STRENGTHENING WEB APPLICATION SECURITY THROUGH TECHNICAL MEASURES

BY

MIKE DAKA

A dissertation submitted to the University of Zambia in fulfilment of the requirements for the degree of Master of Engineering in ICT Security

THE UNIVERSITY OF ZAMBIA

LUSAKA

2023

COPYRIGHT

DECLARATION

I declare that this thesis is solely the result of my work. In cases where collaboration with other individuals has occurred, or material generated by other researchers is included, proper references are provided, and all parties involved are explicitly stated. Furthermore, I confirm that this thesis has not been submitted to any other university or institution for any other degree or examination.

Signature: …….. Date: …….

APPROVAL

This dissertation of Mike Daka has been approved as fulfilling the requirements or partial fulfilment of the requirements for the award of a master's degree in Master of ICT Security by the University of Zambia.

Examiner 1	Signature	Date
……	……..	……
Examiner 2	Signature	Date
……	……..	……
Examiner 3	Signature	Date
……	……..	……
Supervisor	Signature	Date
……	……..	……
Chairperson/ Board of Examiners	Signature	Date
……	……..	……

DEDICATION

My family

ACKNOWLEDGEMENT

I am extremely thankful to my supervisor, Dr Dani E. Banda, for his invaluable direction and assistance in writing this dissertation. I am also appreciative of my peers for their significant perspectives. Moreover, I am profoundly grateful to my family, friends, and colleagues for their support and encouragement. Lastly, I would like to recognise the input of all those who helped me in any capacity to complete this task.

ABSTRACT

This research aimed to investigate web application vulnerabilities and develop effective countermeasures against various malware attacks. The research objectives were to identify and assess prevalent web application malware, evaluate current malware detection approaches, and develop secure web applications with a clear understanding of vulnerabilities and associated countermeasures. A detailed methodology was employed, including a mixed-mode approach combining theoretical analysis and practical application.

The research findings highlighted the common types of web application attacks, including SQL injection, directory traversal, and packet sniffing, and underscored the significance of technical measures such as firewalls, intrusion detection systems, and secure coding practices in enhancing web application security. The investigation revealed key vulnerabilities in web applications and demonstrated how these could be exploited by hackers, leading to potential compromise of confidentiality, integrity, and availability.

The study emphasised the effectiveness of input validation, client-side and server-side validation, and input sanitisation as defences against SQL injection attacks. It also stressed the importance of securing the web server with HTTPS encryption to counter packet sniffing vulnerabilities. The research underlined the importance of implementing technical security measures and maintaining vigilance in identifying and addressing potential security risks.

The research contributes to existing efforts in web application security and provides a foundation for future research and development in this area. The findings also highlight the need for continuous monitoring and proactive management of web application security. Future work should explore emerging technologies and tools for enhancing web application security and developing best practices for secure web application development.

Keywords: *Web Security, Malware Attacks, Detection Techniques, Technical Measures, Risk Management.*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

CHAPTER 1: INTRODUCTION

1.1 Overview

With the rapid growth of web applications, the risk of cyber-attacks has increased significantly. Cybercriminals are constantly finding new ways to exploit vulnerabilities in web applications, causing harm to businesses and individuals alike. This research explores various technical measures that can be employed to strengthen web application security. By identifying and analysing these measures, the researcher hopes to provide valuable insights into enhancing the security of web applications and mitigating the risks associated with cyber threats.

1.2 Background to the Research

Web applications have become essential to modern society and are used for various activities, such as online shopping, social media, and banking [1]. The growth of web applications has brought many benefits, but it has also introduced new risks in the form of cybersecurity threats [2]. Cybersecurity threats can significantly impact businesses and individuals, resulting in data breaches, financial losses, and reputational damage [3]. Recently, the number of cyber-attacks targeting web applications has increased, highlighting the need for effective security measures.

Web application security protects web applications from cyber-attacks by identifying and mitigating vulnerabilities. A web application is any software program that runs on a web server and is accessed using a web browser. The security of web applications is essential to prevent unauthorised access, data breaches, and other cybersecurity threats [4], [5]. The security of web applications can be enhanced through technical measures, including secure coding practices, firewalls, intrusion detection and prevention systems, encryption, and authentication.

Secure coding practices involve coding web applications with security in mind from the outset; this involves using secure coding languages and frameworks, implementing input validation, and avoiding common coding mistakes that can lead to vulnerabilities. Web application firewalls (WAFs) are designed to protect web applications by monitoring and filtering incoming traffic [4]. WAFs can detect and block attacks such as SQL injection, Cross-site Scripting (XSS), and cross-site request forgery (CSRF). In addition, intrusion

detection and prevention systems (IDPS) can help identify and prevent attacks by monitoring network traffic and system activity [6].

Encryption is another essential technical measure for web application security. It involves encoding sensitive data to prevent unauthorised access or data breaches. Encryption can be applied to transit data and rest [7]. For example, SSL/TLS can encrypt data in transit, while database encryption encrypts data at rest.

Authentication is also a critical technical measure for web application security. It involves verifying the identity of users accessing web applications to prevent unauthorised access. Authentication can be achieved through various methods such as passwords, biometrics, and multi-factor authentication [7].

While technical measures can effectively strengthen web application security, they are not foolproof. Attackers can still find ways to bypass these measures, mainly if they are not implemented correctly or kept up to date. Therefore, it is essential to ensure that technical measures are part of a comprehensive web application security strategy, including regular security testing, patch management, and user education [8].

Web application security is the dynamic and complex nature of web applications. Web applications constantly evolve, and new vulnerabilities can be introduced through software updates or changes to the underlying infrastructure; this means that web application security must be an ongoing process that includes regular security testing and vulnerability assessments [9].

Web application security is the increasing sophistication of cyber-attacks. Cybercriminals use advanced social engineering and artificial intelligence techniques to target web applications. This means that web application security measures must be updated to keep up with the latest threats.

Web application security is a multifaceted issue that requires collaboration and cooperation across different areas of an organisation. Developers, security teams, and management must work together to ensure that web applications are developed and maintained with security in mind. Training and education are also essential to ensure employees know the risks associated with web applications and understand how to use them securely.

Another challenge is the difficulty of detecting and responding to web application attacks. Web application attacks can be challenging to detect because they often blend in

with legitimate traffic [10]. Additionally, once an attack has been detected, responding to it can be challenging because it may involve shutting down critical systems or services. To address this challenge, organisations can use tools such as intrusion detection and prevention systems and Security Incident and Event Management (SIEM) solutions to monitor and respond to web application attacks.

Web application security is also complicated because web applications are often developed using third-party components and libraries. These components may have vulnerabilities that attackers can exploit, and keeping track of updates and patches for these components can be challenging. Organisations can use software composition analysis tools to address this challenge to identify and manage third-party components and libraries used in web applications.

There has been a growing trend towards DevOps and agile development methodologies in recent years, emphasising speed and flexibility in software development [9]. While these methodologies can improve the speed and efficiency of web application development, they can also introduce new security risks if security is not considered from the outset. To address this challenge, organisations can incorporate security into their DevOps and agile development processes by using tools such as security automation and integrating security testing into the development lifecycle.

Web application security is a complex and evolving issue that requires ongoing attention and investment. Technical measures such as secure coding practices, web application firewalls, encryption, and authentication are essential components of web application security. However, these measures must be part of a comprehensive web application security strategy, including regular security testing, patch management, user education, and compliance with legal and regulatory requirements. Furthermore, with the increasing sophistication of cyber-attacks and the dynamic nature of web applications, organisations must stay updated with the latest security measures and adapt their security strategies accordingly.

1.3 Statement of Problem

Web application security is a critical concern with the escalating reliance on web applications. Existing research has extensively explored web application vulnerabilities, concentrating on malware attacks such as SQL injection, directory traversal, and packet sniffing [1]. However, there is a dearth of comprehensive understanding regarding the

effectiveness of different technical measures in mitigating these attacks [2]. Moreover, existing literature has a scant focus on specific contexts, such as Zambia, leaving a significant gap in context-specific knowledge [3].

Several studies have made substantial contributions to understanding web application vulnerabilities. For instance, Katanga M and Mbwikalambo R investigated the effectiveness of technical countermeasures in bolstering web application security [4]. Their research focused on firewalls and intrusion detection systems, leaving out other technical measures. Also, their study lacked a geographical focus, particularly on Zambia, creating a gap in context-specific research [5].

In another comprehensive review of web application security, Alghathbar and Alruban identified key security threats and vulnerabilities associated with web applications [6]. They evaluated the effectiveness of various security measures and techniques for mitigating these threats. However, their study did not delve into the effectiveness of specific technical measures in enhancing web application security, indicating a gap in the literature [7].

The research work by Akamai Technologies Inc., titled "Web Application Security: How to Avoid Authentication Attacks," provides useful insights on averting authentication attacks on web applications [8]. However, their paper does not provide empirical evidence on the effectiveness of the measures suggested, revealing another gap in the literature [9].

Given the gaps identified in the existing literature, this research aims to evaluate various technical measures for enhancing web application security, focusing specifically on the context of Zambia.

1.4 Research Aim

This research aims to identify and evaluate various technical measures that can be implemented to enhance web application security and mitigate the risk of cyber-attacks and data breaches.

1.5 Research Objectives

1. To identify and evaluate the various types of malware that commonly attack web applications.

2. To better analyse and assess current web application malware detection techniques to understand their strengths and weaknesses.
3. To design and develop web applications that demonstrate vulnerabilities and countermeasures.

1.6 Research Questions

1. What are the different types of malware that commonly attack web applications, and how do they operate?
2. What are the strengths and weaknesses of current web application malware detection techniques, and how effective are they in detecting and mitigating malware attacks?
3. How can vulnerabilities in web applications be identified, and what technical measures can be implemented to strengthen web application security and prevent malware attacks?

1.7 Significance of the Research

Adequate security measures have become more pressing with the growing reliance on web applications for online banking, e-commerce, and social media activities. This research provides valuable insights into technical measures that can be implemented to enhance web application security, particularly concerning combating malware attacks. The findings of this research are helpful for web application developers, cybersecurity professionals, and organisations that rely on web applications, helping them better understand the nature of the security threat and develop effective measures to protect their systems and data.

1.8 Scope of the Research

This research explores technical measures that can be implemented to strengthen web application security, with a particular emphasis on combating malware attacks. The research involves a review of existing literature on web application security, analysing current malware detection techniques, and developing a web application to demonstrate vulnerabilities and countermeasures. The research is limited to technical measures and does not cover other aspects of web application security, such as policy and training.

1.9 Ethical Considerations

The ethical considerations for this research are significant and were carefully considered throughout the study. One primary ethical concern in this research is handling sensitive data and information, especially during the testing phase of web application attacks. All tests were performed in a controlled environment using designed test sites to mitigate the associated risk. No real-world web applications were tested, and no actual user data was used in this research. Furthermore, any data generated during the research were anonymized and stored securely to ensure privacy and confidentiality.

Another key ethical consideration is the potential to misuse the information and methods presented in this research. While this research aims to improve web application security, the techniques and vulnerabilities discussed could potentially be exploited for malicious purposes by others. To address this risk, the research was conducted strictly on improving security measures and mitigating attacks, not enabling them. Any potentially harmful information was presented responsibly and ethically, emphasizing its use for enhancing security, not undermining it. The research was designed to contribute to the broader efforts in improving web application security, adhering to the ethical guidelines of responsible and beneficial research.

1.9 Summary of Chapter 1

Chapter 1 introduced the research objectives, questions, scope, and justification. The significance of technical measures for enhancing web application security and combating malware attacks was highlighted. In Chapter 2, the literature review provides a comprehensive overview of the existing literature on web application security and malware attacks. The chapter covers the types of malware that commonly attack web applications, current security measures and techniques, and the strengths and weaknesses of existing malware detection techniques. The literature review provides a theoretical foundation for the research, identifies gaps in current knowledge, and informs the research's methodology.

CHAPTER 2: LITERATURE REVIEW

2.1 Overview

Chapter 2 is dedicated to the literature review, which presents a detailed analysis of the current knowledge on web application security and malware attacks. This chapter provides a comprehensive overview of the types of malware that commonly target web applications and an assessment of the current security measures and techniques used to mitigate these attacks. The strengths and weaknesses of the existing malware detection techniques are also discussed, providing insights into the gaps in the current knowledge and informing the development of the research methodology. Overall, this chapter forms the theoretical foundation for the research, helping to identify the key issues and challenges facing web application security.

2.2 Malware in Web Applications

Web applications have become integral to our daily lives, from online shopping to banking and social networking. However, with this increased usage comes an increased risk of cyber-attacks, and one of the most significant threats is malware. Malware can take many forms, each with unique characteristics and attack methods. This research provides an overview of the different types of malware within the scope of the research that commonly targets web applications, including SQL injection, Dirbuster, password sniffing, and packet sniffing.

2.2.1 SQL Injection

Structured Query Language (SQL) injection is one of the most common web application attacks, targeting the application's database [11]. In a SQL injection attack, an attacker inserts malicious SQL statements into the application's input field to gain unauthorised access to the database, as shown in Figure 0.1. The attacker can then use this access to view, modify, or delete sensitive information stored in the database.

The behaviour of SQL injection malware in web applications depends on the type of SQL injection used. Different types of SQL injection exist, including in-band SQLi, blind SQLi, and out-of-band SQLi [12].

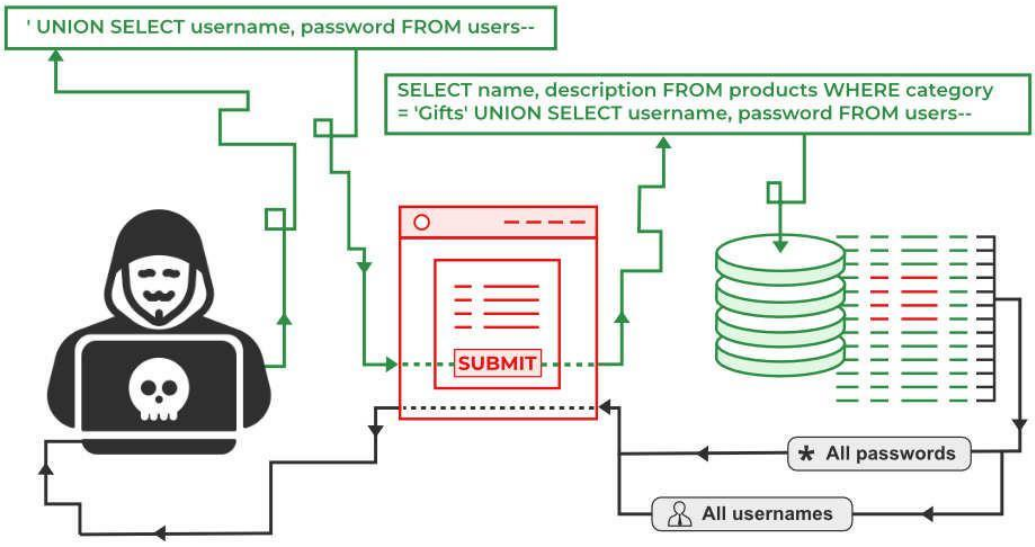


Figure 0.1: SQL Injection Attack Implementation [11]

In-band SQLi is the most common type of SQL injection [13]. It involves the attacker sending an SQL query to the database and receiving the results directly. The attacker can then use this information to exploit vulnerabilities in the web application. For example, the attacker can use SQL injection to steal user credentials by injecting SQL code into the login form.

Blind SQLi, on the other hand, is a type of SQL injection where the attacker does not receive any results directly from the database [14]. Instead, the attacker uses the web application to send requests to the database and receives the results indirectly. This makes it more difficult to detect and exploit vulnerabilities in the web application.

Out-of-band SQLi is a less common type of SQL injection that involves the attacker using the web application to send requests to an external server controlled by the attacker [14]. The external server then sends requests to the database, allowing the attacker to bypass security measures in the web application and extract sensitive information from the database.

Characteristics of SQL injection malware in web applications include the ability to manipulate SQL queries, bypass authentication mechanisms, and extract sensitive information from databases [12]. The malware can also be used to modify or delete database data, execute arbitrary code on the server, and take control of the entire web application.

Prevention and mitigation of SQL injection involve implementing security measures such as input validation, parameterised queries, and stored procedures. Other measures

include using web application firewalls, limiting user privileges, and conducting regular security audits to detect and address vulnerabilities in the web application [14], [15].

2.2.2 Directory Traversal attack

Directory Traversal attack using DirBuster is another type of web application attack that targets the application's file and directory structure [16]. In a Directory Traversal attack, an attacker uses a tool that attempts to brute-force directory and file names on the application's server, as shown in Figure 0.2. The goal is to discover hidden files or directories containing sensitive information, such as configuration files or user credentials [17].

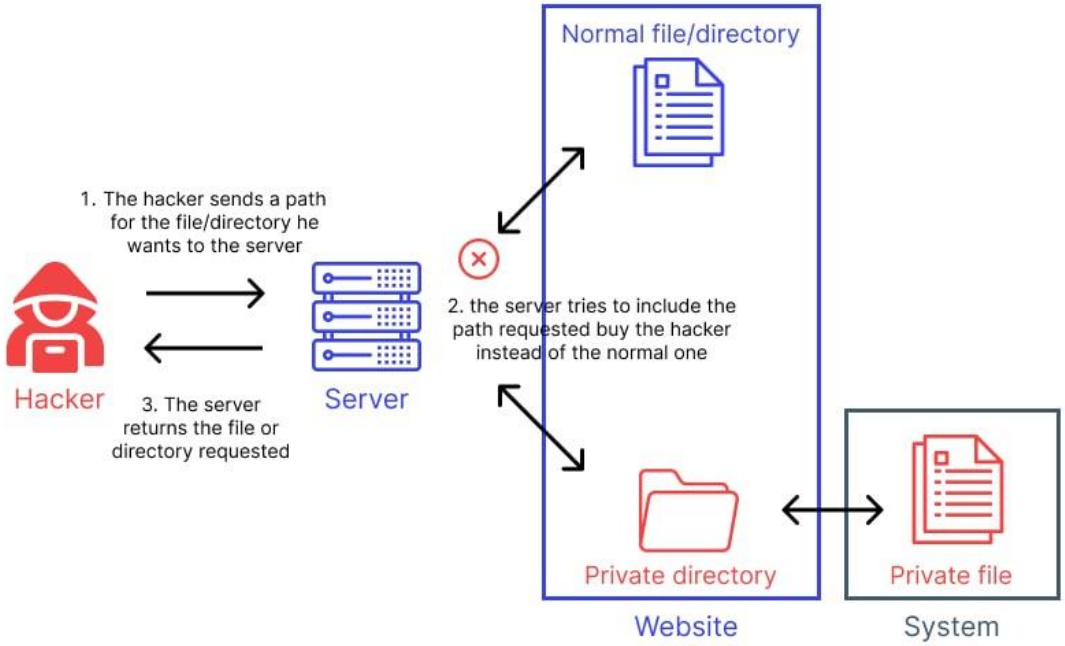


Figure 0.2: Directory Traversal Attack Implementation [18]

The behaviour of DirBuster in web applications is to perform a recursive scan of directories and files, looking for web pages that may be hidden or not linked in the web application [19]. Once DirBuster identifies a directory or file, it will request that page and analyse the response from the server. For example, if the server responds with a Hypertext Transfer Protocol (HTTP) 200 status code, DirBuster will mark that directory or file as valid [20].

The characteristics of Directory Traversal attack malware in web applications include the ability to perform brute-force attacks on web directories and files, identify hidden or non-linked pages in web applications, and extract sensitive information [18]. In addition,

DirBuster can also be used to map the web application's structure, identify potential vulnerabilities in the web application, and identify potential entry points for other types of malware.

To prevent and mitigate the effects of Directory Traversal attack malware in web applications, security measures such as limiting directory listing, implementing access controls, and restricting access to sensitive files and directories should be implemented [16]–[18]. It is also important to keep web applications updated and to conduct regular security audits to detect and address vulnerabilities in the web application. Using tools like web application firewalls can also help to protect against brute-force attacks and other types of web application attacks.

2.2.3 Password Sniffing

Password sniffing is a type of web application attack that targets user credentials. In a password sniffing attack, an attacker intercepts network traffic between the user and the application and captures the user's login credentials [21]. The attacker can then use these credentials to gain unauthorised access to the application or other systems the user can access, as shown in Figure 0.3.

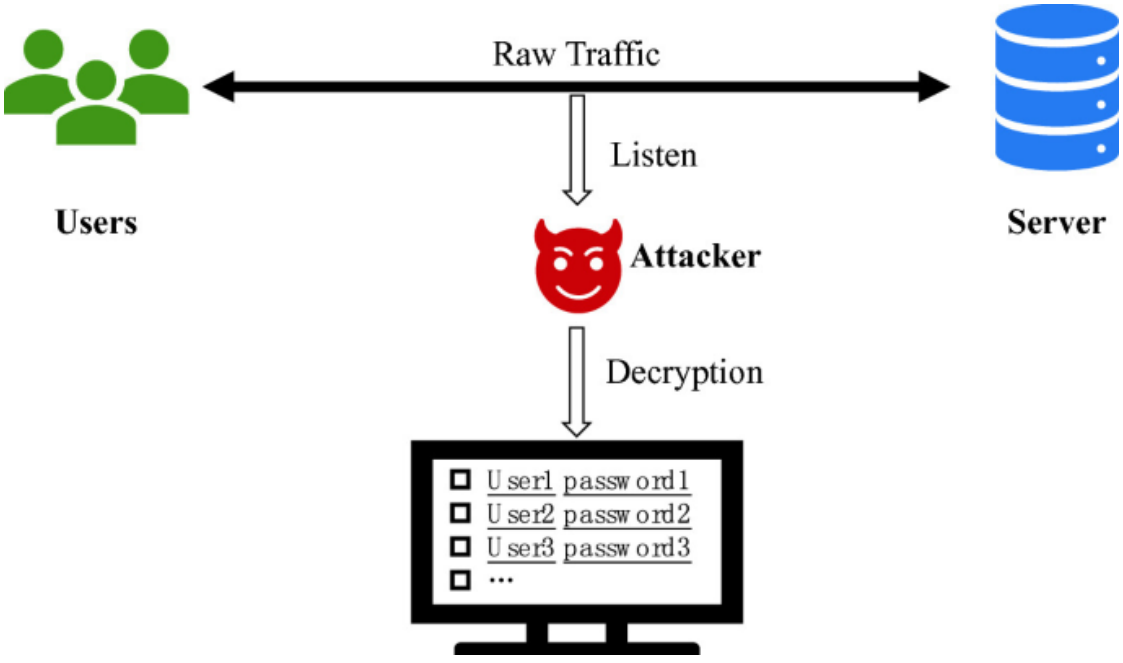


Figure 0.3: Password Sniffing Attack Implementation [22]

The behaviour of password sniffing malware in web applications is to capture login credentials as they are transmitted over the network [22], [23]. The malware will intercept network traffic and search for login credentials in clear text. Once captured, the malware

can transmit the credentials to the attacker's command and control server, where the attacker can use them for malicious purposes.

The characteristics of password-sniffing malware in web applications include the ability to intercept network traffic and capture login credentials in clear text, operate without the user's knowledge, and compromise sensitive information [24]. In addition, password-sniffing malware can collect additional information like credit card numbers or other sensitive data transmitted over the network.

To prevent and mitigate the effects of password sniffing malware in web applications, security measures such as using encrypted network protocols, implementing two-factor authentication, and regularly changing passwords should be implemented [24], [25]. It is also essential to ensure that all devices and software are updated with the latest security patches and updates. Network monitoring tools can also detect and alert abnormal network traffic, which can help identify the presence of password-sniffing malware.[24]

2.2.4 Packet Sniffing

Packet sniffing is another type of web application attack that targets network traffic. In a packet sniffing attack, an attacker captures network traffic between the user and the application and analyses the packets for sensitive information, such as user credentials or other sensitive data [26]. The attacker can then use this information to gain unauthorised access to the application or other systems the user has access to.

In web applications, packet sniffing malware behaves by intercepting network traffic and capturing packets containing sensitive information such as login credentials, cookies, and other sensitive data [27]. The malware can then transmit the captured packets to a command and control server where the attacker can analyse and extract the sensitive information.

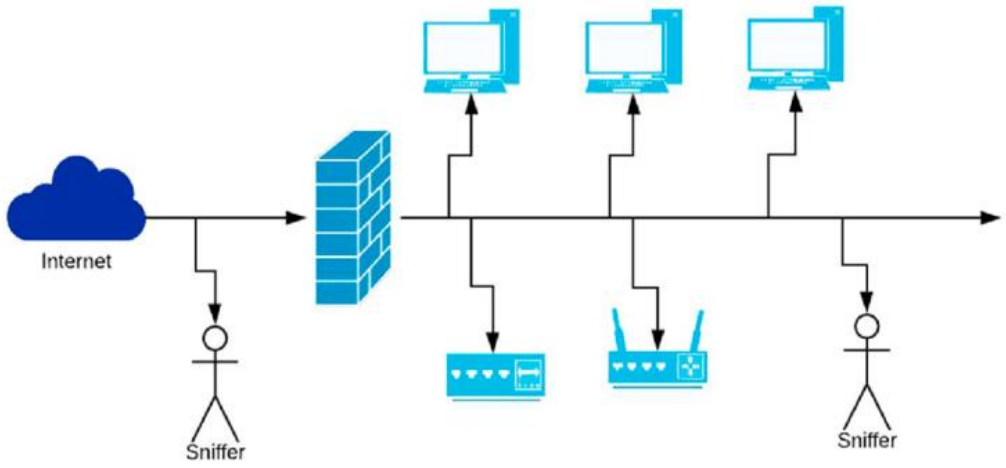


Figure 0.4: Packet Sniffing Attack Implementation [28]

Packet sniffing malware can be difficult to detect as it does not modify the traffic it captures, making it challenging to identify. However, the malware can be detected using network monitoring tools that can identify abnormal traffic patterns and alert security personnel to the presence of the malware [28].

