

STRENGTHENING WEB APPLICATION SECURITY THROUGH TECHNICAL MEASURES

Mike Daka*¹, Dani E. Banda*²

*^{1,2}Department Of Electrical And Electronical Engineering, University Of
Zambia, Lusaka, Zambia.

ABSTRACT

Web application security is becoming increasingly critical as cybercriminals devise new techniques to infiltrate and exploit web-based systems. This research aimed to investigate the various types of malware that commonly attack web applications, the strengths and weaknesses of current malware detection techniques, and technical measures that can be implemented to strengthen web application security and prevent malware attacks. A comprehensive literature review identified several types of malware that pose a significant threat to web applications, including cross-site scripting (XSS) and SQL injection attacks. These attacks can lead to significant financial losses, reputational damage, and even legal liabilities for affected organisations. However, several technical measures can be implemented to strengthen web application security and prevent malware attacks to mitigate the risk of these attacks. These include vulnerability scanning and penetration testing, implementing access control mechanisms, employing encryption technologies, and enforcing secure coding practices. However, the effectiveness of these measures depends on the thoroughness of their implementation and maintenance. We found that the key to effective web application security is a comprehensive and ongoing approach to risk management, which involves identifying and prioritising potential threats, implementing appropriate technical measures, and training employees and individuals to be vigilant and proactive in detecting and reporting potential security incidents.

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

I. INTRODUCTION

The security of web applications is critical for any business that operates on the Internet. Web application security refers to the measures taken to protect web applications from cyber-attacks, which can result in data breaches, financial losses, and reputational damage [1]. In addition, the increasing use of web applications has made them a prime target for cybercriminals who seek to exploit vulnerabilities in these applications [2]. As such, businesses must implement technical measures to enhance web application security.

One of the most effective technical measures for enhancing web application security is the implementation of secure coding practices. Secure coding practices involve writing code resilient to attacks by ensuring it is free from vulnerabilities. This can be achieved by adhering to coding standards and guidelines such as the Open Web Application Security Project (OWASP) Top Ten, which outlines the most common web application vulnerabilities [3]. By following these guidelines, developers can write more secure code and reduce the potential for cyber-attacks.

Another technical measure that can be employed to enhance web application security is encryption. Encryption involves converting data into a secret code to ensure that it cannot be easily intercepted or read by unauthorised individuals; this is particularly important when transmitting sensitive data such as credit card information or personal identification details [4], [5]. In addition, encryption protocols such as Transport Layer Security (TLS) can help protect web applications from threats such as man-in-the-middle attacks.

Web application firewalls (WAFs) are another technical measure that can be employed to enhance web application security. WAFs analyse incoming traffic to identify and block malicious requests, helping to prevent attacks such as cross-site scripting (XSS) and SQL injection [6]. Additionally, WAFs can be configured to monitor and report on web application activity, enabling businesses to detect and respond to security incidents in real time.

With increasing cyber threats, businesses must prioritise securing their web applications. This research has explored various technical measures that can be employed to enhance web application security, including secure coding practices, encryption, and WAFs. By implementing these measures, businesses can reduce the likelihood of cyber-attacks and protect their assets from potential harm. In addition, businesses need to stay

updated with the latest web application security trends and best practices to ensure that they are adequately protected against evolving threats.

1.1. Background

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 transmit 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 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. To address this challenge, organisations can use software composition analysis tools 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.

II. LITERATURE REVIEW

2.1 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 SQL Injection

Structured Query Language (SQL) injection is one of the most common web application attacks, targeting the application's database [7]. 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 [8].

