Wasit Journal for Pure Science

Journal Homepage: https://wjps.uowasit.edu.iq/index.php/wjps/index
e-ISSN: 2790-5241 p-ISSN: 2790-5233

Critical Evaluation of SQL Injection Security Measures in Web Applications

Haneen Mohammed Athab*

Ministry of Education, General Directorate of Education in Wasit Governorate, IRAQ

*Corresponding Author: Haneen Mohammed Athab

DOI: https://doi.org/10.31185/wjps.566
Received 20 November 2024; Accepted 21 December 2024; Available online 30 March 2025

ABSTRACT:SQL injection attacks remain a persistent and critical threat to web application security, driving the need for innovative and adaptive defense mechanisms. This research addresses these challenges through the development and evaluation of advanced tools and methodologies. The *Agent-based Vulnerability Response System* (AVRS) introduces mobile agents and machine learning to enhance traditional intrusion detection systems, enabling real-time threat detection and response. Furthermore, the VIWeb vulnerability scanner with Decision Trees, Support Vector Machines (SVMs), and Artificial Neural Networks (ANNs) is suggested to evaluate malware detection. ANNs perform better than the others, with the greatest accuracy of 86.50%. The results highlight the value of combining machine learning methods with traditional security measures, opening the door to stronger defenses against SQL injection attacks and guiding future technological advancements and cybersecurity implementation projects.

Keywords:SQL Injection, Agent-based Vulnerability Response System (AVRS), Machine Learning, Vulnerability Scanning, Malware Detection

1. INTRODUCTION

SQL injection attacks have become one of the most significant and persistent threats to the security of web applications, thereby posing serious risks to sensitive user information and organizational data. Malicious Uniform Resource Locators (URLs) play a significant role in enabling cybercrimes and attacks by hosting unsolicited material such as spam, phishing schemes, cross-site scripting, and injection attacks. The dynamic nature of malware URLs, especially newly created ones, makes their identification and prevention very difficult. Although old measures are designed to restrict harmful URLs, they sometimes fail to counter the rate at which the threat is changing. This paper comes up with a String-Matching Algorithm that compares URL strings with known malicious patterns that prompt the user to validate the authenticity of the site.

Whereas this is on top of a number of efforts to neutralize malice URLs from the websites, there arises the need to have more serious strategies regarding vulnerabilities in the web-based applications. Toward solving this gap, this article introduces a total system design that is oriented toward vulnerability reduction at an application level aimed at countering cyber assault. The presented structure features the main three main components: namely AVRS, namely VIWeb Scanner and MLMD. These components use advanced agent-based mechanisms for responding to vulnerabilities and machine learning techniques to identify and mitigate injection-related vulnerabilities.

The rapid expansion of web applications in the recent past, driven by advances in web technologies, has necessitated robust cybersecurity measures. The fact that web applications allow users to access services such as online banking, email, and social media from a browser grants them unmatched convenience, especially since they can be accessed from virtually any location. However, this wide reach also makes web applications very vulnerable to security threats, since web servers store very high volumes of sensitive and private data, including user credentials and organizational information. [3].

1.1 SQL Injection Vulnerabilities: A Historical Overview and Their Exploitation

Security professionals continue to worry about the SQL injection vulnerability, even though it has been known about for over twenty years. In 1998, the vulnerability was discovered. A Microsoft SQL server might jeopardize sensitive data by leveraging simple user input commands like "name" and "telephone number" in December 1998, according to renowned security researcher Jeff Forristal's warning in Phrack magazine after the initial discovery. The magazine published the description that was written by Forristal. Despite SQL injection's 1998 disclosure, the information security community didn't appear to take notice until 2002. Along with the advent of deadly worms and viral outbreaks on the internet, the four-year-old vulnerability was revealed shortly after the national event [4].