Characteristics of packet sniffing malware in web applications include capturing network traffic in real-time, extracting sensitive data from captured packets, and operating without the user's knowledge [29].

To prevent and mitigate the effects of packet sniffing malware in web applications, security measures such as using encrypted protocols, implementing two-factor authentication, and regularly changing passwords can be used [29]. Additionally, network monitoring tools can detect and alert abnormal network traffic, which can help identify the presence of packet-sniffing malware [30].

Packet-sniffing malware is a serious threat to the security of web applications as it can capture and extract sensitive information transmitted over the network. Therefore, organisations must implement adequate security measures and regularly monitor network traffic to prevent and detect packet-sniffing malware.

Web application security is an ever-evolving field; new types of malware will continue to emerge as technology advances. Therefore, staying updated with the latest security trends and techniques to protect sensitive information stored in web applications is crucial. This research provides an overview of the types of malware that commonly target

web applications, including SQL injection, Dirbuster, password sniffing, and packet sniffing. By understanding these malware types' characteristics and methods of attack, web application developers can implement adequate security measures to prevent them and protect their users' sensitive information.

2.3 Security Measures for Mitigating Malware Attacks

Web applications have become integral to modern-day business and social interactions [1]. They enable businesses and individuals to reach a wider audience, share information, and provide services. However, the increasing use of web applications has also led to a rise in cyber-attacks, with malware attacks being one of the most common. Malware attacks can result in data theft, service disruptions, financial losses, and reputational damage [4]. Therefore, it is essential to have adequate security measures and techniques in place to mitigate these attacks. This research paper provides an overview of the current security measures and techniques for mitigating malware attacks. This research also evaluates the effectiveness of each security measure and technique and discusses its strengths and weaknesses.

2.3.1 Current Security Measures and Techniques

Several security measures and techniques protect web applications from malware attacks. These include but are not limited to firewalls, intrusion detection systems, anti-malware software, secure coding practices, and user training [7].

Firewalls are used to prevent unauthorised access to a network or system. They examine incoming and outgoing traffic and block any traffic that does not meet the defined security policies. Firewalls can be deployed as software or hardware devices and configured to allow or deny traffic based on the source IP address, destination IP address, port number, and other criteria [4].

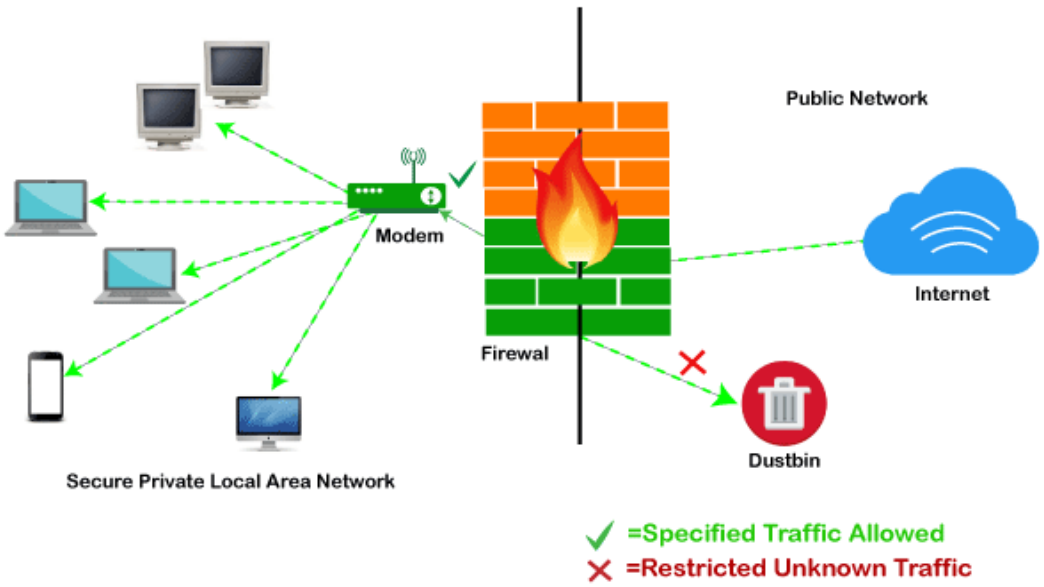


Figure 0.5: Firewall Implementation [31]

Intrusion detection systems (IDS) detect and respond to unauthorised access attempts or other malicious activities. IDS can be either network-based or host-based. Network-based IDS monitor network traffic for suspicious activity, while host-based IDS monitor system logs and events for signs of intrusion.

Anti-malware software is used to detect, remove, and prevent malware infections. Anti-malware software can be either signature-based or behaviour-based. Signature-based anti-malware software uses a database of known malware signatures to identify and remove malware [32]. Behaviour-based anti-malware software monitors software programs' behaviour and blocks any malicious behaviour.

Secure coding practices involve following coding guidelines and standards that ensure the code is secure and free from vulnerabilities. Secure coding practices include, but are not limited to, input validation, output encoding, and proper error handling [33].

User training is essential for creating a security-aware culture. Users must be aware of the risks associated with web applications and how to identify and respond to potential threats. User training can include phishing attacks, password management, and safe browsing habits [33].

2.3.2 Evaluation of the Effectiveness of Each Security Measure and Technique

While each security measure and technique has its strengths and weaknesses, their effectiveness depends on various factors, such as the type of malware being targeted, the

complexity of the web application, and the resources available for implementation and maintenance.

2.3.2.1 Firewalls

Firewalls are one of the most used security measures for protecting web applications. They work by blocking unauthorised access to the application, preventing malware from gaining access to the system. However, firewalls are not foolproof, and attackers can still find ways to bypass them [34].

Due to increasing cyber threats and attacks, web application security is crucial in today's digital world. Firewalls are among the most used security measures for protecting web applications. A firewall acts as a barrier between a web application and the Internet, monitoring all incoming and outgoing traffic and blocking unauthorised access to the application [35]. Firewalls can be implemented in various ways, such as hardware-based or software-based, and can be configured to allow or deny specific types of traffic.

Despite the widespread use of firewalls, they are not foolproof, and attackers can still find ways to bypass them [36]. For example, if a firewall is not configured correctly, it can allow traffic intended to be blocked. Additionally, firewalls cannot detect and prevent all types of attacks, such as those that exploit vulnerabilities in the application itself [37]. In such cases, additional security measures are required to complement the firewall and provide a comprehensive security solution.

One such security measure is Intrusion Detection and Prevention Systems (IDPS). IDPS is designed to detect and prevent attacks targeting web applications. These systems use various methods to detect and prevent attacks, such as signature-based detection, which involves comparing network traffic to known attack signatures, and anomaly-based detection, which involves identifying deviations from normal network behaviour [38]. In addition, IDPS can also prevent attacks by actively blocking malicious traffic.

Another security measure is safe coding practices. Secure coding involves following established coding practices that minimise the risk of vulnerabilities in the code that attackers can exploit, including input validation, error handling, and access control. By following secure coding practices, developers can significantly reduce the risk of vulnerabilities in the application code, making it more difficult for attackers to exploit them [35].

Web application firewalls (WAFs) are another popular security measure that can complement firewalls [39], [40]. WAFs are explicitly designed to protect web applications and can detect and prevent various types of attacks, including SQL injection, cross-site scripting (XSS), and directory traversal attacks. WAFs work by analysing web application traffic and identifying patterns indicative of an attack [41]. They can then block or allow traffic based on predefined rules.

While these security measures can protect web applications, they have strengths and weaknesses. For example, IDPS can effectively detect and prevent attacks, but they can also generate false positives, leading to unnecessary alerts and potential disruption to the application. Secure coding practices can significantly reduce the risk of vulnerabilities in the code, but they require specialised knowledge and skills and may be challenging to implement in complex applications. WAFs can effectively detect and prevent attacks, but they can also be complex to configure and may generate false positives and negatives.

2.3.2.2 Intrusion detection systems (IDS)

Intrusion detection systems (IDS) are another commonly used security measure for detecting and preventing malware attacks [42]. IDS works by monitoring network traffic and looking for patterns that indicate the presence of malware. Once detected, the IDS can either alert the system administrator or take action to prevent the attack. IDS are effective in detecting known malware, but they may not be able to detect new and emerging threats.

Web application security is critical to online security, especially today, where organisations must protect their web-based applications from malicious attacks. One common approach to securing web applications is intrusion detection systems (IDS). Figure 0.6 shows the architecture of the intrusion detection system.

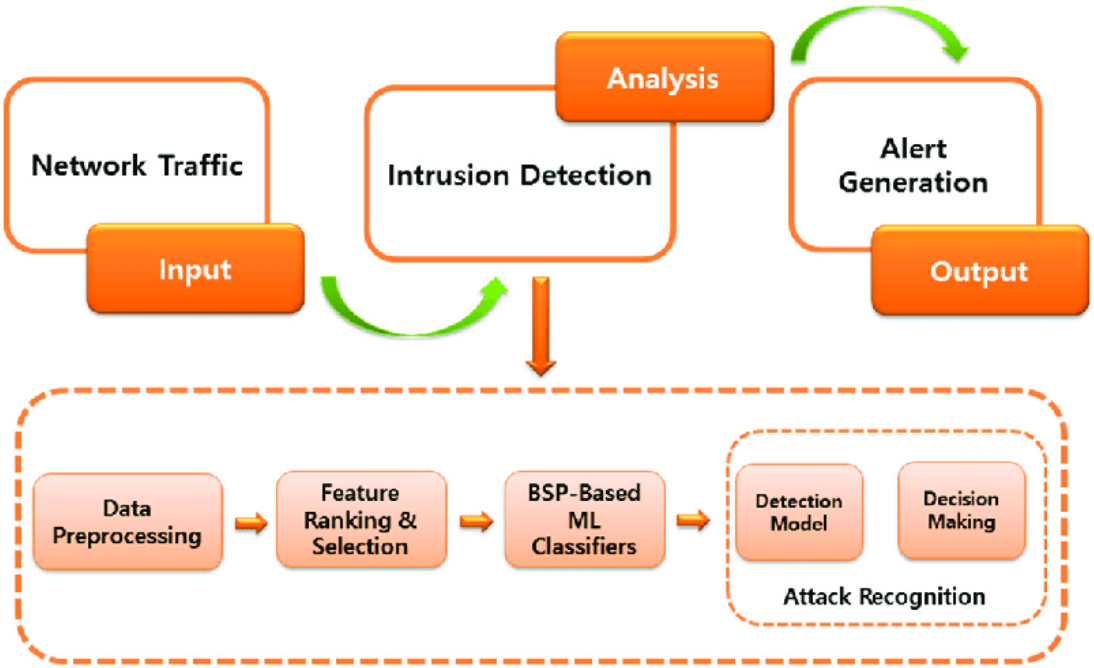


Figure 0.6: Intrusion Detection Systems Architecture [43]

Intrusion detection systems (IDS) are designed to monitor network traffic and look for patterns that indicate the presence of malware [44]. These systems work by analysing network traffic in real-time, searching for known malware signatures or suspicious behaviour that may indicate an attack. IDS can be deployed in different ways, including network-based, host-based, or hybrid.

Network-based IDS, or NIDS, analyse network traffic at specific points, such as routers or switches, and provide alerts when they detect suspicious activity [45], [46]. Host-based IDS, or HIDS, monitor activities on individual computers and servers within the network. Hybrid IDS combine the capabilities of both NIDS and HIDS, providing a more comprehensive approach to intrusion detection.

The primary benefit of IDS is its ability to detect known malware and alert system administrators when an attack is detected [46]. This early warning system allows administrators to prevent data breaches or compromises proactively. IDS can also be configured to take automatic action to block or quarantine malicious traffic, helping to minimise the impact of an attack.

However, IDS does have its limitations. The most significant limitation is their inability to detect new and emerging threats. Malware authors continually create new malware variants that bypass traditional signature-based detection methods used by IDS [47]. In

addition, sophisticated attackers often use evasion techniques to avoid detection by IDS, making it challenging to identify and prevent attacks.

Many modern IDS solutions use machine learning, artificial intelligence, and behaviour analysis to overcome these limitations. These technologies enable IDS to identify anomalous behaviour and detect previously unknown malware based on its behaviour [47]. As a result, these advanced IDS solutions offer higher protection against emerging threats.

In addition to IDS, organisations can implement other security measures. For example, firewalls can control network access. Secure coding practices and regular vulnerability scanning can also help prevent the exploitation of vulnerabilities in web applications.

A comprehensive approach to web application security should combine these different security measures. By implementing multiple layers of security, organisations can significantly reduce the risk of a successful attack on their web-based applications [46].

Intrusion detection systems are an essential component of web application security. While they have limitations, they provide an important defence against known malware and suspicious activity. With the proper configuration, implementation, and additional security measures, IDS can significantly improve an organisation's ability to detect and prevent attacks on their web-based applications.

2.3.2.3 Antivirus

Antivirus software is another widely used security measure for mitigating malware attacks. Antivirus software scans files and programs for known malware signatures [48]. The software quarantines or deletes the infected file if a match is found. However, antivirus software is not effective against new and emerging malware threats.

Web application security is an essential aspect of modern-day computing. With the rise of web applications, there has been a corresponding increase in cyber threats and attacks. Cybercriminals are constantly evolving their techniques to exploit vulnerabilities in web applications to gain unauthorised access to sensitive information. As such, there is a need for robust security measures to ensure the safety and protection of web applications against these attacks.

Antivirus software typically relies on signature-based detection methods, which require the software to know the malware's unique signature [49]. Therefore, the antivirus

software must have a database of known malware signatures to compare with the scanned files and programs. This approach is effective against known malware threats. However, it is ineffective against new and emerging malware threats that do not have a known signature.

Cybercriminals are continually developing new malware threats, and signature-based detection alone cannot keep up with these emerging threats [50]. Antivirus software must employ additional methods to detect and prevent new and emerging malware threats.

One approach that antivirus software can use to detect new and emerging malware threats is behaviour-based detection. Behaviour-based detection analyses the behaviour of programs and files to determine if they exhibit characteristics consistent with malware [51]. For instance, behaviour-based detection can analyse the program's network activity, system calls, and memory usage to determine if it performs malicious activities.

Another approach that antivirus software can use to detect new and emerging malware threats is machine learning algorithms. Machine learning algorithms can analyse large amounts of data to identify patterns and anomalies that may indicate the presence of malware [48]. By analysing data from various sources, machine learning algorithms can identify malware threats that signature-based detection methods may miss.

Apart from antivirus software, there are several other security measures that web applications can employ to enhance their security. One of the most effective security measures is secure coding practices. Secure coding practices involve writing code with security in mind, and it aims to prevent vulnerabilities that cybercriminals can exploit to gain unauthorised access to sensitive information [1], [31], [52].

2.3.2.4 Secure coding practices

Secure coding practices are also crucial for mitigating malware attacks. By following secure coding practices, developers can ensure their code is free of vulnerabilities attackers can exploit [11]. Examples of secure coding practices include input validation, output encoding, and error handling.

Web application security is of paramount importance in today's digital age. With the increasing dependence on web applications for various business processes, ensuring these applications are secure and free from vulnerabilities is imperative [12], [15].

Malware attacks are one of the most significant threats to web application security, and it is crucial to implement secure coding practices to mitigate these attacks.

Secure coding practices are essential for web application security as they help developers identify and address code vulnerabilities [14]. One such practice is input validation, which involves validating all user input to detect invalid or malicious input. This practice can help prevent SQL injection attacks, where attackers inject malicious SQL queries into an application's input fields to gain access to sensitive data.

Another secure coding practice is output encoding, which involves encoding all output before displaying it to the user [12]. This practice can prevent cross-site scripting (XSS) attacks, where attackers inject malicious scripts into an application's output fields to steal sensitive user data or hijack user sessions.

Error handling is also a critical secure coding practice. Developers should ensure that error messages do not reveal too much information about the application's internal workings, such as file paths or database details [13]–[15]. Attackers can use this information to launch targeted attacks on the application.

In addition to secure coding practices, several other measures can be taken to enhance web application security. One such measure is regular software updates. Developers should always stay updated with the latest security patches released by software vendors and apply them promptly to their applications. This can help protect against known vulnerabilities that attackers may exploit.

Another important aspect of web application security is access control. Developers should implement strict access control mechanisms to ensure that only authorised users can access the application's sensitive features and data [15]. This can help prevent unauthorised access to sensitive data and minimise the impact of security breaches.

Encryption is also a critical security measure for web applications. Developers should encrypt sensitive data in transit and at rest to prevent eavesdropping and data theft. SSL/TLS encryption is commonly used for securing web application traffic, while disk encryption can be used to secure sensitive data stored on servers.

Penetration testing is another essential component of web application security. Developers should conduct regular penetration testing to identify vulnerabilities in their applications that attackers may exploit. Penetration testing involves simulating real-world attacks on the application and identifying weaknesses that must be addressed.

2.3.2.5 Regular software updates

Web application security is a critical aspect of protecting organisations from cyber threats. Web applications are widely used for various business operations and store sensitive data that attackers can target. One of the most significant threats to web application security is malware attacks, which can cause severe damage to organisations. Therefore, it is essential to implement measures to help mitigate these attacks.

Regular software updates and patches are crucial in mitigating malware attacks on web applications. Software vendors release updates and patches to fix security vulnerabilities and address other issues attackers can exploit[4]. Organisations can reduce the risk of malware attacks by keeping their software up to date. Failure to update software exposes web applications to known vulnerabilities, allowing attackers to exploit them easily.

Software vendors release updates and patches regularly to fix known security vulnerabilities [48]. As soon as a vulnerability is discovered, the vendor develops a patch to fix the issue; this means that the faster organisations apply software updates, the faster they can address known security vulnerabilities before attackers can exploit them. Conversely, a lack of regular software updates increases an organisation's attack surface, leaving it vulnerable to malware attacks and other cybersecurity threats.

Additionally, failing to keep software up to date can result in compliance issues, leading to severe legal and reputational consequences. Data protection regulations such as GDPR require organisations to take necessary measures to protect personal data. Failing to update software regularly can lead to data breaches, resulting in significant financial penalties and reputational damage.

Organisations must prioritise software updates and patches to stay ahead of the constantly evolving threat landscape. They must have robust procedures in place to manage software updates effectively. Regular software updates should be scheduled, tested, and deployed promptly. Organisations should also ensure that all employees know the importance of applying software updates immediately when they become available.

In addition to regular software updates, organisations can implement other measures to enhance web application security. One such measure is using antivirus software to detect and remove malware from web applications. Antivirus software can scan the application's files and detect any malicious code that may have been injected into them [53].

Regular software updates and patches are crucial for mitigating malware attacks on web applications. Software vendors release updates and patches regularly to fix known security vulnerabilities, and organisations must apply them promptly to stay ahead of constantly evolving threats [54]. In addition to regular updates and patches, implementing measures such as access control, antivirus software, web application firewalls, encryption, and penetration testing can enhance web application security and protect organisations from cyber threats. Furthermore, by prioritising web application security, organisations can ensure the safety of their sensitive data and maintain their reputation.

2.3.3 Strengths and Weaknesses of Each Security Measure and Technique

Each security measure and technique has its strengths and weaknesses. Firewalls, for example, are effective in blocking unauthorised access to web applications, but they may not be able to detect malware that has already gained access to the system [55]. Intrusion detection systems are effective in detecting known malware, but they may not be able to detect new and emerging threats. Antivirus software is effective in detecting known malware signatures, but it may not be able to detect new and emerging threats [56], [57].

Secure coding practices effectively reduce the risk of vulnerabilities that attackers can exploit, but they may not be able to detect all vulnerabilities. Likewise, regular software updates and patches effectively address known vulnerabilities, but they may not be able to address new and emerging threats.

In addition to these strengths and weaknesses, there are other factors that organisations need to consider when choosing which security measures and techniques to use. These include cost, ease of implementation, and compatibility with existing systems.

Malware attacks continue to be a significant threat to web application security. Organisations must implement adequate security measures and techniques to mitigate these attacks [58]–[60]. This research has provided an overview of the current security measures and techniques for mitigating malware attacks, evaluated their effectiveness, and discussed their strengths and weaknesses. It is recommended that organisations use a combination of security measures and techniques to achieve a more comprehensive approach to web application security.

2.4 Critical Issues in Web Application Security

Web application security has become a critical issue for organisations worldwide due to the increasing number of cyber-attacks targeting web applications. Web applications are the primary target of malware attacks as they are accessible from anywhere online. The importance of web application security cannot be overemphasised, as an attack can lead to the loss of sensitive information, financial losses, and damage to the organisation's reputation. This section aims to identify the key issues and challenges facing web application security and discuss the implications of these challenges for developing the research methodology.

2.4.1 Gaps in Current Knowledge

One of the significant gaps in the current knowledge of web application security is the lack of understanding of the latest threats and attack methods hackers use. Attackers are constantly developing new malware and techniques to exploit vulnerabilities in web applications [61], [62]. Another gap is the lack of awareness among organisations regarding the importance of web application security [63], [64]. As a result, many organisations do not adequately understand the risks associated with web applications, which leads to inadequate security measures.

Another gap in current knowledge of web application security is the lack of understanding of the latest threats and attack methods hackers use. Attackers constantly develop new malware and techniques, making it difficult for security professionals to stay updated with the latest security measures [65]. A lack of understanding of these latest threats can lead to inadequate security measures being implemented, leaving web applications vulnerable to exploitation.

The latest threat trends include cross-site scripting (XSS), SQL injection, DDoS attacks, and file inclusion attacks [66]. These attacks can result in significant damage, including financial loss, loss of confidential data, and reputational damage.

Cross-site scripting (XSS) is an attack where an attacker injects malicious code into a website, causing it to execute when users visit the site[67]. This can result in the theft of sensitive data or control over the victim's browser. SQL injection is another standard attack where an attacker manipulates a web application's input to execute arbitrary SQL commands. This can result in data breaches and, in some cases, complete data loss.

Distributed Denial of Service (DDoS) attacks involve overwhelming a server with traffic to render it unavailable, making it impossible for users to access the website. File inclusion attacks allow attackers to include files from remote servers on a web application, providing them with unauthorised access [68].

Another critical gap in web application security is the lack of awareness among organisations regarding the importance of web application security. As a result, many organisations do not adequately understand the risks associated with web applications, which leads to inadequate security measures. Furthermore, this lack of awareness can result in organisations failing to allocate the necessary resources and budgets to implement proper security controls.

A lack of awareness may also lead to the misconception that basic security measures such as firewalls and antivirus software are enough to protect web applications from attacks. However, these measures alone cannot secure web applications against more sophisticated attacks. For instance, a firewall cannot prevent an SQL injection attack [67]–[69].

Organisations must understand that securing web applications requires a multi-layered approach that includes identifying and mitigating vulnerabilities, implementing access control policies, and continuously monitoring and testing the application.

2.4.2 Challenges in Web Application Security

One of the critical issues facing web application security is the use of outdated software and systems. Unfortunately, many organisations continue to use outdated software and systems, which are vulnerable to malware attacks. In addition, the systems often lack critical security updates and patches, making them an easy target for attackers [48]. Another challenge is the increasing use of cloud-based services and third-party applications. While these services provide numerous benefits, they also introduce new security risks.

Another issue is the lack of security testing during the development phase of web applications. Many organisations do not carry out adequate security testing, which leaves vulnerabilities in the application; this creates a significant security risk, as attackers can exploit these vulnerabilities to access sensitive information [49].

2.5 Related works

The related works section of the literature review provides an overview of existing research related to the topic under investigation. This section aims to contextualise the current study within the broader research landscape and identify gaps or limitations in the existing literature. By reviewing previous studies, the current research can build upon existing knowledge and identify areas for further investigation. This section provides a comprehensive and critical literature evaluation, highlighting key findings, methodologies, and limitations of previous studies. Overall, this section is essential for providing a foundation for the current research and establishing its significance and contribution to the field.

2.5.1 Enhancing Web Application Security through Technical Countermeasures

The security of web applications is a growing concern due to the increasing dependence of individuals and organisations on these applications. "Enhancing Web Application Security through Technical Countermeasures: An Empirical Study in the Zambian Context" by Katanga M and Mbwikalambo R is a research paper investigating the effectiveness of technical countermeasures in enhancing web application security in the Zambian context [70].

Katanga M and Mbwikalambo R utilised a mixed-methods research methodology comprising quantitative and qualitative data collection techniques. The quantitative data was collected through a questionnaire survey, while the qualitative data was obtained through interviews with IT professionals and experts in web application security. The study population consisted of 50 IT professionals and experts in web application security in Zambia.

Katanga M and Mbwikalambo R used various tools, including Burp Suite and Wireshark, to assess the effectiveness of technical countermeasures in enhancing web application security. The technical countermeasures tested included firewalls, intrusion detection systems, antivirus software, and vulnerability scanners.

Katanga M and Mbwikalambo R found that technical countermeasures such as firewalls and intrusion detection systems enhanced web application security. However, these measures' effectiveness depended on IT professionals' configuration and management

level. Katanga M and Mbwikalambo R also found that antivirus software and vulnerability scanners were ineffective in detecting and preventing attacks.

The research's discussion of the results highlighted the importance of implementing technical countermeasures as part of a comprehensive security strategy that includes organisational policies, user awareness, and regular security audits. They also emphasised the need to continuously monitor and manage technical countermeasures to ensure their effectiveness.

Katanga M and Mbwikalambo R found that technical countermeasures are important in enhancing web application security. However, their effectiveness depends on proper configuration and management by IT professionals. The study's findings have implications for developing effective web application security strategies in the Zambian context and beyond.