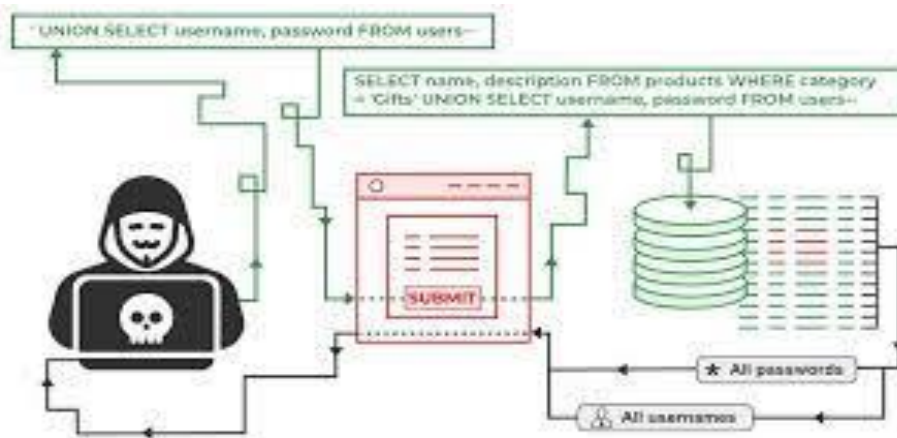


Figure 0-1: SQL Injection Attack Implementation [7]

In-band SQLi is the most common type of SQL injection [9]. 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 [10]. 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 [10]. 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 [8]. 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 [10], [11].

2.3 Dirbuster

DirBuster is another type of web application attack that targets the application's file and directory structure [12]. In a DirBuster attack, an attacker uses a tool that attempts to brute-force directory and file names on the application's server, as shown in Figure 0-1. The goal is to discover hidden files or directories containing sensitive information, such as configuration files or user credentials [13].

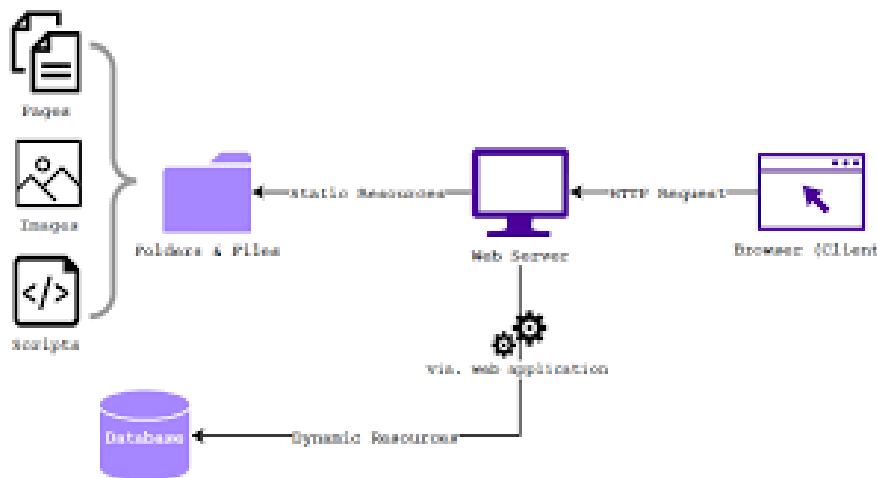


Figure 0-1: DirBuster Attack Implementation [14]

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 [15]. 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 [16].

The characteristics of DirBuster 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 [14]. 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 DirBuster 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 [12]–[14]. 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 [17]–[19].

2.4 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 [20]. 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 0-1.

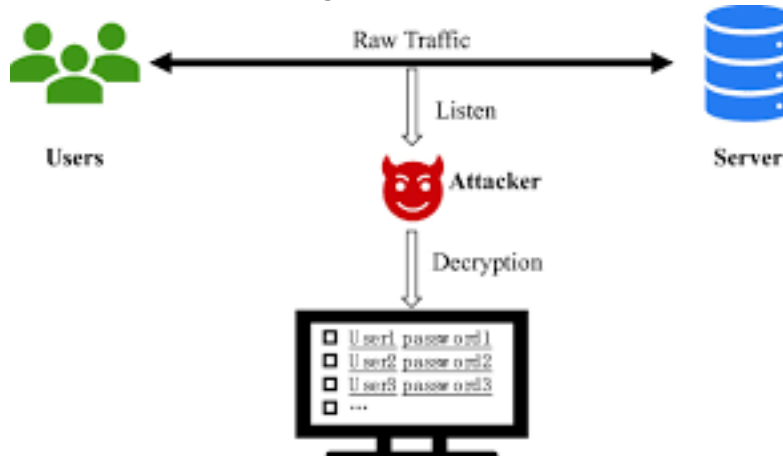


Figure 0-1: Password Sniffing Attack Implementation [21]

The behaviour of password sniffing malware in web applications is to capture login credentials as they are transmitted over the network [21], [22]. The malware will intercept network traffic and search for login credentials transmitted 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[23]–[25].

