the loopholes found in the other four literature reviews. The work provides web content protection with the use of JSON Web Token (**JWT)** used in transferring information which can be verified and trusted with digital signature. It also deals with form grabbing issues by providing **Anti-form grabbing technique** whereby the sensitive information such as account number and amount are being encoded to pseudo characters which the attacker has no knowledge of. Transaction verification is taken care of by rerouting the transaction verification through the **Email** instead of the same session. **One Time Password (OTP)** which can be called **Token** was used as a password that is valid for only one login session or transaction within a limited time which gives client protection by providing stronger authentication at the client side.

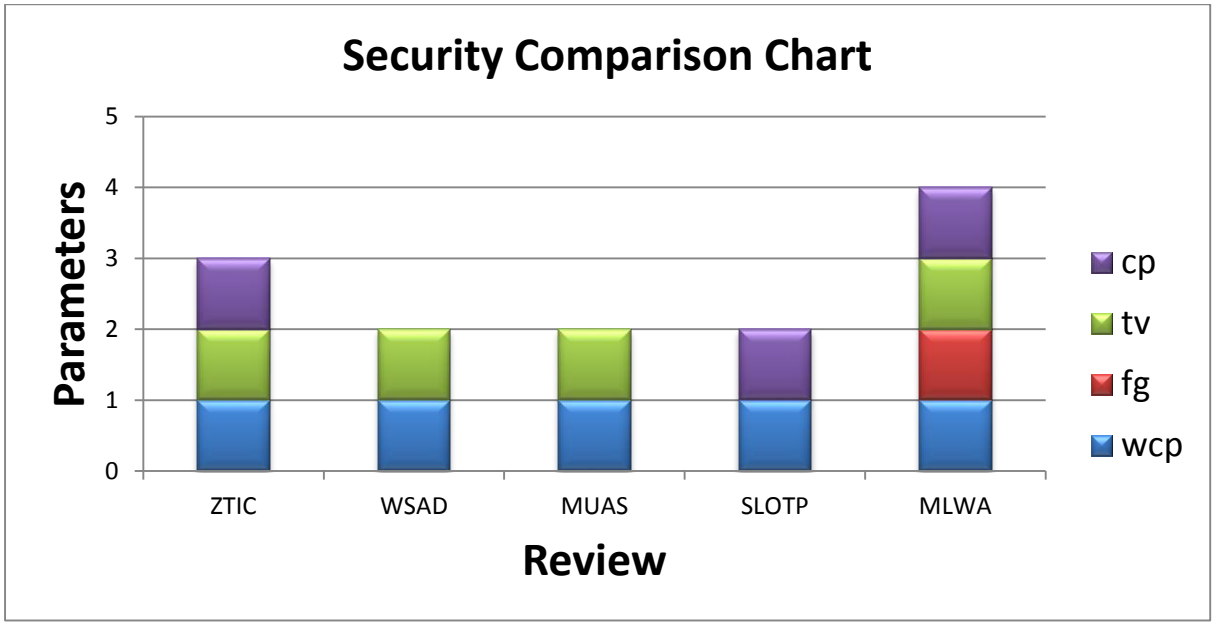


Figure 4.11: Security Analysis Chart

Figure 4.11 summarizes the table in Table 4.4 in order to show the strength in the security analysis of each work compared to **EWSA**.

cp : client protection

tv : transaction verification

fg : form grabbing

wcp : web content protection

As seen above, the components of security done on **EWSA** is higher than the other previous reviews which show that a stronger security has been designed to tackle MitB.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

The main aim of this dissertation is to develop mechanisms for preventing MitB attacks or make these attacks difficult in online financial transactions. The review of the literature pertaining to the work was discussed and also how MitB attack operates. The work also gives insight on the design of the Enhanced Web Security Application Security, the implementation and functionalities as well. The design of the system architecture was thoroughly discussed which demonstrates the working technique of the Anti-Form grabbing method of securing the web, how the JWT is securing the information exchange, how the OTP authenticates the client side for adequate security and also show how the email verification service was employed. The algorithm of both the Anti-Form grabbing technique and Token generation were developed. The information exchange between the client and the server was protected by introducing Anti-form grabbing in which the user inputs are immediately captured and encoded by a dynamically generated JavaScript from the server side making what is displayed on the screen different from what is being typed so even if the attacker "grabs" the form input of the user, the user has only "grabbed" combination of meaningless symbols which the attacker cannot decode. During the information exchange, a token is sent to the user's email and at this time, the user logs in to his email to get the token which will be entered correctly. The email service serves as a way of identity verification by rerouting the verification from the server instead of the same session which the attacker might see and modify to his own. The JSON Web Token will take care of the web content as well which is the "Account Number" and the "Amount" in the entry fields. The information is sent through JWT as part of the headers to the server. The performance of the work was assessed and showed comparative security analysis against previous literature review to show clearly that

the enhanced security approach used in this work is stronger with the help of a table and a chart.