The research area not addressed by Katanga M and Mbwikalambo R is the lack of empirical evidence on the effectiveness of technical countermeasures in enhancing web application security in the Zambian context. This research, Strengthening Web Application Security Through Technical Measures, fills this gap by providing empirical evidence on the effectiveness of technical countermeasures and highlighting the importance of proper configuration and management in enhancing their effectiveness. However, the research is limited by its small sample size and focus on the Zambian context, which limits its generalisability to other contexts.

2.5.2 A Comparative Analysis of Web Application Security Measures

Alghathbar and Alruban's research paper titled "A Comprehensive Survey on Web Application Security" aims to provide a comprehensive review of the existing research on web application security [71]. The authors identify key security threats and vulnerabilities associated with web applications and evaluate the effectiveness of various security measures and techniques for mitigating these threats.

To achieve their aim, the authors used a systematic literature review methodology. First, they searched various databases such as IEEE Xplore, ACM Digital Library, and ScienceDirect to identify relevant research articles. The search was conducted using specific keywords related to web application security, such as "web application security," "vulnerabilities," and "attacks." The search criteria included articles published between 2015 and 2020 in English.

After an extensive search, the authors identified 152 research papers that met their inclusion criteria. The papers were then thoroughly screened based on the title, abstract, and full text. Finally, the authors used a standardised form to extract data from the selected papers, which included information such as the year of publication, the research aim, methodology, and key findings.

The authors used various tools and techniques to analyse the data extracted from the selected papers. They used a content analysis approach to identify and categorise the types of web application vulnerabilities and attacks and the security measures and techniques used to mitigate them. They also used descriptive statistics to summarise the review's findings and identify gaps or limitations in the existing research.

The findings of Alghathbar and Alruban's research paper reveal that web application security remains a significant challenge for organisations worldwide. The authors identified several common web application vulnerabilities, including SQL injection, cross-site scripting (XSS), and file inclusion attacks. However, they also found that various security measures could mitigate many of these vulnerabilities, including encryption, access control, and input validation.

The authors also highlighted the limitations of the existing research on web application security. They noted that most research focused on identifying vulnerabilities and proposing solutions without adequate empirical validation. Additionally, they found that most of the research was conducted in laboratory settings, which may not accurately reflect organisations' real-world challenges.

Overall, Alghathbar and Alruban's research paper provides a comprehensive overview of the existing research on web application security. Their findings highlight the importance of addressing the vulnerabilities and threats associated with web applications and the need for more empirical research to validate proposed solutions.

"Strengthening Web Application Security Through Technical Measures" fills in the gaps identified in Alghathbar and Alruban's (2021) research paper by proposing a practical approach to enhancing web application security through designing and implementing web applications on which the web applications vulnerability is tested. The proposed method can be validated through empirical testing, and the effectiveness of the implemented measures can be evaluated in a real-world context. Additionally, "Strengthening Web Application Security Through Technical Measures" provides a more

comprehensive analysis of the strengths and weaknesses of various security measures and techniques, further contributing to the existing body of knowledge on web application security.

2.5.3 A Comprehensive Survey on Web Application Security

The research work titled "Evaluation of Web Application Security Measures: A Case Study of Zambia" by Lumbwe C. and Lungu J. aims to evaluate the effectiveness of web application security measures in Zambia [72]. The research focused on assessing the level of vulnerability of web applications to attacks, identifying the types of attacks and evaluating the effectiveness of current security measures in mitigating these attacks [72].

Lumbwe C. Lungu used a case study approach to evaluate the effectiveness of web application security measures in Zambia. The study involved surveying 34 organisations in Zambia that use web applications. The survey was conducted using a questionnaire that collected data on the types of web application attacks the organisations had experienced, the security measures they used to protect their web applications, and the challenges they faced in implementing and maintaining these security measures.

Lumbwe C and Lungu J used a questionnaire as the primary tool for data collection. The questionnaire was designed to collect data on the types of web application attacks experienced by organisations, the security measures used to protect web applications, and the challenges faced in implementing and maintaining web application security measures.

Lumbwe C. and Lungu J. found that most organisations in Zambia use firewalls and SSL/TLS encryption as their primary web application security measures. However, the study also found that organisations faced several challenges in implementing and maintaining these security measures. Some of the challenges identified include a lack of skilled personnel to manage web application security, limited resources for implementing security measures, and a lack of awareness of web application security issues among users.

Lumbwe C. and Lungu J. highlight organisations' challenges in implementing and maintaining web application security measures. The study found that most organisations in Zambia rely on firewalls and SSL/TLS encryption as their primary web application security measures. However, these measures are insufficient to protect web applications from attacks. The study also found that organisations face several challenges in

implementing and maintaining web application security measures, including a lack of skilled personnel, limited resources, and user awareness.

Lumbwe C. and Lungu J. conclude that organisations in Zambia must implement more advanced web application security measures to protect their web applications from attacks. The study recommends that organisations invest in skilled personnel, allocate sufficient resources to implement and maintain web application security measures, and raise awareness among users about web application security issues.

Lumbwe C. and Lungu J. provide valuable insights into the state of web application security in Zambia. However, there are some research areas that they did not address. One such gap is the lack of focus on technical countermeasures for improving web application security; this is where "Strengthening Web Application Security Through Technical Measures" research comes in. The study aims to evaluate the effectiveness of technical countermeasures in improving web application security in the Zambian context. The research provides valuable insights into the effectiveness of technical countermeasures in enhancing web application security in Zambia. It fills in the research gap left by the study by Lumbwe C. and Lungu J. by providing a more in-depth analysis of technical countermeasures and their effectiveness in improving web application security.

2.5.4 Web Application Security Measures

The research work on "Web Application Security: A Review of Current Research" by Singh and Shekhar aims to comprehensively review the current web application security research [73]. The authors identified the gaps in the existing literature and presented an overview of the latest research in this field.

The research methodology employed in the study was a systematic review of the relevant literature from various sources, including academic journals, conference proceedings, and books. The authors used specific search terms and inclusion and exclusion criteria to identify relevant studies. They then analysed the selected studies and synthesised the findings to provide a comprehensive review of the current state of research in web application security.

The authors used various tools to evaluate the selected studies, including qualitative, statistical, and content analysis. They used these tools to identify the most critical issues and trends in web application security.

The research findings highlighted the significant challenges and issues facing web application security, including cross-site scripting (XSS) attacks, SQL injection attacks, and session hijacking [73]. The research also identified emerging trends in web application security, such as machine learning and artificial intelligence, for detecting and preventing web application attacks.

The discussion of the results showed that the research community had made significant progress in addressing the security concerns of web applications. However, the authors identified several research gaps that require further attention. For instance, there is a need for more research on the security of mobile web applications and the impact of new technologies, such as blockchain and the Internet of Things (IoT), on web application security.

The research's conclusions highlighted the need for a holistic approach to web application security that incorporates technical, organisational, and human factors. The authors suggested that the best way to enhance web application security is to adopt a defence-in-depth approach that combines various security measures, including secure coding practices, network security, and user education.

Overall, Singh and Shekhar provide a valuable contribution to the field of web application security by presenting a comprehensive review of the current research and identifying the gaps in the existing literature. The findings can help researchers and practitioners develop more effective and efficient web application security solutions.

Compared to "Strengthening Web Application Security Through Technical Measures," Singh and Shekhar provide a broader overview of the current research in web application security, while the former focuses on the empirical evaluation of technical countermeasures in a specific context. As a result, Singh and Shekhar provide a more comprehensive understanding of the various issues and challenges facing web application security and highlight the need for a holistic approach to web application security.

2.5.5 Web Application Security

Web Application Security: How to Avoid Authentication Attacks Akamai Technologies by Akamai Technologies Inc. is a white paper that provides insights on preventing authentication attacks on web applications [74].

The research aim of the white paper is to guide how to protect web applications from authentication attacks, which are among the most common types of attacks on web applications. The paper is based on analysing attack data gathered by Akamai Technologies. The data was collected from over 330 billion daily requests, making it a reliable source for analysis.

The methodology used by Akamai Technologies in this white paper is data analysis. The data was collected using Akamai's Intelligent Edge Platform, a distributed server network that delivers content and applications over the Internet. In addition, the platform includes a web application firewall, Distributed Denial of Service (DDoS) protection, and bot management. The collected data was then analysed to identify trends and patterns in authentication attacks.

The tools used in this research are Akamai's Intelligent Edge Platform and its Security Operations Centre (SOC). The SOC is a team of security experts who monitor and respond to real-time security incidents. In addition, they use advanced security tools and techniques to identify and mitigate security threats.

The research findings indicate that authentication attacks are a significant threat to web applications, with an increase in attacks of over 54% from 2019 to 2020. The most common authentication attacks are credential stuffing, brute-force attacks, and password spraying. The research also found that most attacks originated from botnets and anonymous proxies, making identifying and blocking attackers difficult.

The discussion of the results in the white paper emphasises the importance of implementing strong authentication mechanisms, such as multi-factor authentication, to prevent attacks. The paper also suggests using risk-based authentication, which analyses user behaviour and other contextual data to assess the risk of a login attempt. Additionally, the paper recommends monitoring user accounts for suspicious activity, such as failed login attempts, and implementing rate limiting to prevent brute-force attacks.

The white paper concludes that authentication attacks are a growing threat to web applications and that organisations must implement technical measures to protect their applications. The paper suggests that organisations should implement a defence-in-depth approach that includes multiple layers of security, such as web application firewalls, DDoS protection, bot management, and advanced authentication mechanisms.

The research gap in this white paper is the lack of discussion on the effectiveness of the technical measures suggested. While the paper provides insights on how to prevent authentication attacks, it does not provide empirical evidence on the effectiveness of these measures. Strengthening Web Application Security fills this gap Through Technical Measures, which provides empirical evidence of the effectiveness of technical measures in preventing attacks.

Web Application Security: How to Avoid Authentication Attacks Akamai Technologies by Akamai Technologies Inc. (2022) is a valuable resource for organisations looking to protect their web applications from authentication attacks. The research methodology used by Akamai Technologies is reliable, and the findings provide insights into the most common types of authentication attacks and their origins. In addition, the paper's discussion and recommendations provide practical guidance on preventing attacks, although it lacks empirical evidence on the effectiveness of these measures.

2.6 System Design Methodology

System Design Methodology is a comprehensive and structured process that facilitates the creation of complex systems. It comprises several stages, each of which plays a vital role in ensuring the final product meets its intended objectives. The first phase involves identifying the system requirements, both functional and non-functional. This stage requires a thorough understanding of the system's purpose, scope, and context. Once the requirements are identified, they are analyzed to determine the best design options. This analysis considers factors such as feasibility, cost, and performance.

The next stage involves selecting the most appropriate design from the available options. This decision must be based on a careful evaluation of the pros and cons of each alternative. Once the design is chosen, it is documented in detail to facilitate implementation. Documentation includes specifications, drawings, and other relevant information required for building the system.

Testing is another crucial part of the System Design Methodology. Testing helps identify defects, errors or inadequacies in the design before implementation. In addition, this stage ensures that the system works as expected and meets all specified requirements.

Finally, evaluation ensures the system meets the end user's needs and expectations. The evaluation entails assessing the system's performance in real-world settings, considering

user feedback. Based on the evaluation results, modifications may be made to the original design to improve system performance.

System Design Methodology is an essential process for developing complex systems. It provides a systematic approach to designing, implementing, and evaluating systems, ensuring they meet their intended purposes effectively and efficiently. By following this methodology, designers can minimize the risks associated with designing complex systems while maximizing the chances of delivering successful products.

2.6.1 WaterFall Model

The Waterfall Model is a widely used system development model adopted in this research to design a test site with vulnerabilities and a non-vulnerable site to access and encounter the vulnerabilities, respectively. The Waterfall Model is a sequential design process in which progress flows in one direction, like a waterfall, through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Implementation, and Maintenance. This model is often used for software development, where the product is a complete software system.

The Waterfall Model differs from other system development models, such as the Agile and Spiral. The Agile Model is an iterative, incremental approach to software development that emphasises flexibility, rapid delivery, and continuous improvement. It is often used for projects where requirements change frequently, and the customer wants to see results quickly. On the other hand, the Spiral Model is a risk-driven model that combines the iterative nature of the Agile Model with the systematic, controlled aspects of the Waterfall Model. As a result, it is often used for large, complex projects with a lot of uncertainty and risk.

The Waterfall Model has several advantages and disadvantages compared to other system development models. One of the advantages is that it is easy to understand and use, making it suitable for projects that require a structured approach. The model is also easy to manage, with clear milestones and deliverables that can be tracked and measured. Another advantage is that it is suitable for large, complex projects with well-defined requirements, where a clear understanding of the product is necessary before development can begin.

However, the Waterfall Model has some disadvantages as well. One of the main disadvantages is that it can be inflexible, with little room for changes or modifications

once development has begun. This can lead to delays and cost overruns if changes are required later in development. Another disadvantage is that it can be difficult to determine the requirements accurately at the beginning of the project, leading to errors and problems later in the development process.

Despite the disadvantages, the Waterfall Model was chosen as the methodology for this research because it provides a structured approach to designing a test site with vulnerabilities and a non-vulnerable site to access and encounter the vulnerabilities, respectively. The model allows for a clear understanding of the end product before development begins, which is necessary to ensure the research objectives are met. In addition, the model is suitable for the size and complexity of the project and provides clear milestones and deliverables that can be tracked and measured throughout the development process.

2.6.1.1 System Requirements

This research uses the Waterfall methodology as the system development model. This methodology assumes that all project functional and non-functional requirements can be gathered and understood upfront. The project manager's role is to obtain a detailed understanding of the project sponsor's requirements, and written requirements are typically contained in a single document that outlines each stage of the project, including costs, assumptions, risks, dependencies, success metrics, and timelines for completion.

In the design phase, software developers create a technical solution to the problems outlined in the product requirements, including scenarios, layouts, and data models. A higher-level or logical design is first created, describing the purpose and scope of the project, the general traffic flow of each component, and integration points. Then, this design is transformed into a physical design using specific hardware and software technologies.

During implementation, programmers code applications based on project requirements and specifications, with some testing and implementation also taking place. This phase may be the shortest of the Waterfall process because careful research and design have already been done. If significant changes are needed at this stage, it may mean returning to the design phase.

Verification of testing is essential before a product can be released to customers to ensure that the software has no errors and that all of the requirements have been met, resulting

in a good user experience. The testing team will utilise design documents, personas, and user case scenarios provided by the product manager to develop their test cases.

Finally, in the deployment and maintenance phase, a team will be assigned to take care of updates and release new versions of the software as defects are found and change requests come in from users. Despite its advantages, such as simplicity and ease of use, the Waterfall methodology also has drawbacks. It can be rigid and inflexible, making it challenging to adapt to changing project requirements. However, it remains a widely used system development model due to its many benefits.

Given the nature of this research, which aims to design a test site with vulnerabilities and a non-vulnerable site to encounter them, the Waterfall model's rigidity and emphasis on upfront requirements and design is a significant advantage. In addition, this methodology allows for a systematic and comprehensive approach to designing and developing the test sites, ensuring that all necessary components are accounted for and thoroughly tested before deployment. Therefore, the Waterfall methodology is a suitable choice for this research.

2.6.1.2 Operating Systems and Other Software

In addition to the development process, the research will require specific operating systems and software to be installed on the test environment. The operating system requirements for the test environment include Windows Server 2016 and Ubuntu 20.04 LTS. These operating systems were chosen because they are widely used and have extensive support from their respective communities.

Windows Server 2016 and Ubuntu 20.04 LTS were chosen for the test environment due to their popularity and extensive support from their respective communities. Windows Server 2016 is a widely used server operating system that provides a secure and scalable platform for running business-critical applications. It also includes enhanced security, improved performance, and better management tools. Ubuntu 20.04 LTS, on the other hand, is a popular open-source Linux distribution known for its stability, security, and ease of use. It also comes with a vast collection of software packages, making it ideal for setting up a web application environment.

The choice of these operating systems is significant because they provide a stable and reliable platform for developing and testing web applications. They also have extensive community support, so any issues encountered during the research can be quickly

resolved. Additionally, using popular operating systems ensures that the research results are relevant and applicable to a broader audience, as these operating systems are widely used in the industry.

Other software requirements include the installation of a web server, database management system, and programming languages. The web server used in this research will be Apache HTTP Server version 2.4. The database management system used will be MySQL version 8.0.26. Apache and MySQL are open-source software widely used in web development.

In addition, programming languages such as HTML, CSS, JavaScript, and PHP will be used to develop the test site. These languages are widely used in web development and will allow for the creation of a functional and realistic test environment.

It is important to note that the operating systems and software were selected based on their popularity and extensive community support; this ensures that any issues encountered during the research can be addressed promptly and solutions can be found with the community's help.

2.7 Summary of Chapter 2

The literature review chapter analysed several research works on web application security, including those on security measures, current research, and authentication attacks. It revealed gaps in the research and how they can be filled through technical measures, which will be explored in the methodology chapter. The methodology chapter will detail the research design, sample, data collection, and analysis procedures used to investigate the effectiveness of technical measures in enhancing web application security.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Overview

The methodology chapter presents the approach taken to achieve the research objectives. The research design utilised in this study is a mixed-mode approach consisting of two phases[75]–[77]. The first phase involves a review of existing literature on web application security. The second phase involves using the waterfall model to design a test site with vulnerabilities and a non-vulnerable site to access and encounter the vulnerabilities. The mixed-mode approach provides a comprehensive understanding of web application security by examining theoretical and practical aspects [78]. This chapter provides a detailed description of the methodology employed to conduct this research.

3.2 Research Design

This research employs a mixed-mode research design consisting of two phases. The first phase involves a review of existing literature [71]–[74], [79] on web application security, which allows for a comprehensive understanding of the theoretical aspects of web application security. The second phase involves using the waterfall model to design a test site with vulnerabilities and a non-vulnerable site to access and encounter the vulnerabilities. This practical aspect of the research allows for evaluating the effectiveness of technical measures in strengthening web application security.

The research design begins with a thorough literature review on web application security [71]–[74], [79]; this involves identifying relevant literature from various sources, including academic journals, textbooks, and conference proceedings. The literature review helps to establish a theoretical framework for the research by examining the concepts, theories, and best practices related to web application security.

Following the literature review, the research design moves on to the practical aspect of the research. The waterfall model designs a test site with vulnerabilities and a non-vulnerable site. The test site is intentionally designed with vulnerabilities to simulate real-world situations and allow for the identification of weaknesses in web application security.

The non-vulnerable site is designed to access the vulnerabilities identified in the test site and encounter them; this allows for evaluating the effectiveness of technical measures in

strengthening web application security by identifying vulnerabilities and developing strategies to mitigate them.

The mixed-mode approach provides a comprehensive understanding of web application security by examining theoretical and practical aspects. This approach enables the researcher to develop a deeper understanding of web application security by considering the theoretical and practical implications of the research findings [75].

Overall, the research design in this research is carefully designed to provide a comprehensive understanding of web application security. Furthermore, the mixed-mode approach allows for the evaluation of technical measures in strengthening web application security, which is important in addressing the growing concerns related to web application security.

3.2.1 Test Site Structure

The test site created for this research is designed to function as an e-commerce platform where users can register, log in, browse, and post items for sale, much like a typical online marketplace. As shown in Figure 0.1., the home page serves as the starting point for users, from which they can easily navigate to any other page on the site.

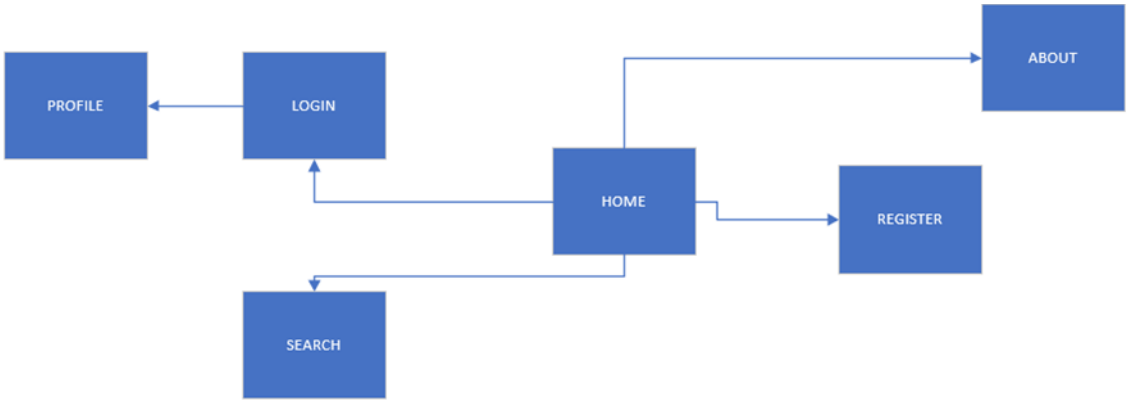


Figure 0.1: Test Site Structure

From the home page, users can navigate to the "Search" page to look for specific items or to the "About" page to learn more about the site and its functionalities. Users who wish to post items for sale or access other member-only features must register and create an account. After registration, users can log in, view their profile, and manage their account information.

The test site also includes a feature for users to post their items for sale. Once logged in, users can use this feature to add their items to the site, making them visible and available

to other users. There is also a feature for searching the available items based on specific keywords.

However, users must create an account to access additional features and functionality. This can be done by clicking the "Register" link in the navigation menu and providing the necessary information. Once the registration process is complete, users can log in using their credentials and gain access to the profile page.

The profile page allows users to view and manage their account information, such as username, password, and email address. In addition, users can perform various actions related to the web application, such as creating, editing, and deleting content.

The navigation structure of the web application is designed to provide a seamless and intuitive experience for users while also ensuring the security and privacy of user data.

3.2.2 Use Case Diagram

Use case diagrams are a visual representation of the functional requirements of a system and serve as a basis for design and development priorities shown in Figure 0.2. In this research, use case diagrams were used to describe the proposed functionality of the new system. The diagram below shows the web application use case diagram used in the investigation. Use cases focus on the interactions between the system and its users and are organised around functional requirements. They describe a sequence of events that leads to a goal and include both a main flow and alternate flows, which describe normal variations to the basic flow and unusual situations.

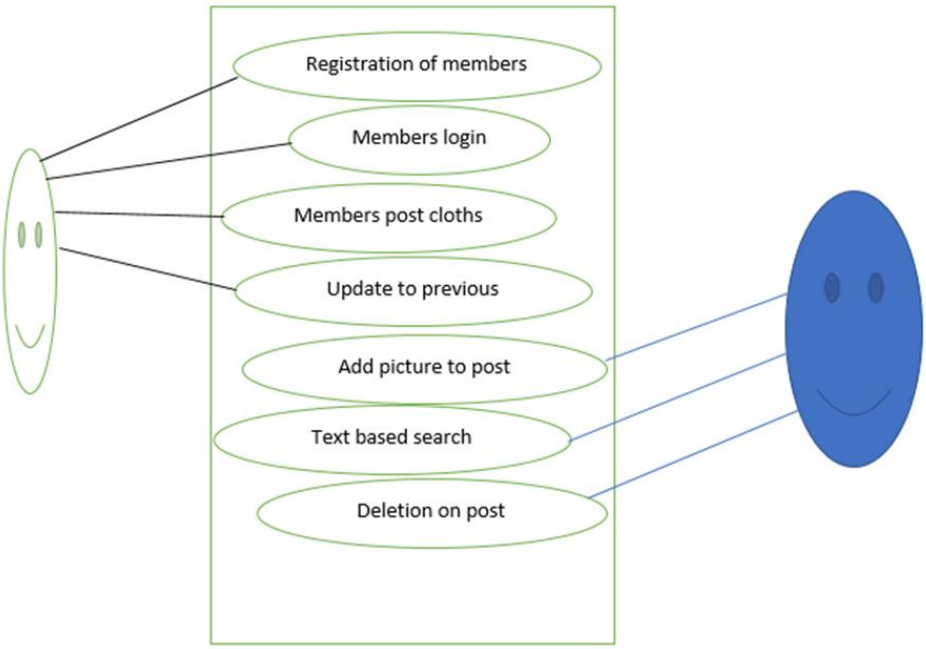


Figure 0.2: Test Web Application Use Case Diagram

The web application being investigated includes several use cases to support its functionality, as shown in Figure 0.2. One of the use cases is for the registration of members. This feature allows new members to create an account with the web application, which they can use to access other features. Another use case is for members to log in to their accounts. Once logged in, members can access their profile information and other application features.

Members can post their clothes on the web application using another use case. This functionality allows members to add clothing items to the platform and share them with other users. Members can also update their posts if they need changes or corrections. Another feature is the ability to add pictures to posts, making the clothing items more visually appealing to other users.

The web application also includes a text-based search use case; this allows users to search for specific clothing items using keywords or other relevant search criteria. In addition, if a user no longer wishes to keep their post, they can delete it using the deletion on the post use case. This functionality helps keep the platform organised and clutter-free by removing unwanted content.

Overall, these use cases help users with a range of functionality to help them navigate the platform and connect with other users. The registration, login, and profile features allow users to access the platform, while the posting and search features help users

connect and engage with each other. The ability to add pictures and update posts helps keep content fresh and engaging, while the ability to delete posts helps keep the platform organised and clutter-free.

3.2.3 Flow Chart

In this research, the process of password enumeration was carried out using a systematic approach, as shown in. The first step was to perform an SQL injection attack that would generate an error message, thereby revealing the table name on the website; this was followed by another injection attack to display all the tables within the named database. Finally, we could enumerate the tables' columns by gaining access to the tables.