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 [26]. 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 [26], [27]. 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.[26]

2.5 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 [28]. 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 [29]. The malware can then transmit the captured packets to a command and control server where the attacker can analyse and extract the sensitive information.

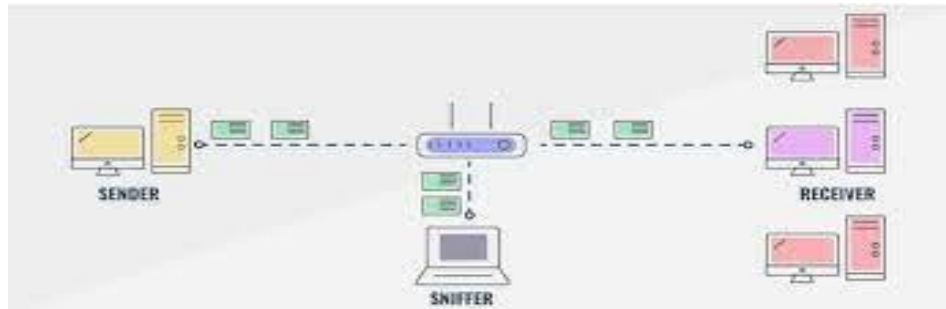


Figure 0-1: Packet Sniffing Attack Implementation [30]

Packet sniffing malware can be difficult to detect as it does not modify the traffic it captures, making it challenging to identify [31]–[33]. 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 [30].

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 [34].

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 [34]. Additionally, network monitoring tools can detect and alert abnormal network traffic, which can help identify the presence of packet-sniffing malware [35].

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.

2.6 Security Measures for Mitigating Malware Attacks

Web applications have become integral to modern-day business and social interactions [36]. 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 [37]. 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.

2.7 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 [38].

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 [37].

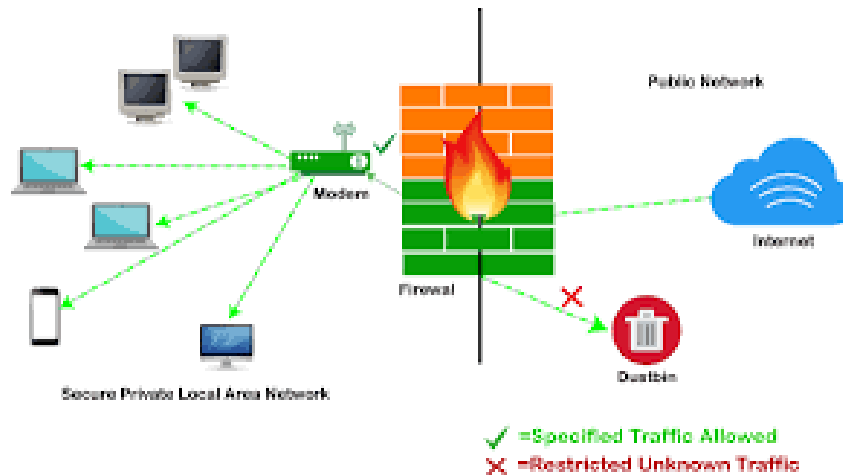


Figure 0-1: Firewall Implementation [6]

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 [39]. 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 [40].

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 [40].

III. METHODOLOGY

This research employs a mixed-mode research design consisting of two phases. The first phase involves a review of existing literature [18], [41]–[44] 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 [18], [41]–[44]; 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.

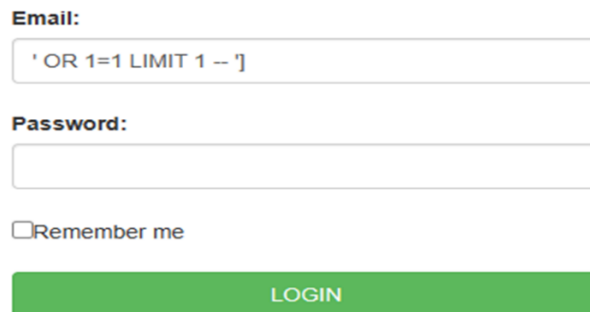
The non-vulnerable site is designed to access the vulnerabilities identified in the test site and encounter them; this allows for evaluating the effectiveness of technical measures in strengthening web application security by identifying vulnerabilities and developing strategies to mitigate them.