Since SQL injection became suddenly famous in 2002, a plethora of research and academic papers have been written and published on the topic. To deal with SQL injection and other malware attacks, these papers and articles have been developed. Just like a lot of other parts of computer technology, SQL injection attacks may be unpredictable at times. By employing the piggyback technique on SQL statements embedded within the user's input, the malevolent SQL injection attacker purposefully carried out an attack, as reported in the December 1998 issue of Phrack magazine. The Piggyback technique is well recognized as a great method for granting unwanted users access to SQL database data. Many types of attacks could compromise the system and get illegal data. Among them are attacks based on tautology, union queries, stored processes, inference, and alternative encoding, among other techniques. Despite the passage of time, the most common type of SQL injection attack in 2013 appeared to be tautology and inference [5]. According to security specialists, both desktop and web applications can be compromised by SQL injection attacks. This vulnerability to SQL injection can be exploited in databases that use SQL. The perpetrators might initiate the assault by copying and pasting the malicious query into their browser's URL bar or by using a web form. One of these techniques could result in a breach of a website. A special kind of attack occurs when the attacker tries to change the HTTP header to get around the malicious code. Modules that collect user data, like IP address, browser type, and spoken language, might be used to track users' activity. Online apps frequently make use of this module. Following its collection via the victim's browser's HTTP connection, this data is subsequently added to a database. A SQL injection attack occurs when a malevolent actor uses the stored procedure method as a string to access the database using the supplied data, which incorporates both dynamic SQL expressions and unformatted user input.

Figure 1 shows the SQL injection attack technique. A SQL injection attack involves an attacker introducing a string-type input into an online application with the intent to modify and alter the SQL statement. By then, the hacker will have complete access to the database. By removing sensitive data and making unauthorized modifications to the database, for example, an exploit of this SQL vulnerability could cause multiple types of database damage.

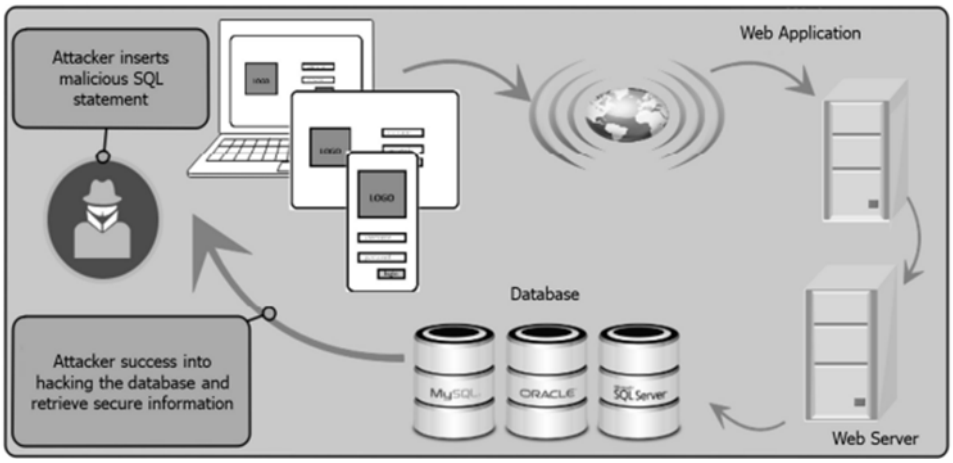


FIGURE 1.-Diagram of SQL Injection Attack Workflow in Web Applications

Figure 1 illustrates the workflow of an SQL injection attack in web applications. The attacker inserts a malicious SQL query into a user input field, which is processed by the web application. This query is executed by the backend database, allowing the attacker to access sensitive information. The diagram shows how different databases like MySQL, Oracle, and SQL Server can be targeted in such attacks. This form of attack can execute instructions at the system level, which means it might potentially cause the system to refuse the application's requests for service. By using a SQL injection attack, a hacker may acquire complete database access to a remote server by skipping the authentication process. Almost every online project nowadays uses SQL databases to keep track of user information. Also, the vast majority of web applications make use of SQL databases that may operate indefinitely in the background. SQL's syntax, like that of any programming language, makes it simple to modify the database command to fit the current data. People in remote

locations can now use the online application and enter data thanks to this. The user can fully control the database and all of its features by reading this data as a command [6].

The authors demonstrated that there are two main types of SQL injection attacks that might affect web applications. The first type of attack that binds SQL statements directly using direct binding is called a direct injection attack. This type of SQL injection is known as "first-order SQL injection" since it targets the SQL query's payload. The first step in running the code is to enter it directly into the user input variables. The SQL query is then used to merge the variables. A second type of attack known as an indirect attack or second-order injection attack involves inserting malicious code into strings that are part of a database table. Attacks of this type are classified as second-order attacks. A SQL query is added to the saved string to execute the malicious code [7]. For the injection procedure to be carried out correctly, the text string must be stopped before any more instructions are carried out. Second-order SQL injection attacks are characterized by a two-step procedure: In the initial stage, the attack payload is stored in a file system or database. Using the attack to generate a SQL query statement is the second step. Web applications that handle user data and registrations are among those that could be subject to second-order SQL injection attacks. The attacker utilizes this function to insert the payload into the user registration function of the database. After that, it employs the technique to alter the database's user data in order to retrieve the attack's payload. This prepares the SQL query that would initiate the second-order injection attack [8].