Once the columns were identified, we proceeded to view the content of the columns, allowing us to extract the required password from the login page on the email address field. This process was achieved through carefully executed steps, ensuring the password enumeration was conducted effectively and efficiently.

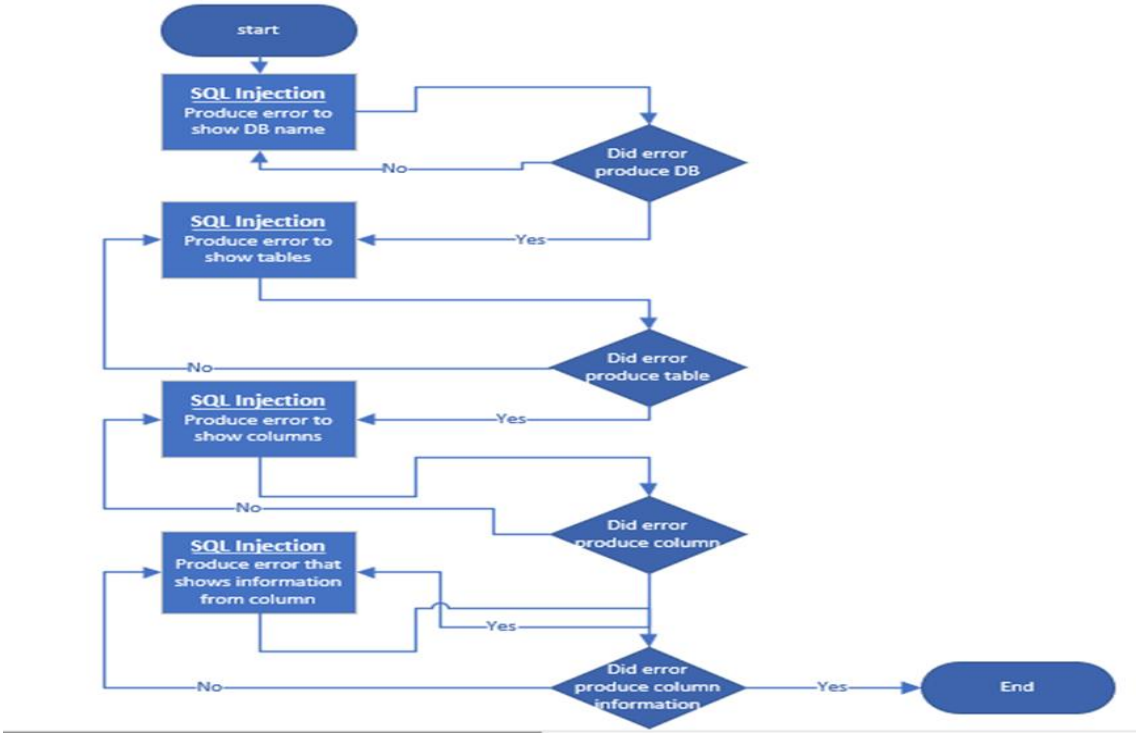


Figure 0.3: Test Web Application Flow Chart

It is important to note that this approach to password enumeration was carried out to investigate the website's vulnerability to SQL injection attacks. The findings from this research can help website developers identify and mitigate the risk of SQL injection attacks, thereby enhancing the security of their websites. In addition, this research

underscores the importance of conducting regular security assessments to identify and address any vulnerabilities within websites or other applications.

3.2.4 Entity Relationship Diagram

The Entity-Relationship (ER) model is a powerful tool used to identify entities represented in a database and define how these entities are related. In this model, Figure 0.4, an enterprise schema is specified that represents the overall logical structure of a database graphically. One of the main benefits of using the ER model is that it allows real-world objects like a person, a car, or a company to be modelled along with the relationships between them.

The ER model comprises E-R diagrams that visualize the relationships between entities in a database. These diagrams are easy to convert into relations (tables), which makes them very useful for designing and implementing databases. In addition, the purpose of real-world modelling of objects makes them an essential tool for database designers and developers.

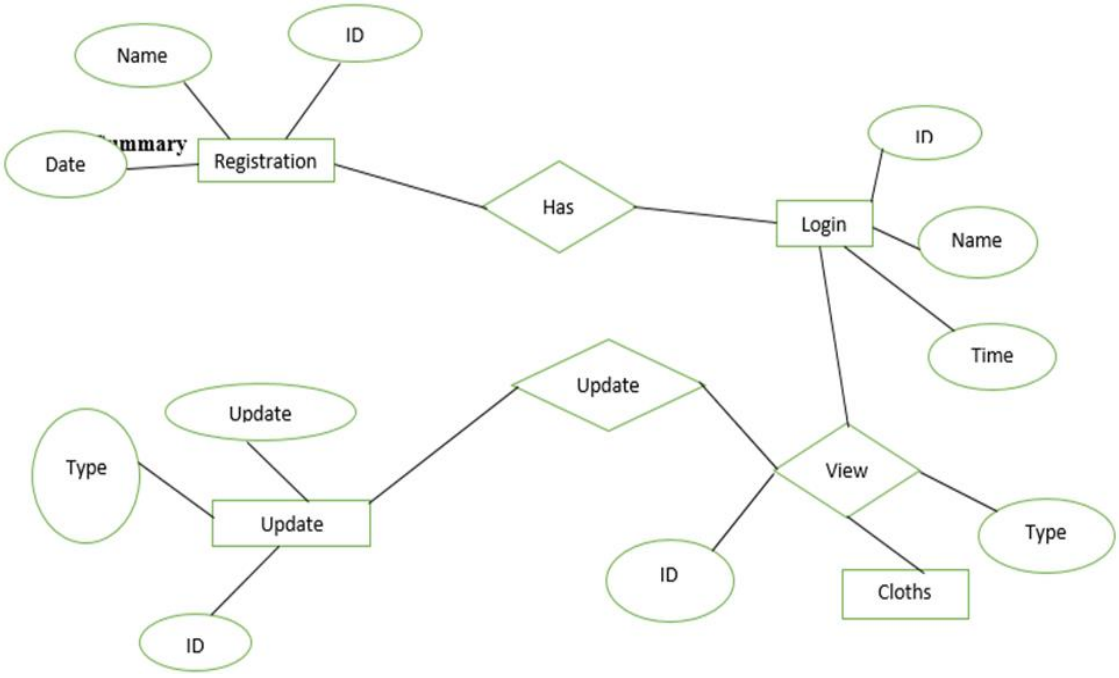


Figure 0.4: Web Application 3.2.4. Entity Relationship Diagram

One of the key features of E-R diagrams is that they require no technical knowledge or hardware support. As a result, they are straightforward to create even by a novice user and are easy to understand. In addition, these diagrams provide a standard solution for visualizing the data logically, which is essential in database design and development.

Overall, the ER model and E-R diagrams provide a powerful toolset for representing and visualizing complex data relationships in a database.

3.5 Research Tools

3.5.1 Hardware Tools

In this research, various hardware tools were used to support the investigations. One of the primary tools is the Dell Precision Tower 5820. This high-performance workstation is designed to meet the needs of professionals who require advanced processing power and graphics capabilities. It features an Intel Xeon processor and up to 256GB of RAM, making it ideal for complex data analysis tasks. The Dell Precision Tower 5820 was used to run software programs, analyze data, and generate reports.

Another device used in this research is the Lenovo ThinkPad X1 Carbon. This lightweight, durable, portable laptop is ideal for fieldwork and on-site investigations. In addition, it features an Intel Core processor, up to 16GB of RAM, and a fast solid-state drive, which makes it ideal for running software programs and capturing data. The Lenovo ThinkPad X1 Carbon will collect data, conduct interviews, and perform on-site investigations.

In addition to these devices, the research team used various peripheral devices to support their investigations. One such device is the HP OfficeJet Pro 9025e All-in-One Printer. This printer features advanced printing technology and can produce high-quality prints quickly and efficiently. It will be used to print out reports, presentations, and other documents related to the research.

Another peripheral device used in this research is the Seagate Backup Plus Hub. This external hard drive has a large storage capacity and fast data transfer rates. It will back up important data, such as research files, reports, and presentations, to ensure they are not lost or corrupted.

These hardware tools enabled the research team to conduct their investigations effectively and efficiently. Using high-performance workstations and portable laptops, they can run software programs, collect data, and analyze results from anywhere. In addition, the peripheral devices, such as printers and external hard drives, will provide additional support for storing, organizing, and sharing data related to the research.

3.5.2 Software Tools

This research uses various software tools to support the testing and analysis process. The software tools are specifically selected based on their compatibility with the hardware tools used and their ability to support the research objectives.

One of the primary software tools used in this research is Burp Suite, a popular web application security testing tool. It is a comprehensive platform that includes various modules such as scanner, proxy, repeater, and intruder used to perform different types of tests on web applications. Burp Suite is compatible with multiple operating systems and is flexible enough to handle various security testing requirements.

Another key software tool this research uses is Kali Linux, a specialized operating system designed for penetration testing and ethical hacking. Kali Linux provides pre-installed security tools frequently used in security testing, including Nmap, Wireshark, and Metasploit. In addition, Kali Linux is designed to provide an environment optimized for security testing and compatible with various hardware tools.

In addition to these primary tools, other software tools are used in this research. For example, the research required specific programming languages like Python, C++, JavaScript, and PHP to develop custom scripts to execute certain attacks. Additionally, various virtualization, containerization, and automation tools streamline the testing process and increase efficiency.

In this research, Dirbuster and Wireshark were also utilised as research tools. Dirbuster is a directory and file brute-forcing program often used to check web application security. It is intended to assist in discovering hidden or incorrectly configured files and directories on a web server. Dirbuster attempts to access various popular directories and file names and analyses the server's replies. This tool is especially helpful for locating sensitive files or directories that may not be easily accessible via standard browsing.

Wireshark is a network protocol analyzer that permits researchers to record and analyse network data in real-time. It gives precise information on the protocols being utilised and the data being communicated and can assist in identifying any security flaws or irregularities. Wireshark can analyse wired and wireless network traffic and supports many network protocols. In the context of this study, Wireshark can be used to analyse network communication between the web application and its clients, hence facilitating the identification of potential security vulnerabilities or suspicious actions.

By integrating Dirbuster and Wireshark into the software toolkit, researchers can improve their ability to find vulnerabilities and evaluate the network communication component of the tested web application. These tools supplement the existing software tools, such as Burp Suite and Kali Linux, to give a thorough and effective approach to web application security testing and analysis.

Overall, the selection of software tools for this research is based on their compatibility with the hardware tools being used and their ability to support the research objectives. By utilising a range of software tools, researchers can effectively test and analyse the security of web applications, identify vulnerabilities, and develop effective mitigation strategies.

3.6 Web Application Attacks

This research performed various web application attacks, including SQL injection, database enumeration, directory traversal, dir buster, and password sniffing. The study aims to identify vulnerabilities in web applications and provide insights into the methods used to exploit them. A comprehensive research methodology involved hardware and software tools and ethical hacking techniques. The research methodology used in this research is designed to ensure the validity, reliability, and accuracy of the results obtained and minimize any potential risks associated with performing web application attacks.

3.6.1 SQL Injections

In the current section of the research, the SQL injection technique known as "Bypassing Login" was performed, as shown in Figure 0.5. This section showcases the command and the outcomes after executing the injection.

The Bypassing Login attack was executed by inserting a specific SQL command, which resulted in the web application authenticating the first user of the system without the need for valid login credentials.

Figure 0.5: Bypassing Login Attack

The attack code is displayed in Figure 0.5, demonstrating the profile page being loaded despite no username or password being entered. This information is crucial for understanding the vulnerability of the web application and how attackers can exploit it.

The SQL injection attack, Bypassing Login," can be further simplified by using a shorter SQL command that achieves the same outcome of logging in as the first user of the system. This approach is illustrated in the query shown in Figure 0.6.

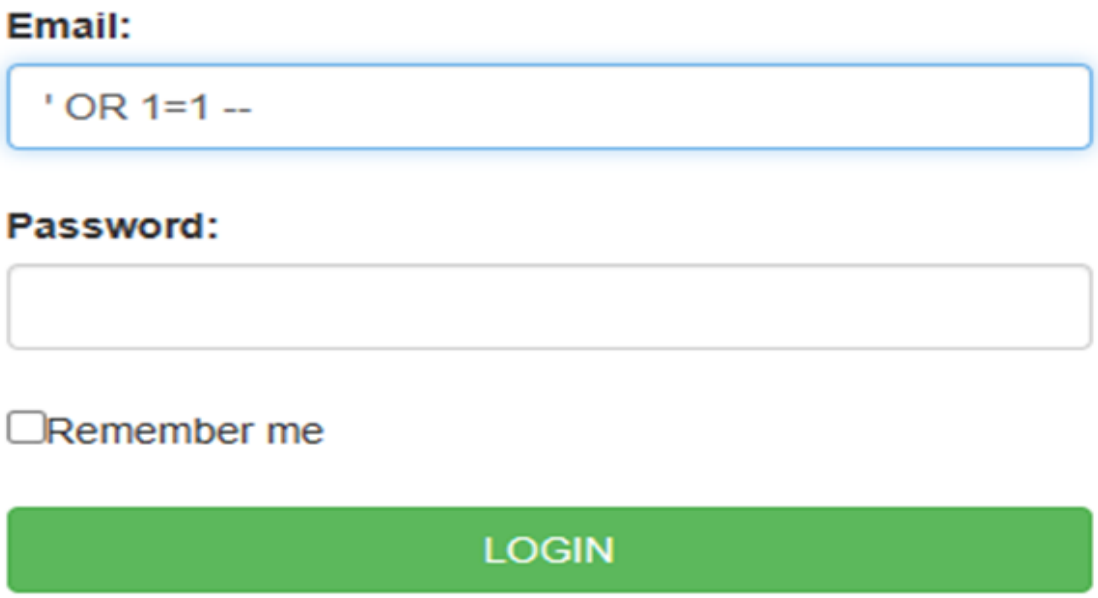


Figure 0.6: Shorter SQL Bypassing Login Attack

3.6.2 Database Enumerations

In this attack, the concept of database version fingerprinting is introduced to gather information about the database server used by the web application. The main idea behind

this technique is to force the database server to generate errors that can potentially reveal information about the database itself.

A specific query was used to perform database version fingerprinting, as shown below:

```
" AND(SELECT 1 FROM(SELECT
COUNT(*),concept(version(),FLOOR(rand(0)*2))x FROM
information_schema.TABLES GROUP BY x)a)—"
```

Listings 1: Shorter SQL Bypassing Login Attack Code Snippet

However, due to the lengthy query, it is not fully visible in Figure 0.7. Therefore, the complete query is pasted in the appendix.

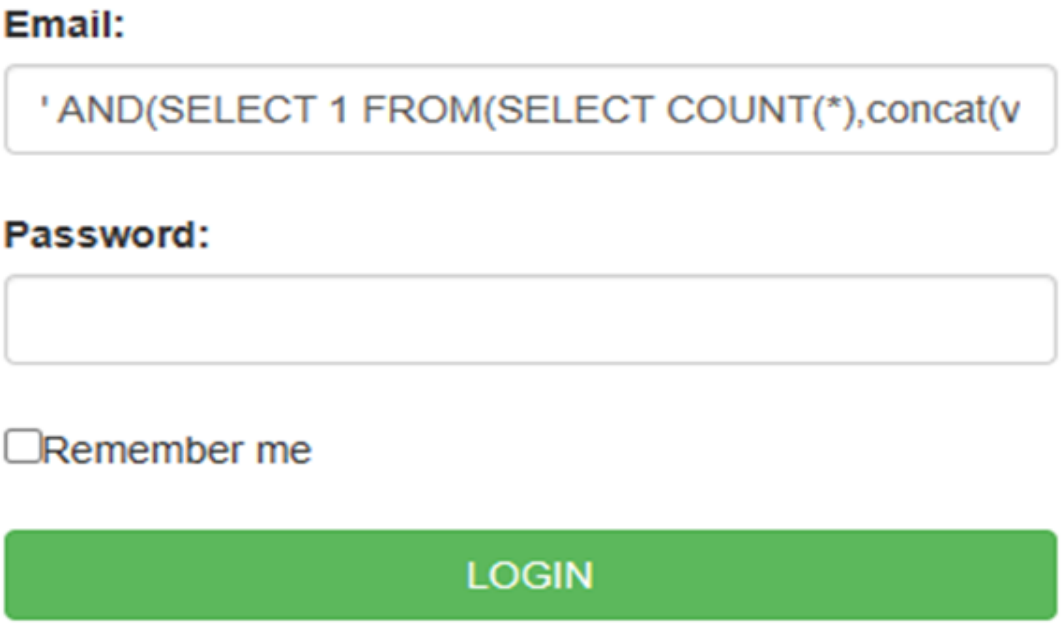


Figure 0.7: Database Enumerations Attack

We took this further and listed the database name itself. Using the query shown below:

```
' AND extract value(rand(),concat(0x3a,(SELECT
concat(0x3a,schema_name) FROM information_schema.schemata
WHERE schema name LIKE '%quick%' )))--
```

Listings 2: Database Enumerations Attack Code Snippet

In the query above, we specifically searched for the database with any name like "quick" because the website name is "quick thrift," and the Figure 0.8, the database name was quick Thrift, as shown below in the error message.

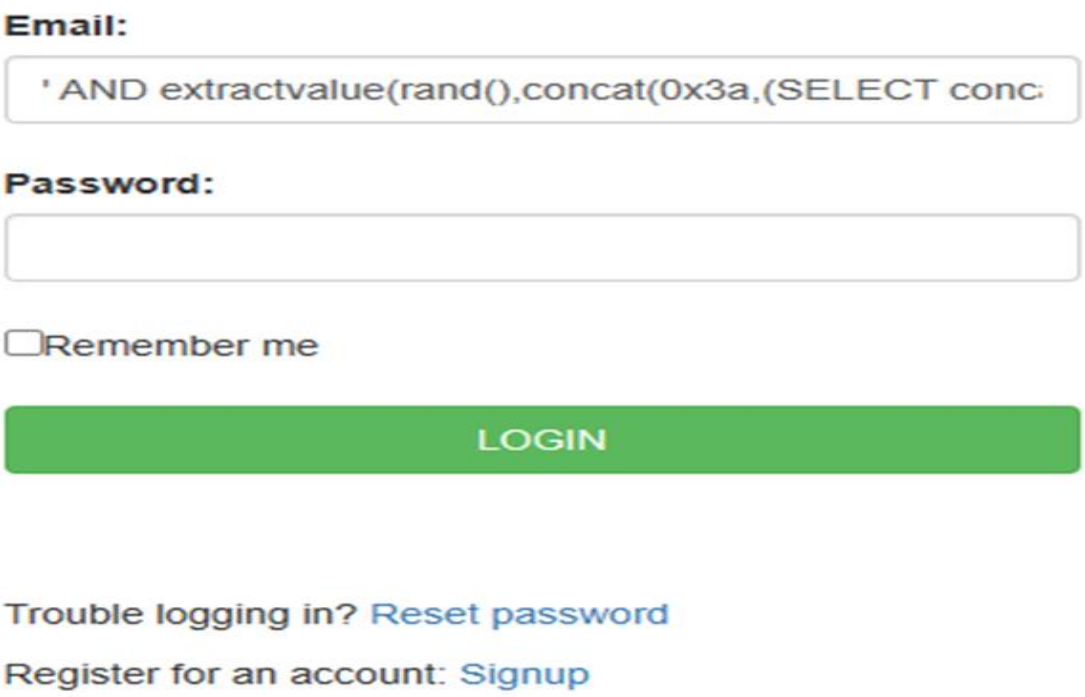


Figure 0.8: Database Name Fingerprinting Attack

3.6.3 Directory Transversal Attack

In this attack, the concept of directory traversal is explained, and how hackers can use it. Directory traversal refers to navigating through different folders on the webserver from the attacker's computer, allowing them to access confidential user information potentially and even source code files of the web application. While server-side code such as PHP is not typically visible to users, other regular files such as media and text files can still be accessed by the attacker, even if they are not intended to be shared. This vulnerability is highlighted as a potential security risk, and solutions or recommendations are proposed to mitigate the issue. The attack can be performed by inserting specially crafted input that can manipulate the directory path and navigate to higher-level directories, granting unauthorized access to sensitive files and information. Figure 0.9 shows the Dirbuster attack setup used.

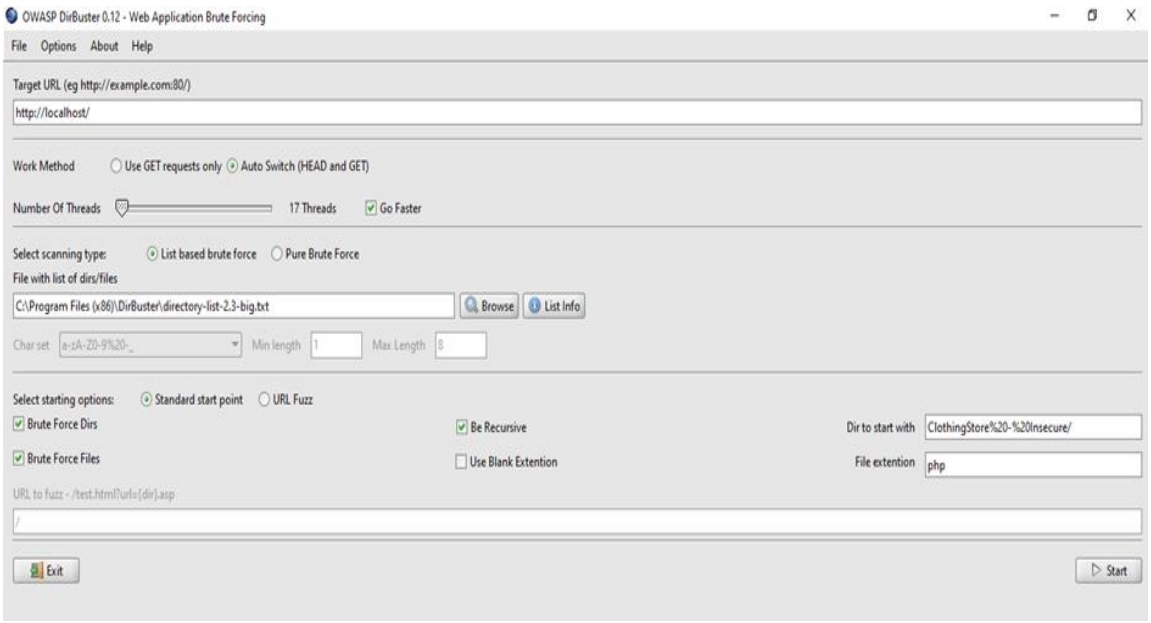


Figure 0.9: DirBuster Attack Setup

The DirBuster is a software developed by the Open Web Application Security Project (OWASP) that was utilized for penetration testing to identify web server directories and files. With this tool, a hacker can gain insight into all the files and directories on a given URL, including sensitive information that can be exploited in an attack.

The DirBuster tool was employed in this research to understand better the files and folders on a website being developed. By utilising the tool, the researcher was able to simulate potential attacks and determine which files and directories may be visible to a potential hacker. This information can be used to identify and strengthen any weaknesses or vulnerabilities in the web application's security, ultimately improving the system's overall security.

3.6.4 Password Sniffing Attack

In this attack, the packet sniffing technique is discussed, which involves the interception and analysis of network traffic. In this case, the tools used for packet sniffing were TCP Dump and Wireshark. The test aimed to determine whether any sensitive information about the web application was being transmitted over the network in clear text.

The tools were installed on the system to perform the test, and network traffic was captured during the authentication process. The objective was to see if password information such as email addresses and passwords were transmitted over the network in

plain text and any other information from the website forms, such as search queries after authentication.

Analysing the captured packets using Wireshark made it possible to identify any instances where sensitive information was transmitted in clear text. This information is crucial for understanding the web application's security and taking steps to protect against packet sniffing attacks.

3.6.5 Packet Sniffing attacks

Packet Sniffing allows capturing and analyzing network traffic transmitted over the network. This attack aimed to determine whether any sensitive information, such as login credentials or form data, was being transmitted in plaintext over the network.

To perform the test, the researcher applied various filters to isolate and capture specific packets of interest. For instance, the "frame contains" command was used to search for specific keywords such as "password" or "email" that are likely to be related to sensitive data. Additionally, filtering by HTTP was applied to capture only the packets related to HTTP requests and responses.

The captured packets were analysed to determine whether any sensitive information was transmitted in plaintext.

The research applied Wireshark, a widely-used packet sniffing tool, to conduct packet sniffing assaults. Wireshark enables the collection and real-time analysis of network traffic, allowing the researcher to inspect the contents of individual packets and locate any sensitive information carried over the network.

Wireshark was used to isolate and collect packets of interest using various filters. Filters such as "frame contains" were used to look for often associated sensitive data keywords such as "password" or "email." In addition, the HTTP protocol was also filtered to capture packets about HTTP requests and responses.

After capturing the packets, we analysed their contents to determine if any sensitive data was transferred in plaintext. This research assists in identifying potential security flaws, such as situations in which sensitive data is sent without encryption.

By integrating Wireshark into our study technique, we were able to successfully run packet sniffing assaults and analyse the security of network traffic, focusing on the delivery of sensitive data.

3.7 Countermeasures Solutions

3.7.1 SQL Injections Solutions

SQL Injections were possible because of the use of dynamic queries that depend on input from the user to be inserted and appended as part of the query. The SQL injections can be made to do several things, as shown in this document. The common problems this research addressed in the source code are shown below.

3.7.2 Input Validation

Input Validation refers to having checks that can be carried out on data before it is sent to the server. Any data that is inconsistent with the requirements should be rejected. This was done both on the server side and the client side.