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 [45].

3.1 SQL Injections

In the current section of the research, the SQL injection technique known as "Bypassing Login" was performed. This section aims to showcase the SQL commands used 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.



Email:
' OR 1=1 LIMIT 1 -- '

Password:

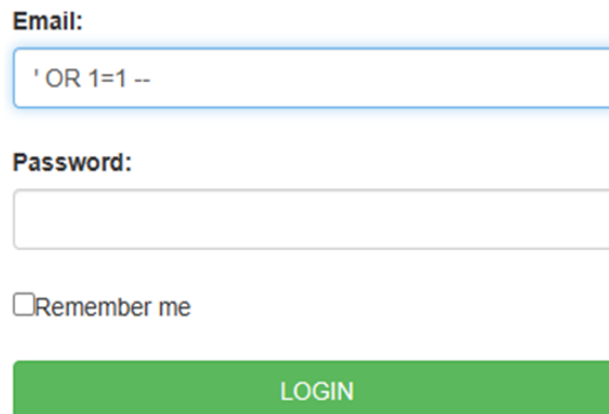
Remember me

LOGIN

Figure 0-1: Bypassing Login Attack

The attack code is displayed in **Error! Reference source not found.**, 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 0-2.



Email:
' OR 1=1 -- '

Password:

Remember me

LOGIN

Figure 0-2: Shorter SQL Bypassing Login Attack

3.2 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 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)—"

However, due to the lengthy query, it is not fully visible in **Error! Reference source not found.** Therefore, the complete query is pasted in the appendix.

Email:

Password:

 Remember me

LOGIN

Figure 0-1: 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%' )))--
```

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 0-2, the database name was quick Thrift, as shown below in the error message.

Email:

Password:

 Remember me

LOGIN

Trouble logging in? [Reset password](#)

Register for an account: [Signup](#)

Figure 0-2: Database Name Fingerprinting Attack

3.3 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 unauthorised access to sensitive files and information. Figure 0-1 shows the Dirbuster attack setup used.

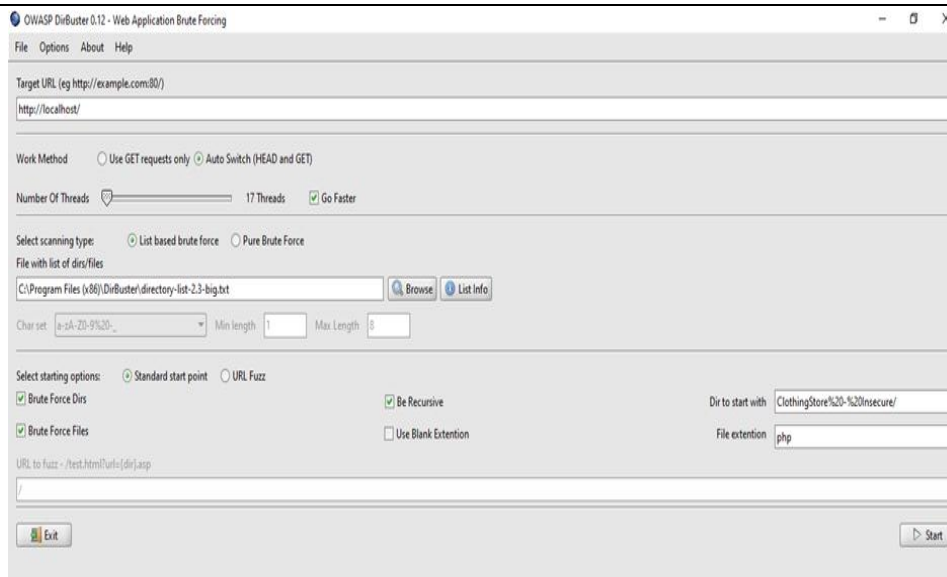


Figure 0-1: DirBuster Attack Setup

The DirBuster is a software developed by the Open Web Application Security Project (OWASP) that was utilised as a tool 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.

3.4 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.

3.5 Packet Sniffing attacks

Packet Sniffing allows capturing and analysing 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.

IV. RESULTS AND DISCUSSION

This section presents the results and analysis of various web application attacks, including SQL injection, database enumeration, directory traversal, DirBuster, and password sniffing conducted in chapter 3 of this research based on research objectives 2: To analyse 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.

4.1 SQL Injections

The SQL injection technique, "Bypassing Login", was performed. The successful operation has shown in Figure 4.1.1.

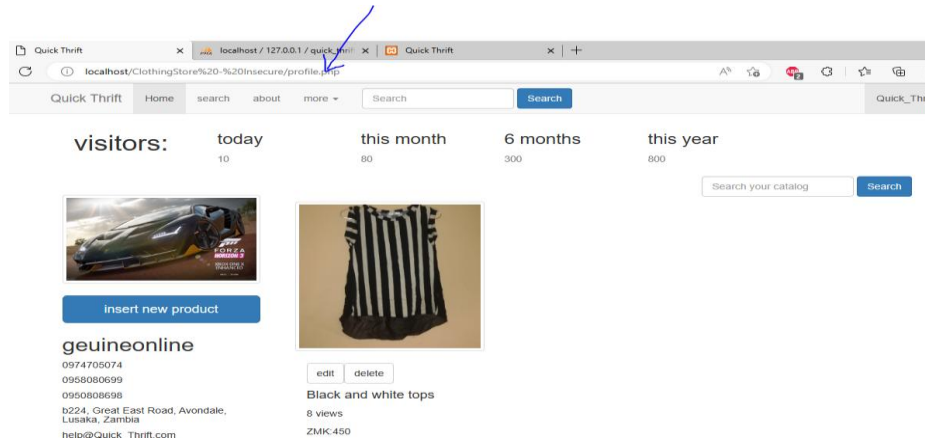


Figure 0-1: SQL Injections Attack Result

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.

If the hacker is aware of the username or email address for a specific victim in what would be characterised as a targeted attack, the hacker can use the email address or the 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.

4.2 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.

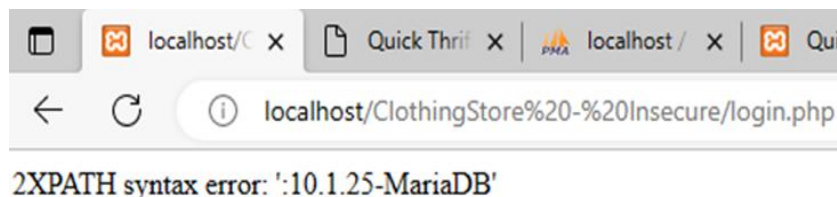


Figure 0-1: Database Version Fingerprinting Result

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 0-1 and the database name "quick thrift" in Figure 0-2.

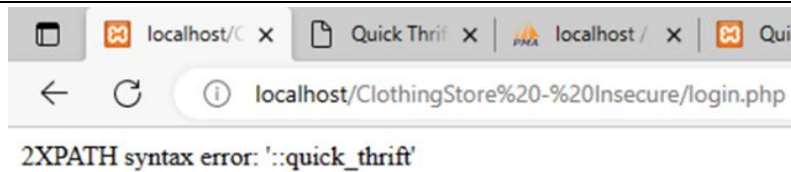


Figure 0-2: Database Table Name Fingerprinting Result

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 0-3.

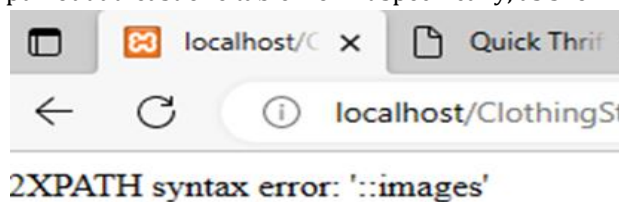


Figure 0-3: Traversing Quick Thrift Database Results

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)))—
```

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 utilising 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)))--
```

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 unauthorised access.

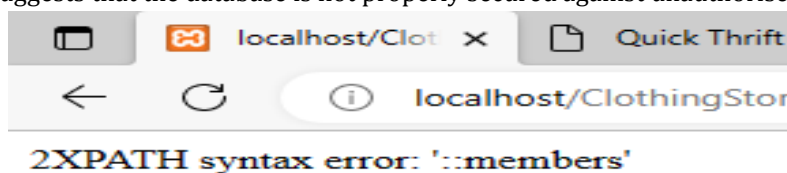


Figure 0-4: Enumerating the Tables Attack

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, parameterised queries, and stored procedures.

Using the SQL statement below, we could recognise the column passwords from the results in Figure 0-5.