5.2 Conclusion

With these security challenges discussed earlier, this dissertation proffers solution to two key areas that MitB penetrates which is the login in aspect and the verification aspect. In order to mitigate these security issues, this research proffers a solution to the problem by introducing an **anti-form grabbing technique** which disallows the attacker from "grabbing" sensitive information, **JWT** is used in transferring information which can be verified and trusted with digital signature and **Token** which was used as an **OTP** for verification through **email** which are part of the aim and objectives of this work and we have successfully been able to achieve that.

5.3 Recommendation

In future research, some other threats of MitB like keylogger can be looked into since the proposed work encodes what is being seen but doesn't provide security against what is being logged by the keylogger through the keystrokes that the user typed. By so doing, adding this security will make the work all-encompassing to deal with MitB.

REFERENCES

Abbasi, A.G., Muftic, S., and Hotamov, I. (2010). Web Contents Protection, Secure Execution and Authorized Distribution, Computing in the Global Information Technology, *Fifth International Multi-conference on Computing in the Global Information Technology, International Multi-Conference* on, pp. 157-162.

Akinwale, T. A., Adekoya, F. A., and Ooju, E. O. (2011). Multi-Level Cryptographic Functions for the Functionalities of Open Database System, Department of Computer Science, University of Agriculture, Abeokuta, Nigeria, pp. 730-735.

Association of German Banks. (2007). *Online banking security*. Berlin: Bundesverband deutsher Banken.

Batchelor, B., *The History of E-Banking*. Retrieved August 11 2014 from http://www.ehow.com/about_5109945_history-ebanking.html

Boswell, W. (2014)., *The History of the Web*. Retrieved August 10 2014 from http://websearch.about.com/od/searchingtheweb/a/webhistory.htm

Canali, D., and Balzarotti, D. (2013). Behind the Scenes of Online Attacks: an Analysis of Exploitation Behaviors on the Web. *20th Annual Network and Distributed System Security Symposium*, San Diego, CA, United States.

Dougan, T., Curran, K. (2012). Man in the Browser Attacks, *International Journal of Ambient Computing and Intelligence*, **4**(1), 29-39.

Fazli, B., Kamarularifin, A., and Jamalul-lail, A. (2012). Mitigating Man-In-The-Browser Attacks with Hardware-based Authentication Scheme *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*. **1**(3): 204-210.

Jason, W., Damien, H., and Justin, P. (2008). Enhanced Security for Preventing Man-in-the-Middle Attacks in Authentication, Data Entry and Transaction Verification. Deakin University: *Australian Information Security Management Conference*, pp 198-212

Jjchai. (2010). *Online banking*. Retrieved August 11 2014 from http://www.slideshare.net/jjchai/online-banking

Nilsson, D. (2012). *Security in Behaviour Driven Authentication for Web Applications*, Master's thesis, Department of Computer Science, Electrical and Space Engineering, Lulea University of Technology.

Rania, A. M., Imed, R., Buchanan, B, and Etimad, Y. F. (2012). *Mobile User Authentication System for e-Commerce Applications*. Department of Computer Science College of Computing and Information Technology, King Abdulaziz University Jeddah, Kingdom of Saudi Arabia, pp. 220-228 .

RSA Lab. (2011). MAKING SENSE OF MAN-IN-THE-BROWSER ATTACKS - Threat Analysis and Mitigation for Financial Institutions. Retrieved April 21 2013 from http://viewer.media.bitpipe.com/1039183786_34/1295277188_16/MITB_WP_0510-RSA.pdf.

SafeNet. (2010). Man-in-the-Browser - Understanding Man-in-the-Browser Attacks and Addressing the Problem. Retrieved November 7 2015 from http://www.safenet-inc.com/.../man-in-the-browser-security-guide/

Scholasticus, K.(2009)., History of Internet Banking. Retrieved August 11 2014 from http://www.buzzle.com/articles/history-of-internet-banking.html

Sidheeq, M., Dehghantanha, A., and Kananparan, G. (2010). *Utilizing trusted platform module to mitigate botnet attacks*, Computer Applications and Industrial Electronics, International Conference on Computing in the Global Information Technology, vol., no., pp. 245-249.

SOLIDPASS SECURITY REBORN, Online Banking Security. Retrieved March 15 2014 from http://www.solidpass.com/solutions/online-banking-security.html

Weigold, W., Kramp, T., Hermann, R., Horing, F., Buhler, P., and Baentsch, M. (2008). The Zurich Trusted Information Channel: An efficient defense against Man-in-the-middle and malicious software attacks TRUST'2008. LNCS, vol. 4968, pp. 75-91.