3.7.3 Client-Side Validation

Client Side Validation makes the checks occur within the web browser using client-side languages such as HTML and JavaScript. Ideally, both implementations should be in place. Figure 5.1 shows how changing the text box in HTML code from text to email forces a validation on the email field that would prevent a hacker from typing an unrecognised email format into the web application.

```
<label for="email">Email: </label>
<input type="email" class="form-control" name="email" />
```

Figure 0.10: Client-Side Validation

By changing the type, the web browser would be checking to make sure that the user has inserted text with an @ symbol and a (.) dot that would make sure that the syntax is correct. In addition, we would also ensure that all the fields in HTML are inserted on the form before submitting the request to the server by adding the "required" keyword at the end of each input line.

3.7.3.1 Server-Side Validation

The server code in PHP could also be written to check whether the input is correct. An example would be the email field, where the email address field is used for SQL injection attacks. The $_POST['email'] variable in the PHP code can be first checked before being

processed by the script. The processing would look for the format of the text inserted and then determine whether the input follows the email address syntax, as shown below.

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
  $emailErr = "Invalid email format";
}
```

Figure 0.11: Server-Side Validation

3.7.3.2 Input Sanitisation

Input sanitization refers to adding functions in the source code that can remove any possibly harmful content that would not work well with the source code of the web application. We have used several built-in PHP functions for this, and I explain the reasoning behind each function below.

1. **Trim()**

This function was used to remove any additional white spaces that input from the form could have. The spaces for all characters are reduced to one. This helped in multiple situations, such as the SQL injection " 'Or 1=1-- " which requires two spaces at the end to recognize the minus symbols as comments. By removing the spaces, the comment is not in effect, which would require including the password.

2. **Strip slashes()**

The function removed back slashes sometimes used in code as escape characters. For SQL injections that use quotation marks, the hacker can add back slashes to prevent the inner quotation marks the developer would have included in the code. This function removes the slashes so that quotes work exactly as intended by the developer.

3. **Htmlspecialchars()**

The function does not allow for content that could be considered HTML code from being rendered on the web application. In short, it reduces the ability to render HTML code on the web page. The result was that the web application would display the code the hacker inserts rather than execute it.

4. **Strip_tags()**

The function was used to insert HTML tags as part of the input on the HTML form elements, such as the text box. The function removes text surrounded by angle brackets, thereby preventing the execution of code. This prevents even cross-site scripting attacks because they require using the <script> tags as part of the input.

5. **Mysqli_real_escape_string()**

The mysqli_real_escape_string function was specifically created to inspect input and then remove the ability of the SQL code from being executed on the database server. By having all input from the text box passing through this function, we can be assured that the web application will not forward SQL code to the database server, which would then execute it.

Figure 5.4 shows how we implemented the functions in the conn.php file so that all the files connecting to the database could reuse the same function. This function was named sanitise and returned the value of any input after passing them through the sanitization functions.

```
function validate($secure)
{

        $validated = sanitization($secure);//pass all content from the form  in a sanitization function
        return $validated;
}
function sanitization($argument)//for Cross site Scripts and SQL injections
{
    $field= htmlspecialchars($argument);//preventing HTML code execution

    $field = trim($argument);//removing extra spaces from input

    $field = stripslashes($argument);// removing backslashes used as escape characters

    $field = mysqli_real_escape_string($argument);//preventing SQL code execution

    return $argument;//process the final filtered text

}
```

Figure 0.12: The mysqli_real_escape_string function

The main reason is to have one function that can be called in all the other scripts. The one function can then implement the functions above, with each function supplying its output to the next function. When all the functions clean the text, the function can then send the data back to the script to be used.

The code in Figure 5.6 prevented all of the SQL injections I demonstrated on top from being entered on the input field, as shown below.

Figure 0.13: SQL injections prevented

Figure 5.8 above, including the other images in the SQL section, would not work because of the validation and sanitization code. This prevented the following:

a) Bypassing log-in using SQL injections
b) Forcing errors through SQL injections
c) Enumeration of the database name
d) Enumeration of database tables
e) Enumeration of database columns
f) Enumeration of column information

All 6 vulnerabilities listed here were prevented using the code shown in this section.

3.7.4 Directory Traversal

Three things were considered when protecting the web application from directory traversal attacks.

1. **Permissions on the server**

We used the operating system to specify the permissions for the users on the system regarding access to the files. This is in terms of read, write and execution permissions and the ability to traverse for directories in the case of Windows.

The operating system can limit what anyone can do with the files, whether they can open them (read permission). For example, in Linux, we can run the command “chmod o-r

images/” to remove the ability for people not logged on to the server from opening the images directory. The. htaccess file on the web server can also override the permissions set by the webserver to limit what files and directories can be accessed and by whom. Doing this would make it such that all files and folders would only be accessible through the web application since the Apache web server would support the software with permission to access the files. The following example commands must be run on the Linux Server for all the directories. We use the images folder in the examples below to demonstrate the configuration.

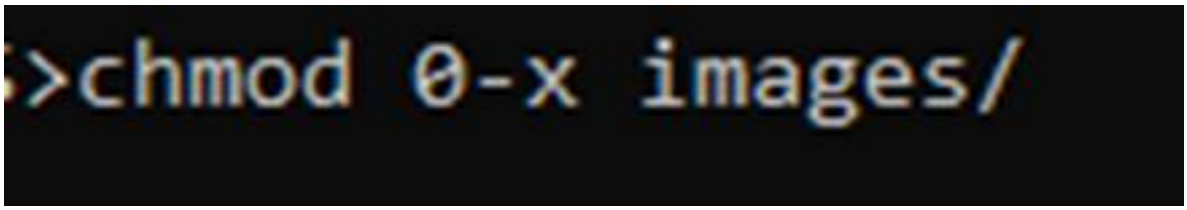


Figure 0.14: Removing execution permissions for other users' example

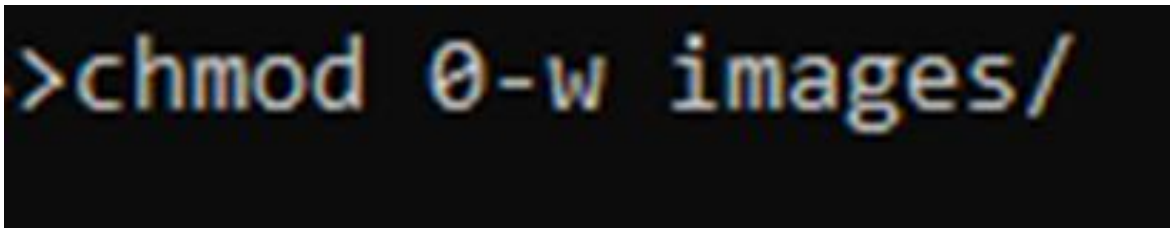


Figure 0.15: Removing editing permissions on the server

2. **Carefully planning the web content stored in the folders**

Secure information should not be placed in files inside the web directory because of directory traversal problems that would make them accessible. Using a database to store information would be more ideal and secure.

3. **Include index.php files**

A simple option that the web developer could do is to include index.php files in all of the web directories. The index.php files would have code redirecting the user to the main page, preventing direct access to the files through the browser. This is the solution I implemented with some aspects of the second point. The first solution is the responsibility of the website's hosting company.

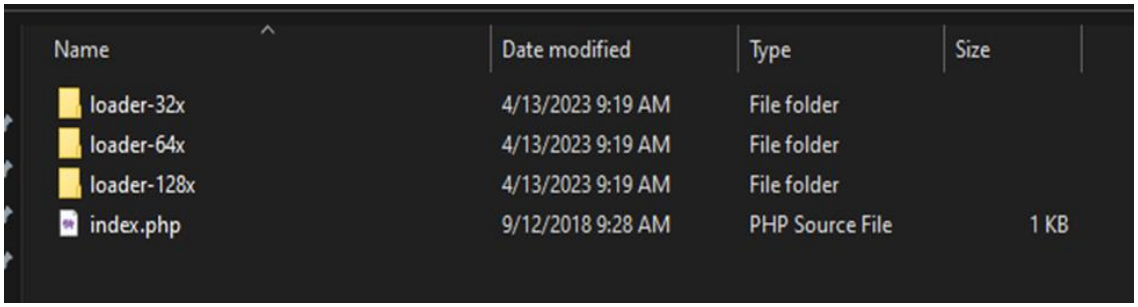


Figure 0.16: Example of index.php placed in the images folder

The content of the index file is shown below. This has code to redirect the hacker out of the folder.

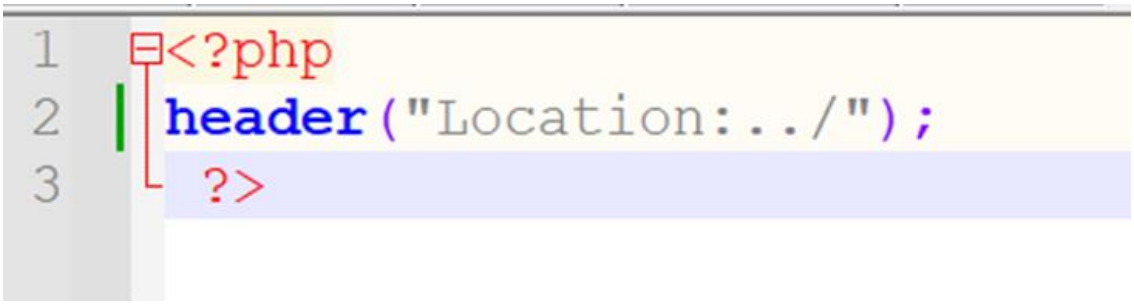


Figure 0.17: The content of the index file

3.7.5 Packet Sniffing

1. **Protecting the web server**

Packet sniffing traffic can be prevented by having the traffic encrypted. Encryption for web application traffic is done on the server side, not in the source code for PHP. The web administrator would have to implement HTTPS. This was done by installing mod_SSL and open_SSL in the case of a Linux web server. Once this was done, the web administrator could generate the SSL certificate using the "req new" command, supply the values for the certificate and sign it for at least 365 days. The httpd.conf can then be opened, and all virtualhosts sections can be uncommented with the certificated files linked. This should enable HTTPS connections on the server. This is the most recommended solution because it means all traffic from the server to any client that connects to the website will be encrypted, so packet sniffing, while still possible, will only capture cyphered text. It would then require the hacker to make a tremendous effort to be able to decipher the encrypted text.

2. Protecting the Individual

Unfortunately, because the solution on top requires a complete re-configuration of the Apache web server, it means that until that occurs, website users would be vulnerable to these attacks. The potential fix for individual users working with the website before the implementation of HTTPS would be to request that they first install Virtual Private Network (VPN) software on their client machines and run the software, then make the connection to a VPN server secondly and thirdly connecting to the web application. The VPN solution helps secure data as it is sent over a public network such as the Internet, which would secure the traffic for most of the path to the server. This is only a temporary fix, not a complete solution if the web server is not configured with HTTPS quickly.

3.8 Summary of Chapter 3

The methodology chapter presented the research design and methods used in this study to investigate web application attacks, specifically SQL injections, database enumerations, directory traversal, DirBuster, and password sniffing. The chapter began by describing the research approach combining theoretical and practical research. The chapter also provides a detailed explanation of the hardware and software tools used in the study, including the specifications and versions of each tool. Next, the research methodology used in each attack is described, including the steps taken to execute the attacks and the data collected from each attack. Finally, the methodology chapter sets the foundation for the results and analysis chapter, which will present and interpret the research's findings in the context of the research questions and objectives.

CHAPTER 4: RESULTS AND ANALYSIS

4.1 Overview

The Results and Analysis chapter presents the research findings involving testing and evaluating web application attacks such as SQL injections, database enumerations, directory traversal, DirBuster, and password sniffing. The findings are presented clearly and concisely, highlighting the vulnerabilities and weaknesses of the web application, as well as the potential impact of the attacks based on the following research objectives: (1) To identify and evaluate the various types of malware that commonly attack web applications, (2) To better analyse and assess current web application malware detection techniques to understand their strengths and weaknesses, and (3) To design and develop web applications that demonstrate vulnerabilities and countermeasures. The chapter also discusses the implications of the results and their significance in the context of web application security, providing recommendations for improving its security posture.

4.2 Literature review results

This section will present a detailed analysis of the literature review conducted on this topic based on research object 1: To identify and evaluate the various types of malware that commonly attack web applications.

4.2.1 SQL Injection Attacks

From the reviewed literature, SQL injection attacks are among the most common web application attacks that exploit vulnerabilities in the application's code. The attack involves injecting malicious SQL code into the application's input fields, which can allow an attacker to gain access to sensitive data or even take control of the application. However, the literature review reveals that various measures can be implemented to prevent SQL injection attacks. These include using prepared statements, input validation, and parameterised queries.

4.2.2 Cross-Site Scripting Attacks

According to the literature review, Cross-site scripting (XSS) attacks are another prevalent web application attack that involves injecting malicious code into a legitimate website. The code can then execute on the user's browser, allowing an attacker to steal

sensitive data or hijack user sessions. The literature review suggests several technical measures to prevent XSS attacks, including input validation, encoding user input, and implementing a Content Security Policy (CSP).

4.2.3 Clickjacking Attacks

Clickjacking attacks are a type of attack that involves tricking users into clicking on malicious links or buttons based on the reviewed literature. The literature review suggests that clickjacking attacks can be prevented by implementing measures such as X-Frame-Options headers and JavaScript-based solutions such as frame-busting code.

4.2.4 Session Hijacking Attacks

The reviewed literature shows that Session hijacking attacks are a type of attack where an attacker gains access to a legitimate user's session by stealing their session ID. The literature review suggests that implementing session timeouts, regenerating session IDs, and using secure cookies can help prevent session hijacking attacks.

The literature review reveals that several technical measures can be implemented to strengthen web application security. These measures include input validation, using prepared statements and parameterized queries to prevent SQL injection attacks, encoding user input and implementing CSP to prevent XSS attacks, implementing X-Frame-Options headers and frame-busting code to prevent clickjacking attacks, and implementing session timeouts, regenerating session IDs, and using secure cookies to prevent session hijacking attacks. By implementing these measures, web application developers can significantly reduce the risk of attacks and ensure the security of their users' sensitive data, thereby achieving research objective 1 of this research.

4.3 Web Application Attack Results

This section presents the results and analysis of various web application attacks, including SQL injection, database enumeration, directory traversal, and password sniffing conducted in chapter 3 of this research based on research objectives 2: To analyze better and assess current web application malware detection techniques to understand their strengths and weaknesses, and 3: To design and develop web applications that demonstrate vulnerabilities and countermeasures.

4.3.1 SQL Injections

The SQL injection technique, "Bypassing Login," was performed, as shown in Figure 0.5. The successful operation is shown below.

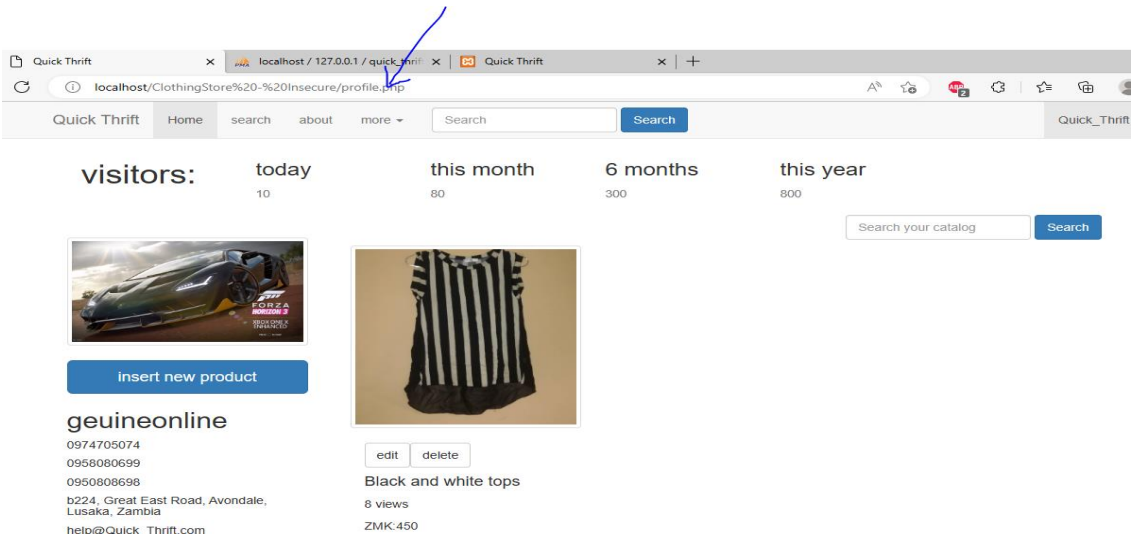


Figure 0.1: SQL Injections Attack Result

The Bypassing Login attack was executed by inserting a specific SQL command, which resulted in the web application authenticating the first user of the system without the need for valid login credentials.

The attack result in Figure 0.1 demonstrates the profile page being loaded despite no username or password being entered. This information is crucial for understanding the vulnerability of the web application and how attackers can exploit it.

The SQL injection attack, Bypassing Login," was further simplified by using a shorter SQL command that achieves the same outcome of logging in as the first user of the system. This approach is illustrated in the query shown in Figure 0.6.

If the hacker is aware of a specific victim's username or email address in what would be characterised as a targeted attack, the hacker can use the email address or username as part of the query so that the injection logs on to the specific account.

This result implies that web applications with poor security measures are highly vulnerable to SQL injection attacks. Using a shorter SQL command to bypass login, attackers can gain unauthorised access to sensitive information or even take control of the entire system; this highlights the need for proper security measures such as input validation and parameterised queries to prevent such attacks. Additionally, web

developers and system administrators must be aware of the potential security risks associated with web applications and take proactive measures to secure their systems.

4.3.2 Database Enumerations

The main idea behind this technique is to force the database server to generate errors that can reveal information about the database in the attack conducted.

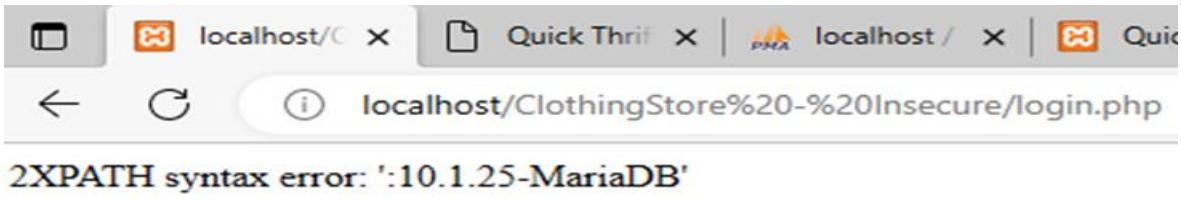


Figure 0.2: Database Version Fingerprinting Result

The query ran successfully because when executed, the web application generated an error that revealed at least one database from the results generated, as shown in Figure 0.2 and the database name “quick thrift” in Figure 0.3.

Figure 0.3: Database Table Name Fingerprinting Result

Given that we were able to retrieve the name of the database. We used the name discovered to figure out the tables within the database using more error messages that I forced with some SQL injections, as shown below. We started by writing the SQL injection, which extracts the value and gives the result. Do note that we had to insert the database name to pull out at least one table from it specifically, as shown in Figure 0.4.

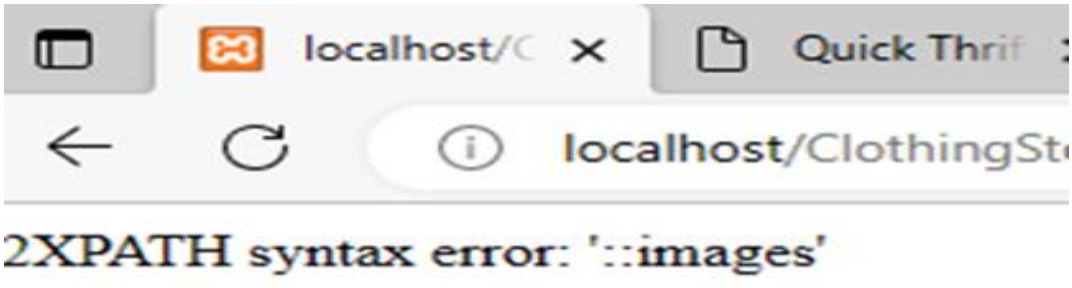


Figure 0.4: Traversing Quick Thrift Database Results

The SQL statement was used to perform the attack.

```
' AND extractvalue(rand(),concat(0x3a,(SELECT concat(0x3a,TABLE_NAME) FROM information_schema.TABLES WHERE table_schema="quick_thrift" LIMIT 0,1)))—
```

Listings 3: Traversing Quick Thrift Database Results Code Snippet

The researcher successfully accessed the first table of the database, namely "images." Subsequently, the researcher refined the query to locate tables without prior knowledge of their names by utilizing the LIKE command to search for tables with text. For instance, searching for "me" yielded the "members" table, while searching for "p" resulted in the "posts" table, and searching for "ne" returned the "newsletter" table. The principle behind this approach is that any query with text that is part of the search term would generate a corresponding result.

The results imply that the web application's database is vulnerable to SQL injection attacks. In this SQL statement,

```
' AND extractvalue(rand(),concat(0x3a,(SELECT concat(0x3a,TABLE_NAME) FROM information_schema.TABLES WHERE table_schema="quick_thrift" AND TABLE_NAME LIKE '%me%'LIMIT 0,1)))--
```

Listings 4: Database Members Search Code Snippet

We searched for “me” on the login page, which was able to return the result showing the members table as shown from the error generated after. The fact that the researcher could access and manipulate the tables using the LIKE command suggests that the database is not properly secured against unauthorized access.

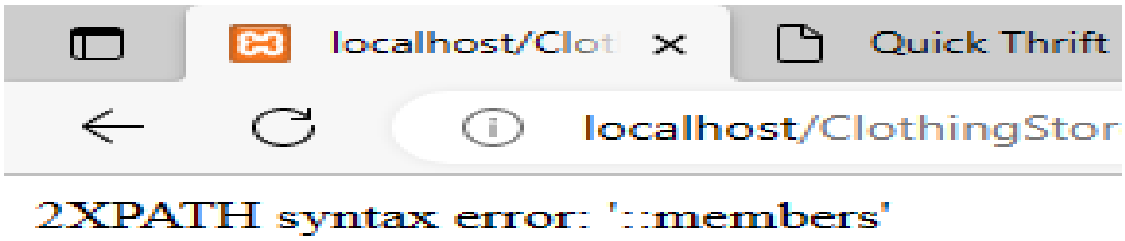


Figure 0.5: Enumerating the Tables Attack Code Snippet

This vulnerability could allow attackers to view, modify, or delete sensitive data within the database. The results highlight the importance of implementing effective security

measures to prevent SQL injection attacks, such as input validation, parameterized queries, and stored procedures.

Using the SQL statement below, we could recognise the column passwords from the results in Figure 0.6.

```
' AND extractvalue(rand(),concat(0x3a,(SELECT concat(0x3a,column_name) FROM information_schema.columns WHERE TABLE_NAME="members" AND column_name LIKE '%p%' LIMIT 0,1)))--
```

Listings 5: Enumerating the Tables Attack Code Snippet

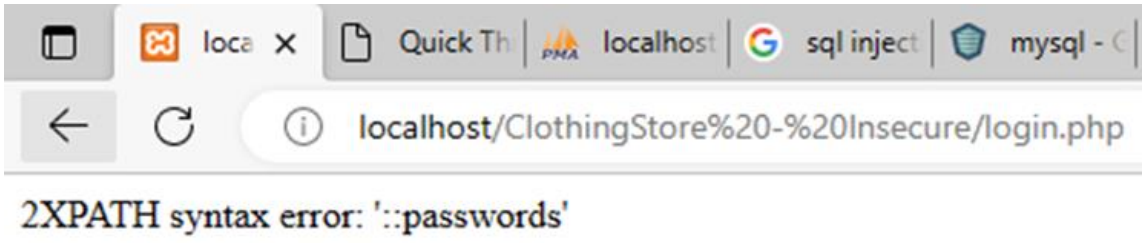


Figure 0.6: Enumerating the Column Names Attack

Finally, using the SQL statement below, the result showed that the username “temwani” and the password “temwani” were found after the query ran, as shown in Figure 0.7.