```
' 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)))-
```

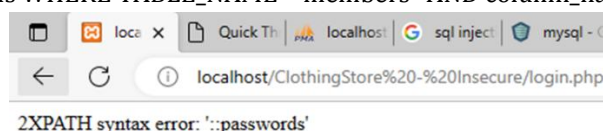


Figure 0-5: Enumerating the Column Names Attack

Finally, using the SQL statement below, the result showed that the username "temwani" and the password "temwani" was found after the query ran, as shown in Figure 0-6.

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

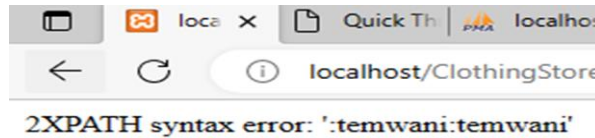


Figure 0-6: Enumerating the Tables Password Attack

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 unauthorised 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.

V. CONCLUSION

One commonly used malware detection technique based on research findings is signature-based detection. The research results show that this approach involves the creation of signatures for known malware and comparing them against incoming traffic to detect and block malware. Signature-based detection is effective against known malware but is limited in detecting new or unknown malware.

Another technique is behaviour-based detection. This method involves monitoring web application behaviour to identify deviations from normal behaviour and detect malware. Behaviour-based effectively detects unknown or new malware but can result in false positives due to normal deviations from typical behaviour.

Another approach is anomaly-based detection, which involves establishing a baseline of normal behaviour and identifying deviations from this baseline as potential malware attacks. This technique effectively detects new or unknown malware and can detect behaviour-based attacks not detected by signature-based detection. However, anomaly-based detection can also produce false positives due to normal behaviour deviations.

In recent years, machine learning-based techniques have gained popularity in detecting and mitigating malware attacks[46]–[48]. These techniques use machine learning algorithms to identify patterns and anomalies in web application traffic that may indicate malware activity. Machine learning-based techniques have shown promising results in detecting unknown and zero-day malware attacks.

Despite the effectiveness of these techniques, they also have weaknesses. For instance, research results indicate that malware detection techniques that rely on signatures may be evaded by attackers who can manipulate traffic to evade detection can deceive machine learning-based techniques can deceive machine learning-based techniques. Furthermore, these techniques can produce false positives, challenging administrators to distinguish them from legitimate traffic.

The strengths and weaknesses of current malware detection techniques for web applications vary depending on the specific technique used. Therefore, no single technique is foolproof, and a combination of techniques may be needed for effective malware detection and mitigation. Organisations should also implement regular software updates and patches to address vulnerabilities and reduce the risk of malware attacks. Additionally, organisations should conduct regular security awareness training for employees to minimise the risk of social engineering attacks.

VI. RECOMMENDATIONS AND FUTURE WORK

1. Web application developers should take security seriously and prioritise 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. Organisations should have a security policy that clearly outlines the security measures that should be implemented in web applications.
4. It is recommended that organisations 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.

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.

VII. REFERENCES

- [1] McAfee, "Web Application Security: A Beginner's Guide." 2021.
- [2] The Open Web Application Security Project (OWASP), "Web Application Security Best Practices." 2021.
- [3] Amazon Web Services (AWS), "Security Best Practices for Web Applications." 2021.
- [4] M. A. Khan, M. T. Quasim, N. S. Alghamdi, and M. Y. Khan, "A Secure Framework for Authentication and Encryption Using Improved ECC for IoT-Based Medical Sensor Data," IEEE Access, vol. 8, 2020, doi: 10.1109/ACCESS.2020.2980739.