1.2 Contribution of the Research

The research provides a comprehensive assessment of vulnerabilities in SQL injection and corresponding mitigative strategies, with a focus on over URL attack vectors, thereby making the contribution to the field of web application security. It defines and classifies some common patterns of attacks such as query manipulation and XSS to illustrate how malicious attackers can exploit input parameters against the integrity of an application otherwise. Further, the research critiques security measures such as the Boyer-Moore string matching algorithm and server-side input validation techniques with regards to their capabilities to defend against SQL injection attacks. Additionally, the study offers practical recommendations on improving web application security, such as using advanced mechanisms for scanning URLs, better input validation protocols, and strict coding styles. Such research would be pertinent to developers and security experts in fortifying web applications against emerging threats..

2. LITERATURE REVIEW

Web Application Security Risks and SQL Injection Attacks

The increasing popularity of web applications has led to significant security risks in particular with the increase in the number of SQL injection attacks. Karthikeyan et al. (2021) assert that for over two decades, the vulnerability of SQL injection remains one of the major threats toward the security of online applications. Web applications with poor construction are particularly vulnerable, leading to potential exploitation through SQL injection (SQLIA) attacks, where malicious SQL statements are executed to gain unauthorized access to databases [9]. This attack can compromise sensitive data, triggering data breaches, fraud, and system failures.

Sharma, C. et al. (2016) Online applications that are easy to use have proliferated in tandem with the ever-improving web development technologies. These apps are now a part of everyone's everyday life due to the extensive usage of online banking, purchasing, booking, and trading. Also, the profit-driven online business sector has noticed this growth. This is because an organization gains access to a global platform with a successful application. The database stores sensitive information about individuals and businesses, making it the most valuable asset of an online service. The most serious hazard comes from SQLIA, which attacks databases used by web applications. If an attacker gains control of the application, they can do anything they want with it, like delete databases, steal sensitive information, or commit financial fraud. Based on real-world research, this report provides a comprehensive summary of SQL injection attacks. This includes deploying an injection mechanism for each attack and the kind of assault that it is. Online programs, phony databases, and a variety of websites all use this method. Additionally, the most crucial security method for online application databases is discussed in order to guard against SQL injection attacks [11].

SQL Injection Detection and Prevention Strategies

Several detection and prevention techniques have been the main focus of efforts to reduce the hazards associated with SQL injection. Devakunchari, R., Abirami, J., and Valliyammai, C. (2015) In this era of the internet, people have basic needs that can be satisfied by using online services. Massive volumes of data must be gathered and stored in order to process these internet-based services. Another type of interaction media that is in charge of the daily storing of vast volumes of data are social media platforms. Databases containing this data must always be maintained. The usage of SQL

queries can be used to access, modify, and retrieve data from databases. The proof of data for future purposes is compromised if an opponent finds a way to alter inquiries through the use of unauthorized access. This type of SQL injection attack has been shown to be rather common in web applications. To protect the vast volume of data, such an attack must be neutralized and dealt with as soon as possible. The study's findings provide insight into the importance of data, the effects of SQL injection attacks on data, and the techniques used to identify and stop these attacks [10].

Machine Learning Approaches for SQL Injection Detection

A. Pramod et al. (2015) A web-based system's functionality could be jeopardized by a wide range of threats. Actually, a range of services built on databases are offered via web-based systems. As a result, they are susceptible to structured query language (SQL) injection attacks. Because of this, a lot of study has been done to deal with attacks like this. The defense method, which is used in the great majority of protection techniques, finally provides a high number of positive rates during times of severe reaction time. Actually, attacks involving SQL code injection are a major worry for web-based systems at all times. Despite the fact that hackers still find this type of attack enticing, it is becoming more advanced. Consequently, a number of research initiatives have been proposed as possible remedies for this issue. The supplied technologies are founded upon core principles such as a statistical or dynamic approach, machine learning, or deep learning. The purpose of this essay is to investigate and analyse the many approaches used to detect and avoid SQL injection attacks. In addition, it provides an overview of the difficulties, problems that still exist, and potential future developments in terms of remedies in this area [12].