```
' AND extractvalue(rand(),concat(0x3a,(SELECT concat(password,0x3a,password) FROM members LIMIT 0,1)))--
```

Listings 6:Password Attack Code Snippet

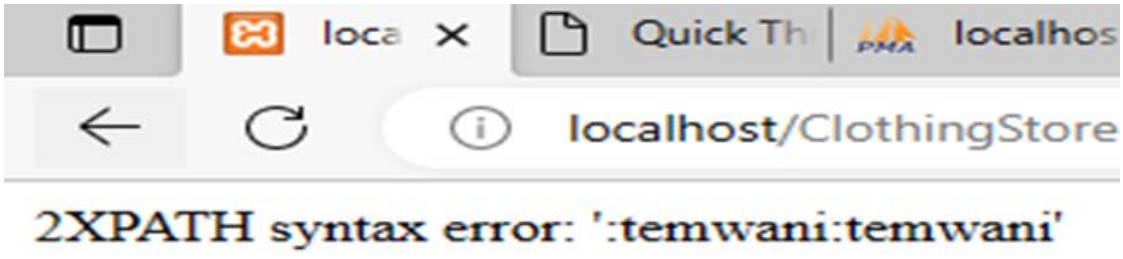


Figure 0.7: Enumerating the Tables Password Attack

The results imply that the database used by the web application is vulnerable to database version fingerprinting, and the error generated during the execution of the query could provide valuable information to a potential attacker. The attacker could use this information to exploit any known vulnerabilities in the database version and gain unauthorized access to sensitive data stored in the database. It also highlights the

importance of implementing security measures such as regular updates and patches to ensure that known vulnerabilities are addressed promptly to prevent such attacks. Additionally, it underscores the need for secure coding practices to prevent the introduction of SQL injection vulnerabilities in the first place.

3..4.3 Directory Transversal Attack

The search results showed all the folders on the system for the webserver, as shown in the screenshot below. Figure 0.8. The results here use the list view, which had all the files on the website, so the screenshot only shows the first files that could be listed, but scrolling down would reveal even more files and folders. This concerns web application development because carelessly placed files in the web directory could be easily accessible to outsiders who might need to be logged on to access the files.

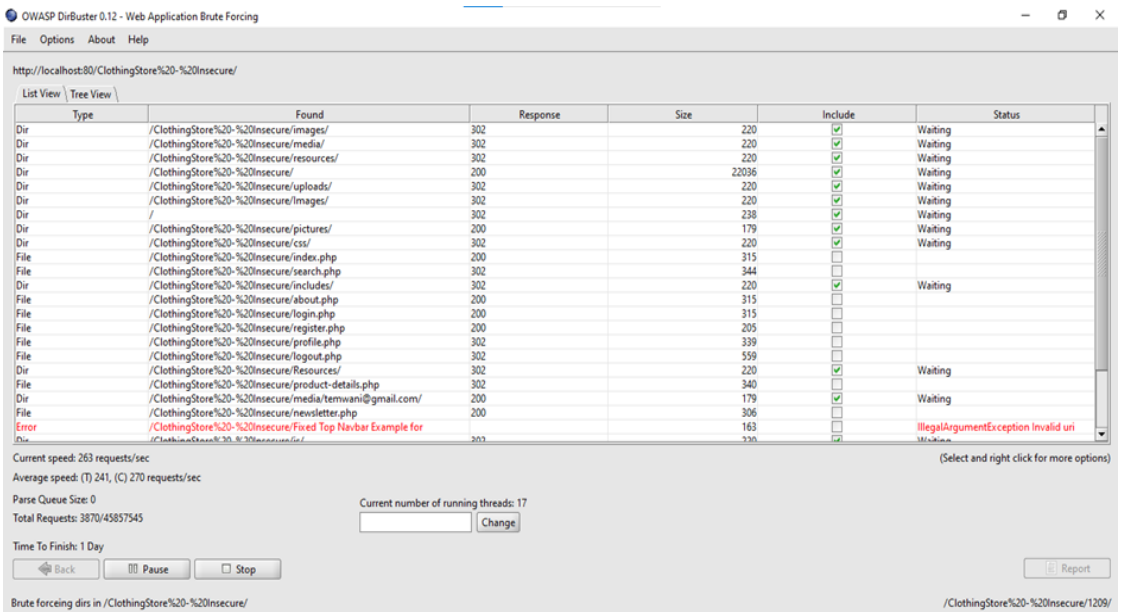


Figure 0.8: Dirbuster Results in List View

We transitioned from a list view to a tree view to improve the ease of navigation while using a user interface like that of Windows. This change was reflected in the screenshot, highlighting the directories for images and uploads, including CSS, JavaScript, and individual PHP files, although not all of them could be accessed. However, the knowledge obtained through this exercise could be used to access relevant resources via a web browser by specifying the corresponding URLs. Then, we used this information to target specific resources of interest to potential attackers, as shown in Figure 0.9.

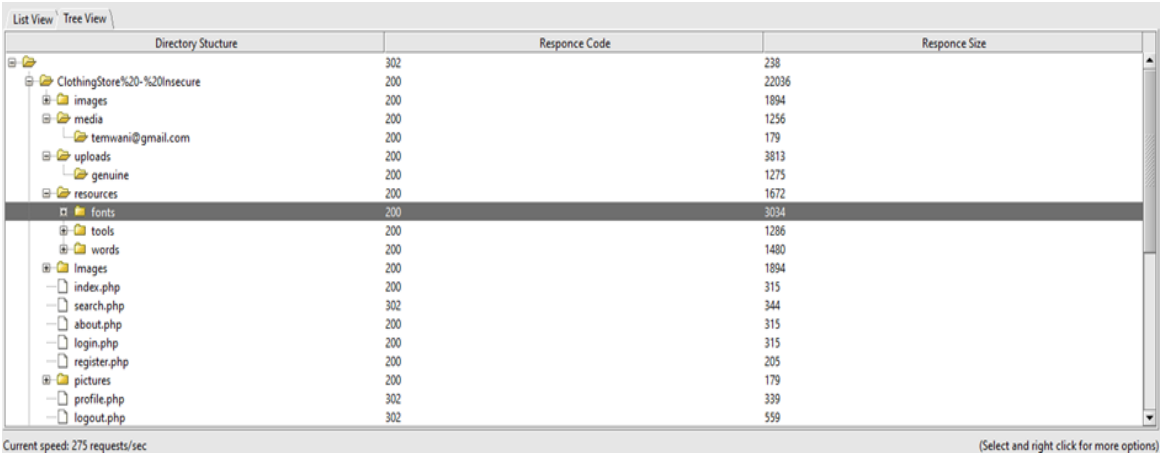


Figure 0.9: Dirbuster Results in Tree View

After that, we open a web browser, as shown in Figure 4.9, to access the files over the network. The screenshot below shows how the results were then used to open the folder for pictures. Do note that these pictures can then be opened or downloaded. The same is true for all other files, not PHP or HTML; the files would open if the browser recognises them or trigger a download if the browser does not recognise them.

Figure 0.10: File access over the Network

Accessing web server files through a directory traversal attack poses various security concerns for web application security. First, it enables hackers to gain knowledge of the web server's files, allowing them to identify sensitive files such as those used by system

administrators that are not accessible to regular users. Additionally, the attacker can utilize the directory traversal attack to enumerate system users. Many systems create folders to store individual user files, and through this method, the hacker can discern the usernames and email addresses of the users. In the given illustration, the media directory is examined to determine that a user with the email address temwani@gmail.com exists and that their media files are listed with the user's email address. These vulnerabilities can allow attackers to gain unauthorised access to sensitive data or user accounts, potentially causing severe consequences.

3.4.4 Password Sniffing Attack

We started the test by running the program and selecting the wireless interface since this is what we used for the internet connection, as shown in Figure 0.11.

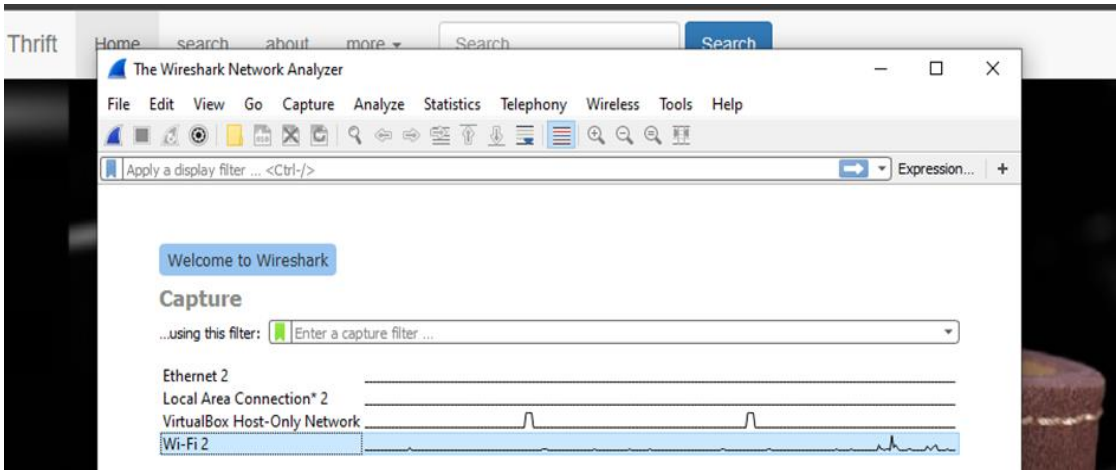


Figure 0.11: Password Sniffing Wireless Attack

The website was running on a server within the same network, so the URL has a local IP address without any domain name. We then clicked the blue button to begin capturing as we accessed the web application.

The tool captured traffic on multiple layers, but we were only interested in HTTP traffic, particularly for the website. Figure 0.12 shows the traffic being captured.

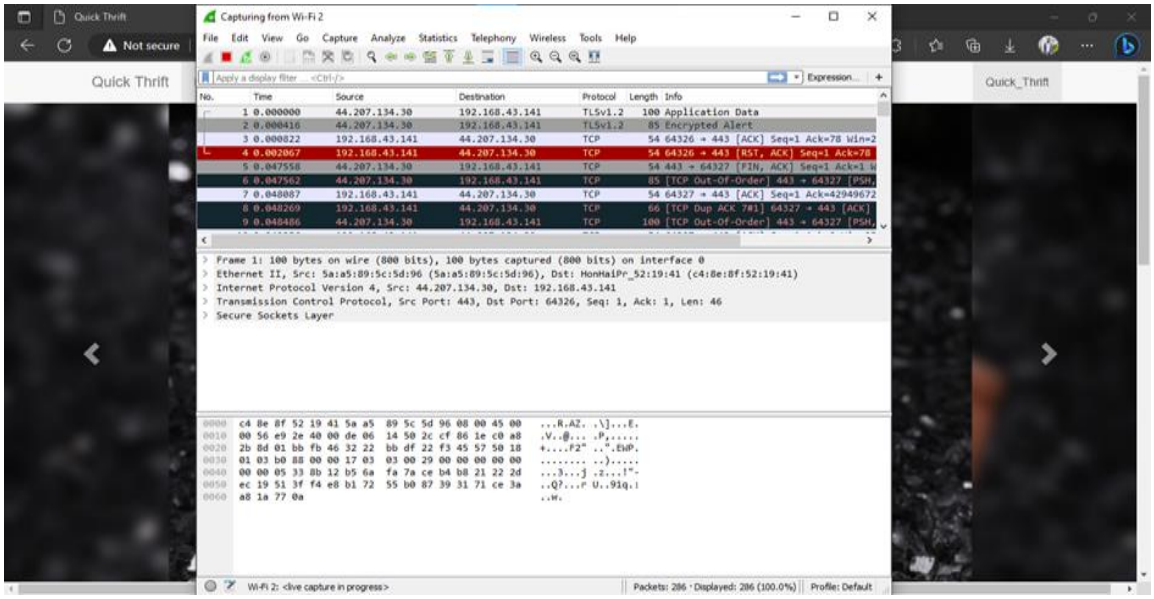


Figure 0.12: Captured Traffic on Multiple Layers

While the traffic was being captured, we went to the log-on page, allowing us to authenticate and log on to the system, as shown in Figure 0.13.

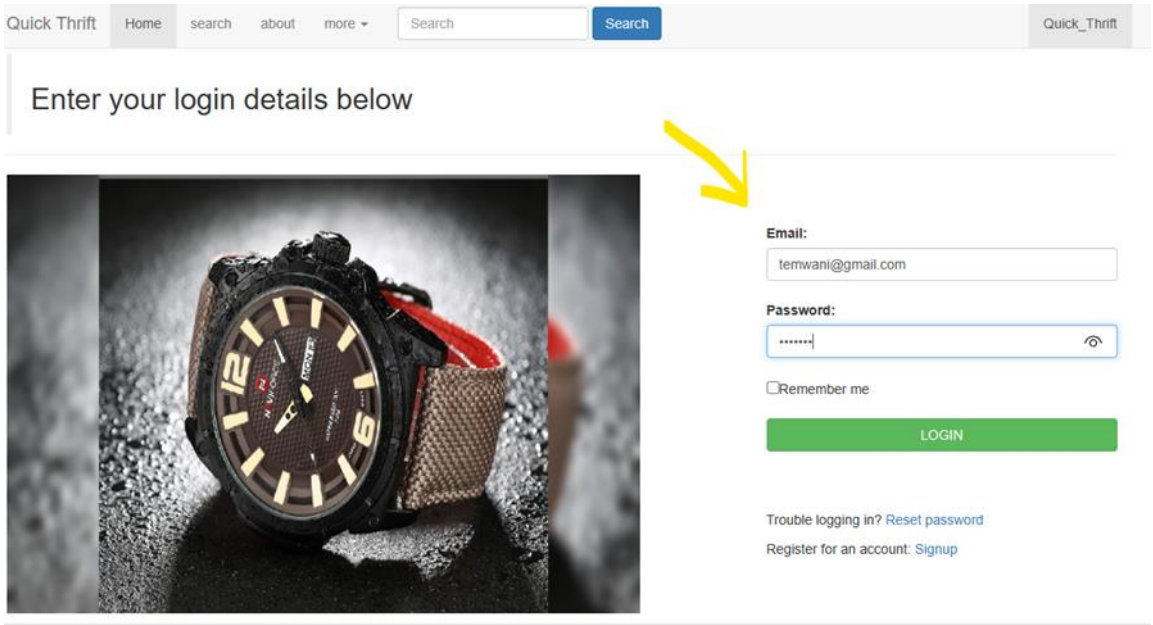


Figure 0.13: Password Sniffing Attack Login

Once authenticated, we could look at the profile page and search for a purple shirt on the site. The screenshot below shows the authenticated session running and the search type in the search box. At this point, we stopped and went to see whether the tool could capture anything, as shown in Figure 0.14.

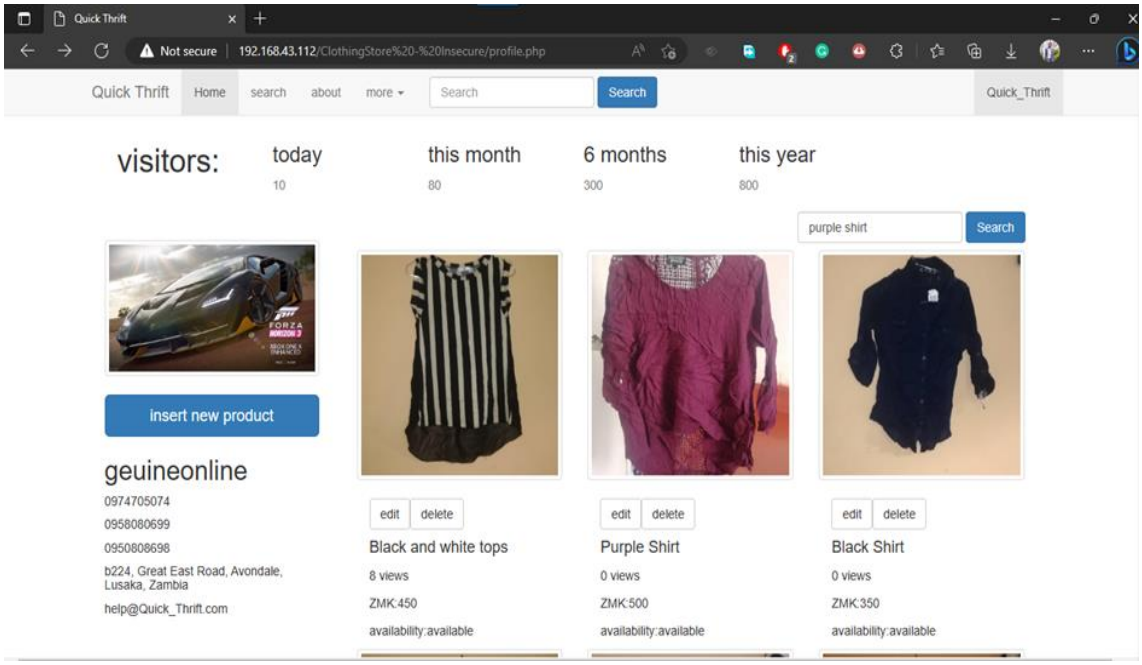


Figure 0.14: Password Sniffing Attack Logged in

3.4.5 Packet Sniffing attacks

Packet Sniffing attacks cause a few problems for the web application's security, as listed below.

We know what files the webserver contains; we know what interests them. For example, files used by the administrator that are not available to normal users can be seen through this process, as shown in Figure 0.15, Figure 0.16, and Figure 0.17.

We enumerated the users of the system. Most systems create folders to keep files for individual users. For example, in the figure below, we can determine by looking at the media directory that there is a user named temwani@gmail.com, and their media files are listed with the user's email address.

Packet sniffing attacks pose significant security risks to web applications, allowing attackers to access sensitive files and enumerate system users. Implementing measures such as encryption, secure protocols (e.g., SSL/TLS), and network segmentation can be highly useful for system administrators in mitigating packet sniffing threats. These countermeasures enhance the confidentiality and integrity of data transmitted over the network, preventing unauthorized access and protecting user privacy.

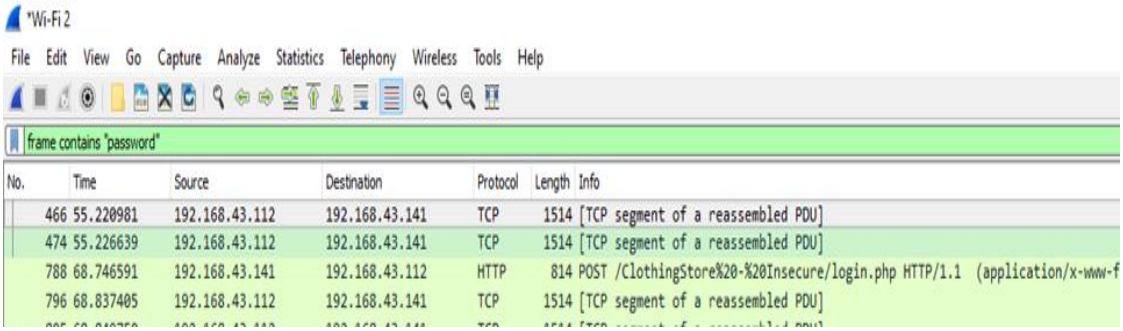


Figure 0.15: Result of the traffic from the "login.php A

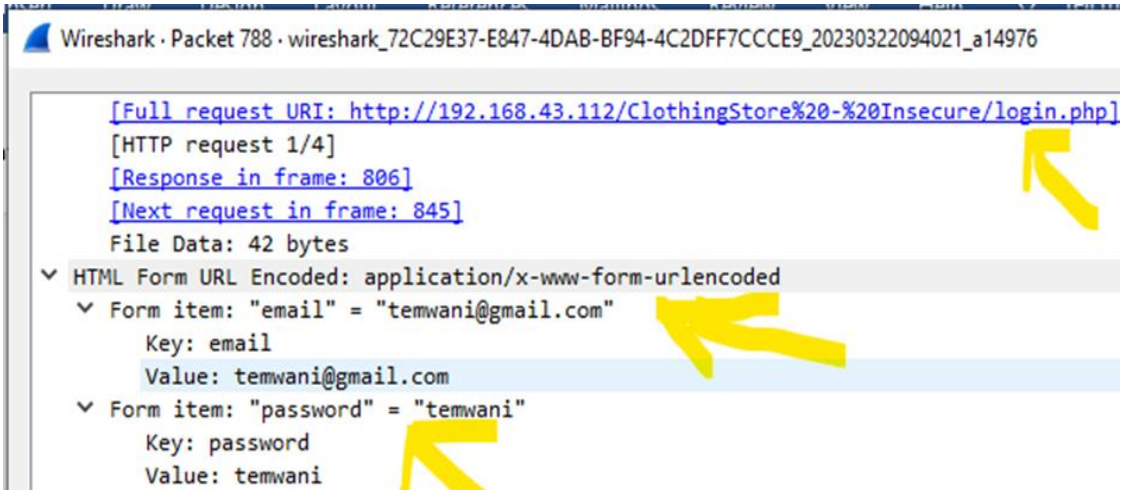


Figure 0.16: Result of the traffic from the "login.php B

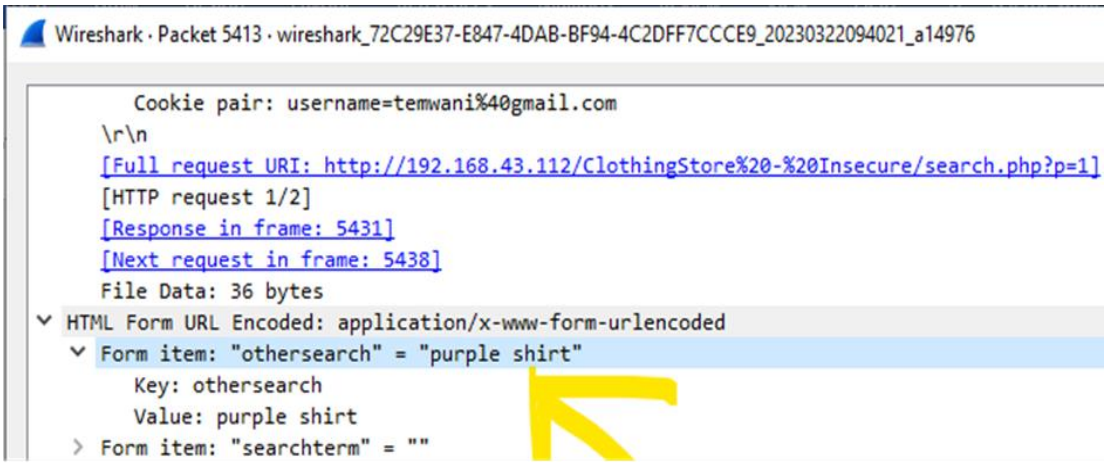


Figure 0.17: Result of the traffic from the "login.php C

The process carried out in the study revealed that the web application transmitted data in plain text, and as a result, any user's data transmitted through the web application was

visible to other network users; this implies that the web application lacks real confidentiality, and any data, whether a message, a click on a button or link, a username or password supplied on a log-in form, URLs viewed by the user, and information the server responds with can be easily viewed. Such a situation could lead to a complete compromise on confidentiality, making it possible for a third party to perform a Man in The Middle (MITM) attack where traffic would be captured, modified slightly, and sent back to the network without the user's knowledge.

The reviewed literature indicates that the web application's transmission of data in plain text poses a significant risk to confidentiality. The lack of encryption allows potential attackers to intercept and manipulate sensitive information, compromising user privacy. Implementing secure transport protocols, such as SSL/TLS, would be highly beneficial for system administrators to ensure the confidentiality of transmitted data and prevent Man-in-The-Middle (MITM) attacks.

3.5 Summary of Chapter 4

The results and analysis review chapter of "Strengthening Web Application Security through Technical Measures" presents the findings of the researcher's efforts to identify vulnerabilities in web applications using various attack techniques. The researcher successfully performed attacks such as SQL injection, directory enumeration, and packet sniffing, revealing significant security flaws in the targeted web applications. These flaws include weak password storage, lack of encryption in data transmission, and insufficient access controls. The chapter highlights the implications of these vulnerabilities, which include the risk of confidential information exposure, unauthorized access, and potential malicious attacks by third parties. The chapter concludes by emphasising the importance of implementing technical security measures such as encryption, input validation, and access controls to enhance web application security.

CHAPTER 5: CONCLUSION, RECOMMENDATIONS, AND FUTURE WORKS

5.1 Overview

The conclusion of the research underscores the significance of technical measures in strengthening web application security. The research highlights the numerous vulnerabilities and weaknesses in web applications and shows that hackers can exploit these to compromise confidentiality, integrity, and availability. To enhance the security of web applications, the research recommends implementing several technical measures, including firewalls, intrusion detection and prevention systems, encryption, secure coding practices, access controls, and vulnerability scanning and testing. Additionally, the research recommends that web application developers and administrators remain vigilant and proactive in identifying and addressing potential security risks. Finally, the research proposes several future research directions, including exploring emerging technologies and tools for web application security and developing best practices for securing web applications.

5.2 Conclusion

This research aimed to identify and assess prevalent web application malware, evaluate current malware detection approaches, and develop secure web applications with a clear understanding of vulnerabilities and associated countermeasures. Through a detailed methodology, the research identified common web application attacks, including SQL injection, directory traversal, password sniffing, and packet sniffing. This achievement addressed the first research question: "What are the different types of malware that commonly attack web applications, and how do they operate?"

The research findings emphasized the importance of comprehensive security measures such as firewalls, intrusion detection systems, and secure coding practices in enhancing web application security. The effectiveness of these measures in reducing vulnerabilities was evaluated, addressing the second research question: "What are the strengths and weaknesses of current web application malware detection techniques, and how effective are they in detecting and mitigating malware attacks?"

Furthermore, the research developed a test site with vulnerabilities and a non-vulnerable site, demonstrating how these vulnerabilities could be exploited and the countermeasures

to mitigate them. This addressed the third research question: "How can vulnerabilities in web applications be identified, and what technical measures can be implemented to strengthen web application security and prevent malware attacks?"

The research findings proved the importance of having comprehensive security measures to reduce web application vulnerabilities. It was determined that input validation, client-side and server-side validation, and input sanitization are effective defences against SQL injection attacks. The research also addressed directory traversal attacks by emphasizing the necessity of setting the correct permissions on the server, carefully organizing online content storage, and including index.php files in directories to prevent direct access. In addition, the research recommended securing the web server with HTTPS encryption to counter packet sniffing vulnerabilities.

By implementing a complete strategy that incorporates secure coding methods, strong validation procedures, proper server configuration, and encryption protocols, web application developers may defend their apps against malware assaults and increase the security of their applications; this fulfills the research aim and objectives and offers a comprehensive understanding of web application security, contributing significantly to the field. The findings of this study lay the groundwork for future research and development in this area.:

5.3 Recommendations

Based on the findings and analysis of this research, the following recommendations were made:

1. Web application developers should take security seriously and prioritize security measures during development.
2. Web application owners and administrators should regularly update their web applications to the latest version and monitor them for any vulnerabilities or signs of a breach.
3. Organizations should have a security policy that clearly outlines the security measures that should be implemented in web applications.
4. It is recommended that organizations should conduct regular security audits and penetration testing to identify vulnerabilities in their web applications and take corrective action.

5. Regular training and awareness programs should be conducted for developers and administrators to ensure they are aware of the latest threats and mitigation techniques.
6. The use of web application firewalls (WAFs) should be considered an effective measure to detect and prevent web application attacks.
7. The use of anti-malware software on web servers and end-user devices is recommended to detect and prevent malware attacks.
8. It is important to keep up-to-date with the latest developments and trends in web application security to stay ahead of new threats and vulnerabilities.
9. Finally, web application owners should have a disaster recovery plan to respond quickly and efficiently to security breaches.

5.4 Future Works

Based on the findings of this research, several areas could be explored in future research to improve further the detection and mitigation of malware attacks in web applications.