- [5] N. Lungu, D. Banda, and N. Luka, "SIDEBAR ATTACKS ON GPUS," International Research Journal of Modernization in Engineering Technology and Science, Feb. 2023, doi: 10.56726/IRJMETS33377.
- [6] Cisco, "Next-Generation Firewalls: An Overview," 2022.
- [7] D. Bisson, "New SQL Injection Attack Technique Emerges," Tripwire, Jan. 2022.
- [8] J. Davis, "SQL Injection: A Beginner's Guide to Understanding and Prevention," HealthITSecurity, Mar. 2022.
- [9] T. Keary, "The Dangers of SQL Injection and How to Prevent Them," CSO Online, Feb. 2022.
- [10] M. Samuels, "SQL Injection Attacks on the Rise as Hackers Exploit Pandemic," ZDNet, Mar. 2022.
- [11] M. Burnett, "Preventing SQL Injection Attacks with Parameterized Queries," Security Boulevard, Jan. 2022.
- [12] A. Kumar, "DirBuster – Brute Force Directories and Files Hidden in Web Application," The Hack Today, Oct. 2021.
- [13] D. Balaban, "Using DirBuster to Find Hidden Directories on Web Servers," TechNadu, Nov. 2021.
- [14] A. Crenshaw, "DirBuster: A Tool for Finding Hidden Web Objects," in Black Hat Asia, 2022.
- [15] K. Makan, "Burp Suite Tutorial: Using DirBuster to Find Hidden Web Content," Infosec Institute, Jun. 2021.
- [16] S. Rana, "DirBuster Tutorial: Finding Hidden Files and Directories," TechViral, Mar. 2021.
- [17] 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.
- [18] 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.
- [19] Google Cloud, Modern Web Application Security. Google LLC, 2022.
- [20] Varonis, "Password Sniffing: How It Works and How to Defend Against It." Nov. 2021.
- [21] McAfee, "Protect Yourself from Password Sniffing." Nov. 2021.
- [22] TechBeacon, "Password sniffing attack: What it is and how to prevent it." Jan. 2022.
- [23] Cloudflare, Web Application Security: A Beginner's Guide. Cloudflare Inc., 2022.

- [24] 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.
- [25] M. Heiderich and G. Heyes, "XSS Without HTML: Client-Side Template Injection with AngularJS," in *Black Hat Europe 2016*, 2016.
- [26] Norton LifeLock, "Password Sniffing: How Hackers Steal Your Credentials." Jan. 2022.
- [27] Avast, "What is Password Sniffing and How Can You Prevent It?" Feb. 2022.
- [28] SolarWinds MSP, "Packet Sniffing – A Threat to Your Network Security." Jul. 2021.
- [29] TechBeacon, "Packet Sniffing: What It Is and How to Protect Yourself." Mar. 2022.
- [30] McAfee, "How to Detect and Prevent Packet Sniffing." Feb. 2022.
- [31] 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.
- [32] A. Tyagi and A. Rastogi, "Understanding Cross-Site Scripting (XSS) Vulnerabilities," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 687–707, 2018.
- [33] J. Manico and A. Detlefsen, "The Anatomy of Cross-Site Scripting Attacks," *OWASP Top Ten Project*, 2017.
- [34] Avast, "Packet Sniffing: What it is and what you can do about it." Jun. 2021.
- [35] Norton LifeLock, "What Is Packet Sniffing and How Can You Avoid It?" Oct. 2021.
- [36] Veracode, "Web Application Security Checklist: Best Practices for Secure Development," Mar. 2021, [Online]. Available: <https://www.veracode.com/security/web-application-security/checklist>
- [37] 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>
- [38] 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/>
- [39] 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>
- [40] 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>
- [41] 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.
- [42] 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.
- [43] Akamai Technologies, *Web Application Security: How to Avoid Authentication Attacks*. Akamai Technologies Inc., 2022.
- [44] 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.
- [45] 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.
- [46] 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.
- [47] R. Thangaraj and S. Santhosh Baboo, "A Deep Learning Approach for Intrusion Detection System in Cloud-Based Networks," *J Ambient Intell Humaniz Comput*, 2022.
- [48] MarketsandMarkets, "Next-Generation Firewall Market to Reach \$6.71 Billion by 2026." 2021.