Firstly, more research could be done to explore the effectiveness of machine learning and artificial intelligence techniques in detecting and mitigating malware attacks. These techniques could improve the accuracy and speed of malware detection and response.

Secondly, further investigation could be done into the effectiveness of different countermeasures for specific malware attacks. This would help to identify the most effective measures for different types of attacks and provide more targeted guidance for improving web application security.

Thirdly, additional research could be conducted on developing new, more advanced malware attacks and how to detect and mitigate them. Finally, as malware attacks evolve and become more sophisticated, staying updated with the latest threats and techniques for preventing and responding to them is important.

Overall, there is much scope for future research in web application security and malware detection and mitigation. By continuing to explore new approaches and techniques, it should be possible to develop more effective measures for preventing and responding to malware attacks and ultimately improve the security of web applications for users and organisations.

5.5 Summary

This study enabled us to recognize the importance of implementing tight security measures on the web application to combat vulnerability issues. Furthermore, the a need to ensure that the system is updated with the latest security technologies.

REFERENCES

[1] Veracode, “Web Application Security Checklist: Best Practices for Secure Development,” Mar. 2021, [Online]. Available: https://www.veracode.com/security/web-application-security/checklist

[2] HealthITSecurity, “Ransomware Attacks Targeting Healthcare Providers Show No Signs of Slowing Down,” Sep. 2021, [Online]. Available: https://healthitsecurity.com/news/ransomware-attacks-targeting-healthcare-providers-show-no-signs-of-slowing-down

[3] F. Times, “Cybersecurity firm FireEye hacked by a state-sponsored group,” Dec. 2020, [Online]. Available: https://www.ft.com/content/ec2f7b9e-8408-401d-af6f-3a029c42d00e

[4] G. V. Research, “Web Application Firewall (WAF) Market Size, Share & Trends Analysis Report By Solution (Hardware, Software), By Service, By Deployment, By End Use, By Region, And Segment Forecasts, 2021 - 2028,” Mar. 2021, [Online]. Available: https://www.grandviewresearch.com/industry-analysis/web-application-firewall-waf-market

[5] WhiteSource, “Securing Web Applications: A Step-by-Step Guide,” Sep. 2021, [Online]. Available: https://www.whitesourcesoftware.com/resources/blog/securing-web-applications/

[6] Synopsys, “10 Web Application Security Best Practices for Developers,” Jun. 2021, [Online]. Available: https://www.synopsys.com/blogs/software-security/web-application-security-best-practices/

[7] Imperva, “The State of Web Application Security in 2021,” Feb. 2021, [Online]. Available: https://www.imperva.com/blog/state-of-web-application-security-report-2021/

[8] C. S. O. Online, “Why web app security remains a critical issue and how to address it,” Aug. 2021, [Online]. Available: https://www.csoonline.com/article/3623088/why-web-app-security-remains-a-critical-issue-and-how-to-address-it.html

[9] P. Technologies, “Web Application Security Statistics Report 2021,” Mar. 2021, [Online]. Available: https://www.ptsecurity.com/upload/corporate/ww-en/analytics/Web-Application-Security-Statistics-Report-2021-eng.pdf

[10] O. Foundation, “OWASP Top Ten Web Application Security Risks,” Nov. 2020, [Online]. Available: https://owasp.org/Top10/

[11] D. Bisson, “New SQL Injection Attack Technique Emerges,” *Tripwire*, Jan. 2022.

[12] J. Davis, “SQL Injection: A Beginner’s Guide to Understanding and Prevention,” *HealthITSecurity*, Mar. 2022.

[13] T. Keary, “The Dangers of SQL Injection and How to Prevent Them,” *CSO Online*, Feb. 2022.

[14] M. Samuels, “SQL Injection Attacks on the Rise as Hackers Exploit Pandemic,” *ZDNet*, Mar. 2022.

[15] M. Burnett, “Preventing SQL Injection Attacks with Parameterized Queries,” *Security Boulevard*, Jan. 2022.

[16] A. Kumar, “DirBuster – Brute Force Directories and Files Hidden in Web Application,” *The Hack Today*, Oct. 2021.

[17] D. Balaban, “Using DirBuster to Find Hidden Directories on Web Servers,” *TechNadu*, Nov. 2021.

[18] A. Crenshaw, “DirBuster: A Tool for Finding Hidden Web Objects,” in *Black Hat Asia*, 2022.

[19] K. Makan, “Burp Suite Tutorial: Using DirBuster to Find Hidden Web Content,” *Infosec Institute*, Jun. 2021.

[20] S. Rana, “DirBuster Tutorial: Finding Hidden Files and Directories,” *TechViral*, Mar. 2021.

[21] Varonis, “Password Sniffing: How It Works and How to Defend Against It.” Nov. 2021.

[22] McAfee, “Protect Yourself from Password Sniffing.” Nov. 2021.

[23] TechBeacon, “Password sniffing attack: What it is and how to prevent it.” Jan. 2022.

[24] NortonLifeLock, “Password Sniffing: How Hackers Steal Your Credentials.” Jan. 2022.

[25] Avast, “What is Password Sniffing and How Can You Prevent It?” Feb. 2022.

[26] SolarWinds MSP, “Packet Sniffing – A Threat to Your Network Security.” Jul. 2021.

[27] TechBeacon, “Packet Sniffing: What It Is and How to Protect Yourself.” Mar. 2022.

[28] McAfee, “How to Detect and Prevent Packet Sniffing.” Feb. 2022.

[29] Avast, “Packet Sniffing: What it is and what you can do about it.” Jun. 2021.

[30] Norton LifeLock, “What Is Packet Sniffing and How Can You Avoid It?” Oct. 2021.

[31] Cisco, “Next-Generation Firewalls: An Overview,” 2022.

[32] TechTarget, “How to Test Web Application Security with Penetration Testing,” Aug. 2021, [Online]. Available: https://searchsecurity.techtarget.com/feature/How-to-test-web-application-security-with-penetration-testing

[33] D. Reading, “The Evolution of Web Application Security: How to Keep Up with the Latest Threats,” Feb. 2021, [Online]. Available: https://www.darkreading.com/application-security/the-evolution-of-web-application-security-how-to-keep-up-with-the-latest-threats/a/d-id/1340266

[34] CNN, “Biden signs executive order to improve US cybersecurity after Colonial Pipeline attack,” May 2021, [Online]. Available: https://www.cnn.com/2021/05/12/politics/biden-executive-order-cybersecurity/index.html

[35] S. Institute, “Firewall Rules Best Practices.” 2021.

[36] MarketsandMarkets, “Next-Generation Firewall Market to Reach $6.71 Billion by 2026.” 2021.

[37] C. S. O. Online, “Firewalls and the Zero Trust Model,” 2022.

[38] PCMag, “How to Choose a Firewall Solution,” 2021.

[39] S. Review, “The Top Firewall Providers of 2021.” 2021.

[40] D. Reading, “Firewalls vs Intrusion Detection Systems (IDS): What is the Difference?” 2022.

[41] TechTarget, “Firewall Management in the Cloud Era,” 2021.

[42] X. Chen and Y. Wang, “A Novel Hybrid Intrusion Detection System Based on Improved Particle Swarm Optimization and Extreme Learning Machine,” *Appl Soft Comput*, vol. 87, 2020.

[43] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: techniques, datasets and challenges,” *Cybersecurity*, vol. 2, no. 1, 2019, doi: 10.1186/s42400-019-0038-7.

[44] T. Albarqouni and others, “Evaluating the Performance of Intrusion Detection Systems in Industrial Control Systems Environments,” *IEEE Trans Industr Inform*, vol. 17, no. 6, pp. 4148–4158, 2021.

[45] A. Singh and A. Singh, “Intrusion Detection System using Random Forest Algorithm with Improved Feature Selection Technique,” *International Journal of Advanced Science and Technology*, vol. 30, no. 7s, 2021.

[46] J. P. Adeshina and others, “An Overview of Machine Learning Techniques for Intrusion Detection Systems,” *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 3, 2022.

[47] R. Thangaraj and S. Santhosh Babu, “A Deep Learning Approach for Intrusion Detection System in Cloud-Based Networks,” *J Ambient Intell Humaniz Comput*, 2022.

[48] C. Preimesberger, “How to Choose the Right Antivirus Software for Your Business,” *eWeek*, Dec. 2022, [Online]. Available: https://www.eweek.com/security/how-to-choose-the-right-antivirus-software-for-your-business

[49] J. E. Dunn, “Antivirus vs. Anti-Malware: What is the Difference?,” *Sophos*, Jan. 2023, [Online]. Available: https://www.sophos.com/en-us/medialibrary/PDFs/technical%20papers/sophos-antivirus-vs-anti-malware-wpna.pdf

[50] E. Johnson, “The Future of Antivirus: AI and Machine Learning,” *Security Boulevard*, Feb. 2023, [Online]. Available:

https://securityboulevard.com/2023/02/the-future-of-antivirus-ai-and-machine-learning/

[51] N. J. Rubenking, “The Best Antivirus Protection for 2023,” *PCMag*, Apr. 2023, [Online]. Available: https://www.pcmag.com/picks/best-antivirus-protection

[52] Gartner, “The Evolving Role of Firewalls in Network Security.” 2023.

[53] A. Spadafora, “Antivirus software: What it is, and why you need it,” *TechRadar*, Mar. 2023, [Online]. Available: https://www.techradar.com/news/antivirus-software-what-it-is-and-why-you-need-it

[54] P. A. Networks, “Firewall Best Practices for Securing Modern Networks,” 2021.

[55] T. Guardian, “Ransomware attacks on the rise – how can businesses protect themselves?,” Apr. 2021, [Online]. Available: https://www.theguardian.com/technology/2021/apr/12/ransomware-attacks-on-the-rise-how-can-businesses-protect-themselves

[56] M. & Company, “The rise of deepfake technology: Implications for cybersecurity,” Aug. 2021, [Online]. Available: https://www.mckinsey.com/business-functions/risk/our-insights/the-rise-of-deepfake-technology-implications-for-cybersecurity

[57] B. B. C. News, “Why the SolarWinds hack is unlike any other cyber attack,” Dec. 2020, [Online]. Available: https://www.bbc.com/news/technology-55326276

[58] Reuters, “FBI warns ransomware attacks becoming more targeted, sophisticated,” Sep. 2021, [Online]. Available: https://www.reuters.com/technology/fbi-warns-ransomware-attacks-becoming-more-targeted-sophisticated-2021-09-22/

[59] D. Reading, “Data Breaches Exposed 36 Billion Records in First Half of 2021,” Jul. 2021, [Online]. Available: https://www.darkreading.com/threat-intelligence/data-breaches-exposed-36-billion-records-in-first-half-of-2021/d/d-id/1341414

[60] Symantec, “Web Application Security: What You Need to Know.” 2021.

[61] Incapsula, “7 Essential Tips for Securing Your Web Applications.” 2021.

[62] DZone, “Secure Your Web Applications with These 6 Best Practices.” 2021.

[63] McAfee, “Web Application Security: A Beginner’s Guide.” 2021.

[64] Imperva, “Top Web Application Security Risks in 2021.” 2021.

[65] Netsparker, “The Ultimate Guide to Web Application Security.” 2021.

[66] D. Yang, J. Kim, S. Chung, and H. Park, “Modernizing Content Security Policy for Preventing Cross-Site Scripting Attacks,” *ACM Trans Internet Technol*, vol. 20, no. 3, pp. 1–24, 2020.

[67] M. Heiderich and G. Heyes, “XSS Without HTML: Client-Side Template Injection with AngularJS,” in *Black Hat Europe 2016*, 2016.

[68] R. Malhotra and A. K. Verma, “Cross-Site Scripting (XSS) Attack Techniques and Protection Mechanisms: A Review,” *International Journal of Computer Science and Information Security*, vol. 11, no. 9, pp. 14–23, 2013.

[69] A. Tyagi and A. Rastogi, “Understanding Cross-Site Scripting (XSS) Vulnerabilities,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 687–707, 2018.

[70] M. Katanga and R. Mbwikalambo, “Enhancing Web Application Security through Technical Countermeasures: An Empirical Study in the Zambian Context,” *Journal of Emerging Trends in Computing and Information Sciences*, vol. 13, no. 2, pp. 89–97, 2022.

[71] K. Alghathbar and A. Alruban, “A Comprehensive Survey on Web Application Security,” *Journal of King Saud University-Computer and Information Sciences*, vol. 33, no. 1, pp. 38–46, 2021.

[72] C. Lumbwe and J. Lungu, “Evaluation of Web Application Security Measures: A Case Study of Zambia,” in *Proceedings of the 2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, 2018, pp. 1–5.

[73] S. Singh and J. Shekhar, “Web Application Security: A Review of Current Research,” *Int J Comput Appl*, vol. 179, no. 28, pp. 1–5, 2020.

[74] Akamai Technologies, *Web Application Security: How to Avoid Authentication Attacks*. Akamai Technologies Inc., 2022.

[75] H. Snyder, “Literature review as a research methodology: An overview and guidelines,” *J Bus Res*, vol. 104, 2019, doi: 10.1016/j.jbusres.2019.07.039.

[76] M. Patel and N. Patel, “Exploring Research Methodology,” *International Journal of Research and Review*, vol. 6, no. 3, 2019.

[77] K. B. M. Noor, “Case study: A strategic research methodology,” *American Journal of Applied Sciences*, vol. 5, no. 11. 2008. doi: 10.3844/ajassp.2008.1602.1604.

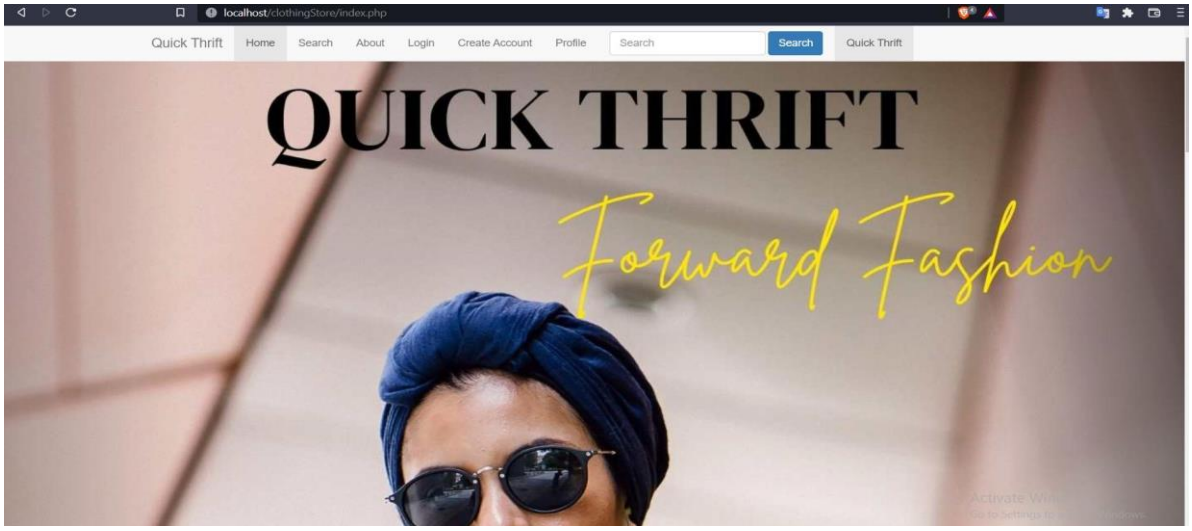
[78] T. Moher and G. M. Schneider, “Methodology and experimental research in software engineering,” *Int J Man Mach Stud*, vol. 16, no. 1, 1982, doi: 10.1016/S0020-7373(82)80072-2.

[79] M. Ngosa and P. Nyirenda, “A Comparative Analysis of Web Application Security Measures,” in *Proceedings of the 2019 International Conference on Information and Communication Technologies (ICICT)*, 2019, pp. 1–6.

APPENDICES

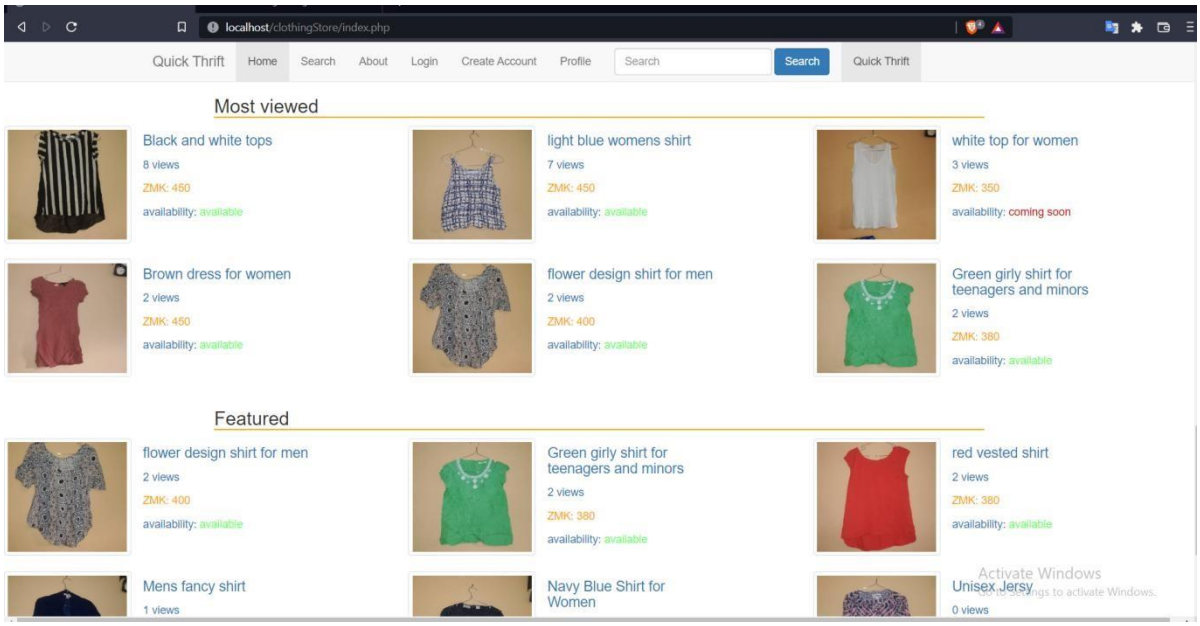
Appendix 1: Website Screenshots

The following screenshots show the web application and its usage.



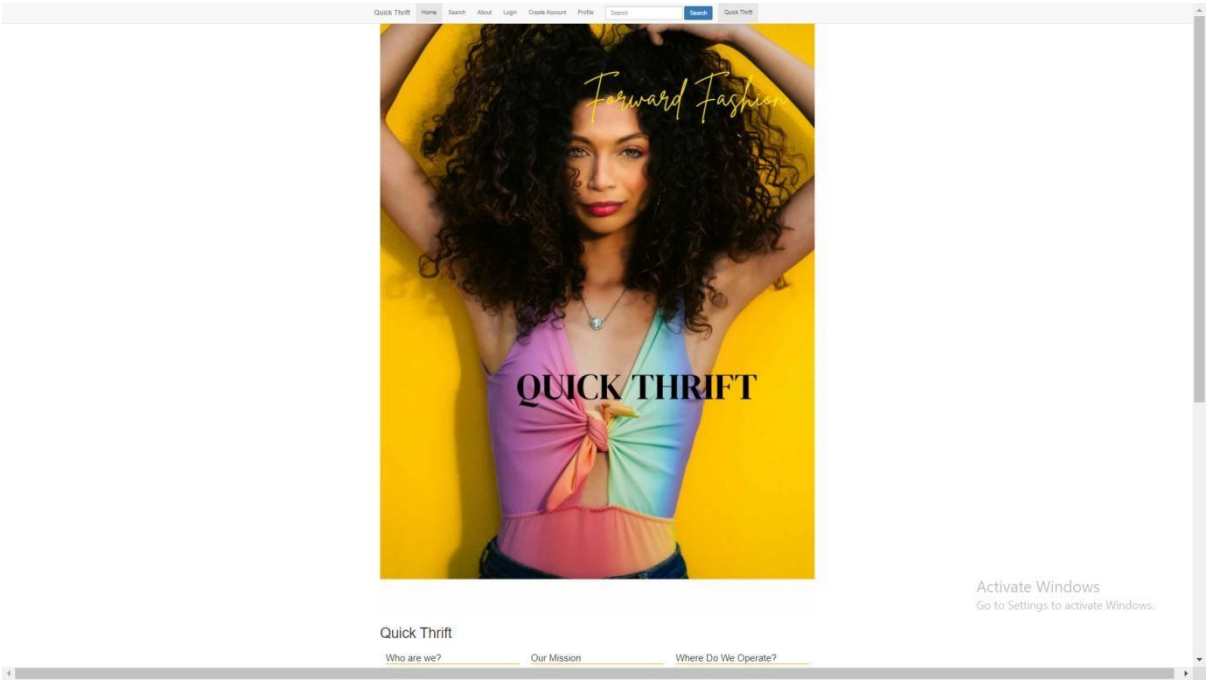
Homepage

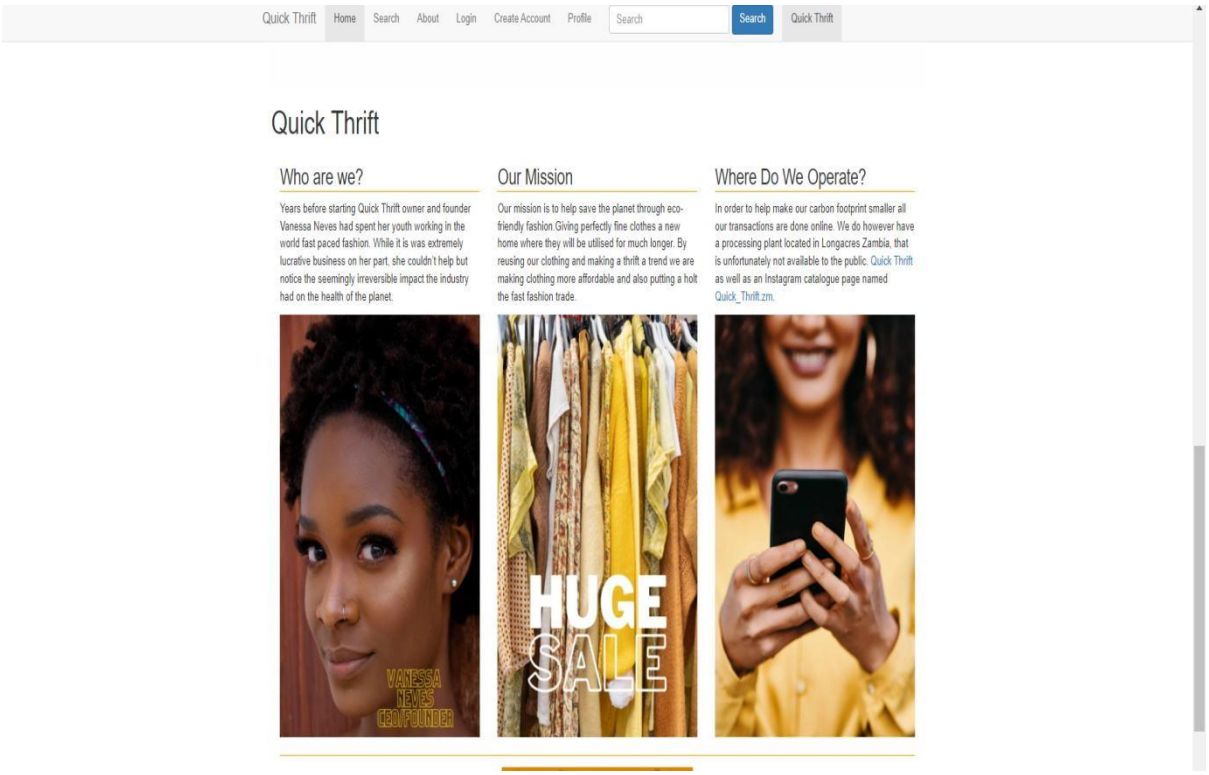
This shows scrolling images about the business and lists items on sale, from most viewed to featured listings.



About Us Page

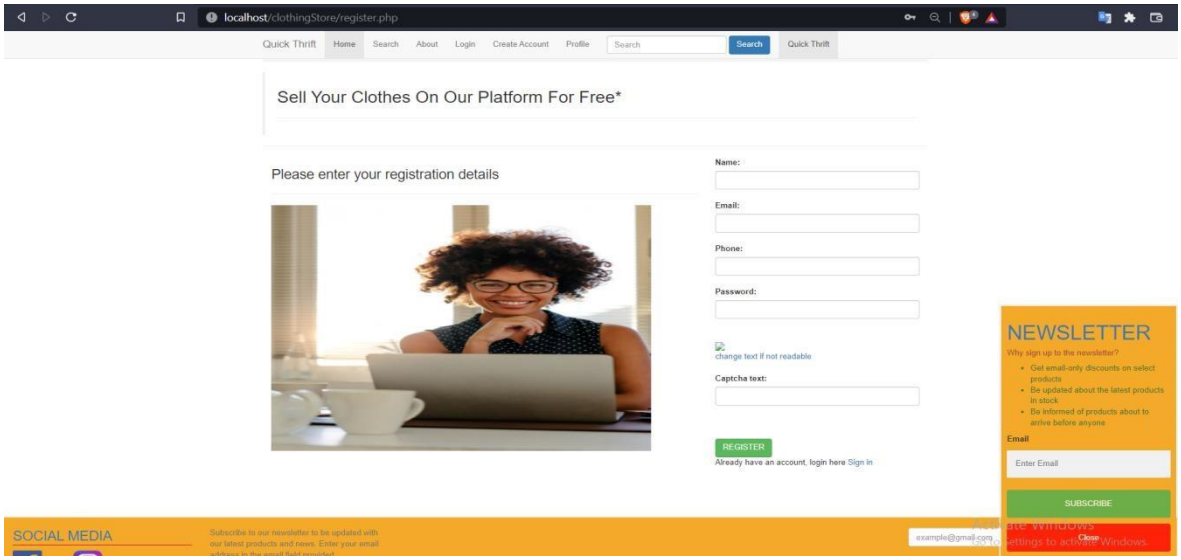
The page is about showing information about the company. The basic information, including the founder and how to work with the company, and some contact details are also on the page.





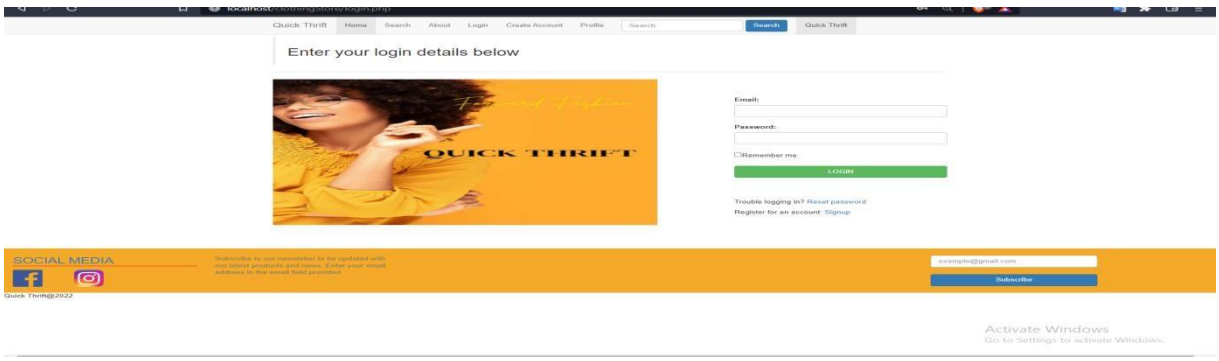
Register Page

This page allows new users to create an account on the system. First, new accounts are added to the database and activated by an administrator. Then, the user inserts the full name, email address, phone number, and the correct captcha and sends the form information.



Login Page

If the account is active, the user can then log on to the account and post new listings.



Uploading Items to sell (Admin)

Once logged in, the user can post items that should be listed on the website, including the images associated with the item.

Appendix 2: Sample Code

Index.PHP code

```
<?php
include("conn.php");
 include("includes/header.php");
?>
   <div id="carousel-example-generic" class="carousel slide" data-ride="carousel">
      <ol class="carousel-indicators">
        <li data-target="#carousel-example-generic" data-slide-to="0" class=""></li>
        <li data-target="#carousel-example-generic" data-slide-to="1" class=""></li>
      <li data-target="#carousel-example-generic" data-slide-to="2" class="active"></li>
        <li data-target="#carousel-example-generic" data-slide-to="3" class=""></li>
        <li data-target="#carousel-example-generic" data-slide-to="4" class=""></li>
      </ol>
      <div class="carousel-inner" role="listbox">
        <div class="item">
                                    <img class = 'fill' style='width:100%;' data-
src="holder.js/1140x500/auto/#777:#555/text:First slide" alt="First slide [1140x500]"
src="uploads/try1.jpg" data-holder-rendered="true">
          <!--<div class="carousel-caption">
                <p style="font-size:3em;border:2px solid orange;padding:4px;" class='text-
primary'>Take advantage of the black friday promotions</p>
            </div>-->
       </div>
        <div class="item">
```

```
                                <img class = 'fill' style='width:100%;' data-src="holder.js/1140x500/auto/#666:#444/text:Second slide" alt="Second slide [1140x500]" src="uploads/try2.jpg" data-holder-rendered="true">
          <!--<div class="carousel-caption">
                <p style="font-size:3em;border:2px solid orange;padding:4px;" class='text-primary'>Up to 30% off</p>
            </div>-->
        </div>
        <div class="item active">
                                <img class = 'fill' style='width:100%;' data-src="holder.js/1140x500/auto/#555:#333/text:Third slide" alt="Third slide [1140x500]" src="uploads/try3.jpg" data-holder-rendered="true">
          <!--<div class="carousel-caption">
                <p style="font-size:3em;border:2px solid orange;padding:4px;" class='text-primary'>Free deliveries nationwide</p>
            </div>-->
        </div>
        <div class="item">
                                <img class = 'fill' style='width:100%;' data-src="holder.js/1140x500/auto/#555:#333/text:Third slide" alt="Third slide [1140x500]" src="uploads/try4.jpg" data-holder-rendered="true">
          <!--<div class="carousel-caption">
                <p style="font-size:3em;border:2px solid orange;padding:4px;" class='text-primary'>Promotion runs between 23rd - 30th November</p>
            </div>-->
        </div>
        <div class="item">
```

```
                                <img class = 'fill' style='width:100%;' data-src="holder.js/1140x500/auto/#555:#333/text:Third slide" alt="Third slide [1140x500]" src="uploads/try5.jpg" data-holder-rendered="true">
          <!--<div class="carousel-caption">
                <p style="font-size:3em;border:2px solid orange;padding:4px;" class='text-primary'>Genuine leather</p>
            </div>-->
        </div>
       </div>
                                                <a class="left carousel-control" href="http://getbootstrap.com/examples/theme/#carousel-example-generic" role="button" data-slide="prev">
        <span class="glyphicon glyphicon-chevron-left" aria-hidden="true"></span>
        <span class="sr-only">Previous</span>
       </a>
                                                <a class="right carousel-control" href="http://getbootstrap.com/examples/theme/#carousel-example-generic" role="button" data-slide="next">
        <span class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span>
        <span class="sr-only">Next</span>
       </a>
     </div>


    <!-- end of courosel -->
    <div id="container">
    <div class='row'>
       <!-- most viewed -->
         <div class='col-xs-12 col-sm-12 col-lg-12 col-md-12'>
```

```
  <div  class='col-xs-2 col-sm-2 col-lg-2 col-md-2'>
  </div>
  <div  class='col-xs-8 col-sm-8 col-lg-8 col-md-8'>
     <h3 style='border-bottom:2px solid orange;'> most viewed</h3>
  </div>

  <div  class='col-xs-2 col-sm-2 col-lg-2 col-md-2'>
  </div>
</div>
<?php


$name = $views = $price = $availability = $image1 = "";
$sql = "SELECT * FROM posts ORDER BY views desc";
     $image1 = '';
     $runQuery = mysqli_query($con,$sql)or die(mysqli_error($con));
     if($runQuery)
     {
        $count = 0;
        while($data = mysqli_fetch_array($runQuery))
        {
           $id = $data['postid'];
           $name = $data['name'];
           $views = $data['views'];
           $price = $data['price'];
           $availability = $data['availability'];
```

```
                    //get image thumbnail
                        $getimage = mysqli_query($con,"SELECT * FROM images WHERE
postid='$id'")or die(mysqli_error($con));
                    if($getimage){
                        while($data = mysqli_fetch_array($getimage)){
                            $image1 = $data['image1'];


                            if($image1 == "")
                            {
                                $image1 = "uploads/genu.jpg";
                            }
                        }
                    }

                    if($count<6){
                        if($image1 == "")
                            {
                                $image1 = "uploads/genu.jpeg";
                            }
                        if($availability == "available")
                        {
                            $color = "#66FF66";
                        }else{
                            $color = "#CC0000";
                        }
                         echo "<a href='product-details.php?id=".$id."'>";
                        echo      "<div class='col-xs-12 col-sm-6 col-md-6 col-lg-4'>
```

```
                    <div class='someDiv'>
                                </div>
                                    <img src="'.$image1.'" class='col-xs-6 col-sm-4 thumbnail'
height='150' width='150'>
                                <div class='col-sm-6 col-xs-6'>
                                <h4>".$name."</h4>
                                <p>".$views." views </p>
                                <p style='color:orange;'>ZMK: ".$price."</p>
                                                    <p>availability:<span  style='color:$color;'>
".$availability."</span></p>
                                </div>

                              </div>

                            </a>
                            ";
                        $count++;
                    }

                }
            }


        ?>


    </div>
```

```
<div class='row'>
    <!-- featured -->
    <div class='col-xs-12 col-sm-12 col-lg-12 col-md-12'>
      <div  class='col-xs-2 col-sm-2 col-lg-2 col-md-2'>
      </div>
      <div  class='col-xs-8 col-sm-8 col-lg-8 col-md-8'>
          <h3 style='border-bottom:2px solid orange;'> featured</h3>
      </div>

      <div  class='col-xs-2 col-sm-2 col-lg-2 col-md-2'>
      </div>
    </div>
     <?php


    $name = $views = $price = $availability = $image1 = "";
    $sql = "SELECT * FROM posts WHERE featured='1' ORDER BY views desc";
        $image1 = '';
        $runQuery = mysqli_query($con,$sql)or die(mysqli_error($con));
        if($runQuery)
        {
            $count = 0;
            while($data = mysqli_fetch_array($runQuery))
            {
                $id = $data['postid'];
                $name = $data['name'];
```

```
                    $views = $data['views'];
                    $price = $data['price'];
                    $availability = $data['availability'];
                    //get image thumbnail
                       $getimage = mysqli_query($con,"SELECT * FROM images WHERE
postid='$id'")or die(mysqli_error($con));
                    if($getimage){
                       while($data = mysqli_fetch_array($getimage)){
                          $image1 = $data['image1'];



                       }
                    }

                    if($count<6){
                       if($image1 == "")
                          {
                             $image1 = "uploads/genu.jpeg";
                          }
                             if($availability == "available")
                       {
                          $color = "#66FF66";
                       }else{
                          $color = "#CC0000";
                       }
                        echo "<a href='product-details.php?id=".$id."'>";
                       echo      "<div class='col-xs-12 col-sm-6 col-md-6 col-lg-4'>
```

```
                    <div class='someDiv'>
                                </div>
                                    <img src='".$image1."' class='col-xs-6 col-sm-4 thumbnail'
height='150' width='150'>
                                <div class='col-sm-6 col-xs-6'>
                                <h4>".$name."</h4>
                                <p>".$views." views </p>
                                <p style='color:orange;'>ZMK: ".$price."</p>
                                                    <p>availability:<span  style='color:$color;'>
".$availability."</span></p>
                                </div>

                              </div>

                            </a>
                            ";
                        $count++;
                    }

                }
            }


        ?>
    </div>
    </div>
    <?php  include("includes/footer.php");?>
```

```
    </body>
</html>
```

Login.php

```
<?php
include("conn.php");
$username = $password = $dbpassword = $dbuser= $dbstatus = $loginProblem ="";
$cookieName = "username";
if(isset($_COOKIE[$cookieName])){
    $_SESSION['username']=$_COOKIE['username'];
   // header("Location:profile.php");
}


if($_SERVER['REQUEST_METHOD']=='POST'){
    echo "2";
    if(empty($_POST['email'])||empty($_POST['password'])){
        $loginProblem .= "<p> enter username and password</p>";

    }else{
        $username = validate($_POST['email']);
        $password = validate($_POST['password']);

    }
    if($loginProblem ==""){
```

```
                $sqlCheck = "SELECT email,password,activation FROM members WHERE email='$username';";
        $runCheck = mysqli_query($con,$sqlCheck)or die(mysqli_error($con));
        if($runCheck){
            while($inTheDb = mysqli_fetch_array($runCheck)){
                $dbpassword = $inTheDb['password'];
                $dbuser = $inTheDb['email'];
                $dbstatus = $inTheDb['activation'];


            }
                if($username == $dbuser && $password == $dbpassword && $dbstatus =='activated'){
                if(isset($_POST['me']))
                {
                   $_SESSION['username']=$username;
                        setcookie($cookieName,$username,time()+(86400*30),"/");
                        header("Location:profile.php");
                }else{
                        $_SESSION['username']=$username;
                        header("Location:profile.php");
                }


            }else{
                  $loginProblem .= "<p>your last login attempt was unsuccessful <br> make sure your credentials are correct<br>make sure you are activated if you are from registering </p>";
                 $_SESSION['error']=$loginProblem;
```

```
            }


        }
    }else{


    }
}


?>
<?php
include("includes/header.php");
?>


<div class="container">
    <div class="row">
      <div class="col-lg-12">
          <ol class="breadcrumb">
              <li><a href="index.php">Home</a></li>
              <li class="active">Sign up</li>
          </ol>
      </div>
    </div>
    <blockquote>
      <h2>Enter your login details below</h2>
    </blockquote>
```

```
    <hr>
      <div class="row">
        <div class="col-sm-8">
              <img src="uploads/genuine2.png" alt="login home picture" height="450" width="650">
        </div>
    <div class="col-sm-4 contactBox">
                        <form name="login" method="POST" action="<?php echo $_SERVER['PHP_SELF']; ?>">
                      <p style="color:red";><?php if(isset($_SESSION['error'])){echo $_SESSION['error']; }?> </p>
            <br><br>
            <label for="email">Email: </label>
            <input type="text" class="form-control" name="email" />
            <br>
            <label for="password">Password:</label>
            <input type="password" class="form-control" name="password"/>
            <br>
            <input type="checkbox" name="me">Remember me
            <br><br>
                    <input type="submit" class="btn btn-large btn-block btn-success" value="LOGIN">
            <br>
      <!-- Login APIs -->
      <div class="buttons col-sm-12" style="padding-left: 0;">
          <!--  <a href="" class="btn btn-primary" style="color: #fff;" role="button"><i class="fa fa-facebook"></i>acebook login</a>
```

```
                    <a href="" style="color: #fff; float: right;" class="btn btn-danger"
role="button"><i class="fa fa-google" style="color:#fff;"></i>mail login</a>
    --> </div><br><br>
            <p>Trouble logging in?<a href="forgot.php"> Reset password</a> </p>
            <p>Register for an account: <a href="register.php">Signup</a> </p>
        </form>
    </div>

      </div>
      </div><br><br>



<?php
include("includes/footer.php");
?>
```

Register.php

```
<?php
include("includes/header.php");

$_SESSION['SIGNUP']="";
include("conn.php");//database connection and external function for removing special
characters

$type = $password = $passConfirm = $fullname = $phone =$email =$username
=$request = $captcha = $captchaM = $RegError = "";
```

```
if($_SERVER["REQUEST_METHOD"]=="POST")
{
    echo "<p>test in</p>";

if (empty($_REQUEST['captcha'])) {//captcha code url   http://code.google.com/p/cool-php-captcha
        if  (empty($_SESSION['captcha'])  ||  trim(strtolower($_REQUEST['captcha']))  != $_SESSION['captcha']) {
        $captchaM= "WRONG CAPTCHA";
        $RegError .= "WRON CAPTCHA";
    } else {
        $captcha_message = "Valid captcha";

    }
}

if(empty($_POST['email']))
{
    $RegError .="<p> the email is empty</p>";
}else
{
    $email = $_POST['email'];
    if(!filter_var($_POST['email'],FILTER_VALIDATE_EMAIL))
    {
        $RegError .= "<p>the email format is invalid </p>";
    }else{
        $email = validate($email);
```

```
    }
}
if(empty($_POST['name']))
{
    $RegError .="<p> name is empty</p>";
}else
{
    $username = $_POST['name'];

    $username = validate($username);


}

if(empty($_POST['phone']))
{
    $RegError .="<p>phone is empty</p>";
}else
{
    $phone = $_POST['phone'];

    $phone = validate($phone);


}
if(empty($_POST['password']))
{
```

```
    $RegError .="<p> please check that the password field is not empty</p>";
}else
{
    $password = $_POST['password'];

    if(strlen($password)<6)
    {
        $RegError .= "<p> password is too short</p>";
    }else
    {
        $password = validate($password);

    }

}
if ($RegError == "")
{
    echo "<p>test no error</p>";
    $checkIfUserExistsSql = "SELECT name,email FROM members;";
                    $runCheckQuery      =      mysqli_query($con,$checkIfUserExistsSql)or
die(mysqli_error($connectionToServer));

    if($runCheckQuery)
    {

        while($inDbUser = mysqli_fetch_array($runCheckQuery))
```

```
        {


            $currentUser = $inDbUser['name'];
            $currentEmail = $inDbUser['email'];
            if($currentUser == $username)
            {
                $RegError .= "<p> username is already taken</p>";
            }else if($currentEmail == $email)
            {
                $RegError .= "<p>email is already in use </p>";
            }
        }


        if($RegError == "")
        {
$insertSql = "INSERT INTO members(name,email,phone,password,activation)
VALUES('$username','$email','$phone','$password','inactive');";


$runInsert = mysqli_query($con,$insertSql)or die(mysqli_error($con));


if($runInsert)
{
    $RegError .= "<p>your account has been created, check your email to verify and start
using your account</p>";
    $Message = "Hello, $username, click on the link to activate your account, if activated
successfully you will be taken to the login page <a
href='http://Quick_Thrift.net/activate.php?user=$username'>click here to activate</a>";
```

```
    $_SESSION['SIGNUP'] = $RegError;
    mail($email,"Quick_Thrift - ACTIVATION",$Message);
    $_SESSION['username']= $username;
     $_SESSION['message'] = "<p>check your email address to activate your account please </p>";
    $cooke = $username;
    header("refresh:6;url=index.php");
    //echo "<a href='activate.php?user=$username'>click here to activate</a>";


}
}
}
}else{
    $_SESSION['SIGNUP']="<p>failed to run query</p>";


}


}


?>


<div class="container">
    <div class="row">
      <div class="col-lg-12">
          <ol class="breadcrumb">


          </ol>
```

```
    </div>
  </div>
  <blockquote>
    <h2>Sale On our Platform For Free</h2><hr>


  </blockquote>
  <hr>
  <div class="col-sm-8">
            <h3  class="text-danger"  style='color:red;font-weight:bold;'><?php    echo $RegError;?></h3>
      <h3> Enter your registration details</h3>
    <h4> <?php  if(isset($_SESSION['message']))echo $_SESSION['message'];?></h4>
      <hr>
    <img src="uploads/try1.png" alt="login home picture" height="450" width="650">
  </div>
  <div class="col-sm-4 contactBox">
            <form    name="register"    method="POST"    action="<?php        echo $_SERVER['PHP_SELF']; ?>">
      <label>Name (individual/company):</label>
      <input type="text" class="form-control" name="name"><br>


      <label>Email: </label>
        <input type="text" class="form-control"  name="email" />
      <br>
      <label> phone:</label>
        <input type="text"  class="form-control" name="phone"/>
      <br>
```

```
    <label>Password:</label>
      <input type="password" class="form-control"  name="password"/>
    <br>
    <br>
    <img src="captcha.php" id="captcha" /><br/>
    <!-- CHANGE TEXT LINK -->
    <a href="#" onclick="
      document.getElementById('captcha').src='captcha.php?'+Math.random();
      document.getElementById('captcha-form').focus();"
      id="change-image">change text if not readable</a><br/><br/>
    <div id="captchathis">
      <?php
      if($_SERVER['REQUEST_METHOD']=='POST'){
          echo <<<HTML
    <h2>$captchaM</h2>
    <table>
    <tr>


    </tr>
    <tr>


    <td>$request</td>
    </tr>
    </table>

HTML;
```

```
}
?>
<label for="captcha">Captcha text:</label>
<input type="text"  class="form-control" name="captcha"/><br /><br />


<br>
<input type="submit" class="btn btn-success" value="REGISTER">
<br>
<p>Already have an account, login here <a href="login.php">Sign in</a></p>
</form>
</div>
    <br><br><br>
</div>
</div>
<br>


<?php
include("includes/footer.php");
?>
```

Appendix 3: Ethical Consideration Certificate

THE UNIVERSITY OF ZAMBIA

DIRECTORATE OF RESEARCH AND GRADUATE STUDIES

Great East Road Campus | P.O. Box 32379 | Lusaka 10101 | **Tel:** +260-290 258/291 777
Fax: (+260) 211 290 258/253 952 | **Email:** director.drgs@unza.zm | **Website:** www.unza.zm /directorates/drgs

APPROVAL OF STUDY

IORG No. 0005376
NASREC- IRB No. 00006465

25th September, 2023

REF NO. NASREC-2023- MAY - 016

Mr. Mike Daka,
The University of Zambia
School of Engineering,
P.O. Box 32379
LUSAKA

Dear Mr. Daka,

RE: "STRENGTHENING WEB-APPLICATION SECURITY THROUGH TECHNICAL MEASURES"

Reference is made to your protocol dated as captioned above. NASREC resolved to approve this study and your participation as Principal Investigator for a period of one year.

REVIEW TYPE	ORDINARY REVIEW	APPROVAL NO. NASREC-2023—MAY- 016
Approval and Expiry Date	Approval Date: 25th September , 2023	Expiry Date: 24th September, 2024
Protocol Version and Date	Version - Nil.	24th September, 2024
Information Sheet, Consent Forms and Dates	• English.	To be provided
Consent form ID and Date	• Version - Nil	To be provided
Recruitment Materials	• Nil	Nil
Other Study Documents	• Interview Guide.	

Specific conditions will apply to this approval;

As Principal Investigator it is your responsibility to ensure that the contents of this letter are adhered to.

If these are not adhered to, the approval may be suspended. Should the study be suspended, study sponsors and other regulatory authorities will be informed.

Conditions of Approval

- No participant may be involved in any study procedure prior to the study approval or after the expiration date.
- All unanticipated or Serious Adverse Events (SAEs) must be reported to NASREC within 5 days.
- All protocol modifications must be approved by NASREC prior to implementation unless they are intended to reduce risk (but must still be reported for approval). Modifications will include any change of investigator/s or site address.
- All protocol deviations must be reported to NASREC within 5 working days.
- All recruitment materials must be approved by NASREC prior to being used.
- Principal investigators are responsible for initiating Continuing Review proceedings. NASREC will only approve a study for a period of 12 months.
- It is the responsibility of the PI to renew his/her ethics approval through a renewal application to NASREC.
- Where the PI desires to extend the study after expiry of the study period, documents for study extension must be received by NASREC at least 30 days before the expiry date. This is for the purpose of facilitating the review process. Documents received within 30 days after expiry will be labelled "late submissions" and will incur a penalty fee of K500.00. No study shall be renewed whose documents are submitted for renewal 30 days after expiry of the certificate.
- Every 6 (six) months a progress report form supplied by The University of Zambia Natural and Applied Sciences Research Ethics Committee as an IRB must be filled in and submitted to us. There is a penalty of K500.00 for failure to submit the report.
- When closing a project, the PI is responsible for notifying, in writing or using the Research Ethics and Management Online (REMO),both NASREC
- and the National Health Research Authority (NHRA) when ethics certification is no longer required for a project.
- In order to close an approved study, a Closing Report must be submitted in writing or through the REMO system. A Closing Report should be filed when data collection has ended and the study team will no longer be using human participants or animals or secondary data or have any direct or indirect contact with the research participants or animals for the study.
- Filing a closing report (rather than just letting your approval lapse) is important as it assists NASREC in efficiently tracking and reporting on projects. Note that some funding agencies and sponsors require a notice of closure from the IRB which had approved the study and can only be generated after the Closing Report has been filed.
- A reprint of this letter shall be done at a fee.
- All protocol modifications must be approved by NASREC by way of an application for an amendment prior to implementation unless they are intended to reduce risk (but must still be reported for approval). Modifications will include any change of investigator/s or site address

or methodology and methods. Many modifications entail minimal risk adjustments to a protocol and/or consent form and can be made on an Expedited basis (via the IRB Chair). Some examples are: format changes, correcting spelling errors, adding key personnel, minor changes to questionnaires, recruiting and changes, and so forth. Other, more substantive changes, especially those that may alter the risk-benefit ratio, may require Full Board review. In all cases, except where noted above regarding subject safety, any changes to any protocol document or procedure must first be approved by NASREC before they can be implemented.

Should you have any questions regarding anything indicated in this letter, please do not hesitate to get in touch with us at the above indicated address.

On behalf of NASREC, we would like to wish you all the success as you carry out your study.

Yours faithfully,

Dr. M. Kaonda

VICE - CHAIRPERSON
THE UNIVERSITY OF ZAMBIA NATURAL AND APPLIED SCIENCES RESEARCH ETHICS COMMITTEE - IRB

cc: Director, Directorate of Research and Graduate Studies
Assistant Director (Research), Directorate of Research and Graduate Studies
Assistant Registrar (Research), Directorate of Research and Graduate Studies

Appendix 4: Paper Publication Certificates

International Research Journal Of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

e-ISSN: 2582-5208

Ref: IRJMETS/Certificate/Volume 05/Issue 05/50500012320 Date: 05/05/2023

Certificate of Publication

This is to certify that author "**Mike Daka**" with paper ID "**IRJMETS505000 12320**" has published a paper entitled "STRENGTHENING WEB APPLICATION SECURITY THROUGH TECHNICAL MEASURES" **in International Research Journal Of Modernization In Engineering Technology And Science (IRJMETS), Volume 05, Issue 05, May 2023**

Editor in Chief

IRJMETS Impact Factor 7.868

We Wish For Your Better Future

www.irjmets.com