Traditional Defense Methods

Vieira, M., and Antunes, N. (2009) When online services are deployed, there are frequently significant software flaws that could be dangerously abused. It is generally accepted that using online vulnerability scanners is an easy way to assess the security flaws in web applications. However, prior research has demonstrated that the applicability of these methods in situations involving Web services is highly doubtful. In actuality, the high number of false-positives and the scant coverage that was achieved through actual application highlight the serious limitations of these technologies. This post will show that it is possible to create a Web service vulnerability scanner that performs better than the ones that are sold commercially. The goal of the study is to demonstrate this. Consequently, we present a technique for identifying SQL injection vulnerabilities, one of the most common and dangerous types of vulnerabilities that can be present in online settings. Our experimental evaluation shows that our approach performs significantly better than popular commercial tools. It achieves a very high detection coverage rate with extremely few false positives [13].

This paper addresses a large gap in the current SQL injection defense approaches by exploring the combination of advanced machine learning techniques with traditional security measures. Existing solutions have mainly focused on static methods such as input validation or blacklist-based filtering, which have been proven inadequate to keep pace with ever-changing attack techniques. However, the use of mobile agent-based systems for real-time threat detection and response in SQL injection attacks is limited in its exploration. This research attempts to provide a more dynamic and adaptive defense mechanism, incorporating agent-based vulnerability response systems and machine learning models for detecting and mitigating SQL injection attacks, thereby promoting effective and proactive security solutions for web applications.

3. METHODOLOGY

The proposed system uses MSMA to identify malicious URLs and prevent SQL injection attacks. The proposed methodology involves three major modules: URL scanning, malware detection through machine learning, and agent-based response systems.

3.1 URL BASED ATTACKS

An abstract identification of the resource's location, which is supplied via URLs, enables "locating" resources. In order to disseminate web pages, a hacker can simply take over a web server and alter certain portions of a URL. Cross-site scripting, SQL injection, remote code execution, and email header injection are just a few of the numerous risks that the supplied MSMA assessments for Figure 2 is a diagram that shows how the URL www.learnlive.co.in is structured. The name of the website and its contents make up a URL. [14] The data is input from the user and is used for background executing operations.

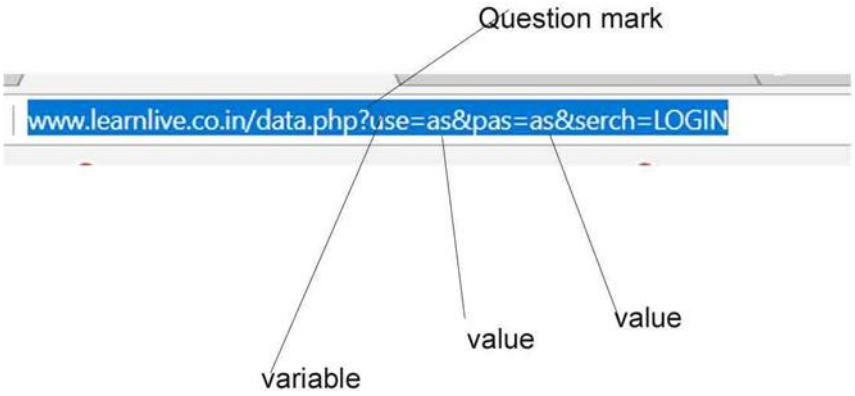


FIGURE 2. –A Sample URL

When a user logs in to the website, the variables use=as and pas=as are transmitted to the web server in this specific instance. The server is generating a SQL query after receiving these data. Once the variable is found to match, the browser is redirected to the requested page. The data kept on the server can be hijacked by a malicious user by directly changing the URL values (use=1 or 1=1 and pas=1 or 1=1) [15].

The topic of URL scanning has been the focus of several published studies with encouraging results. Fuqiang Yu proposed an algorithm based on Boyer-Moore pattern matching in his essay titled "For the purpose of determining whether or not the URL is secure, this method takes into account the viral characteristics that are stored in the database."" Using this method, the viral signature and the URL source are matched. Everything related to the descriptions of malicious internet addresses is included in the database.

TFA Rahman and colleagues presented a technique for identifying SQL injection in online environments using the Boyer-Moore string matching algorithm in their work "SQL Injection Attack Scanner." This model uses the analysis method of Boyer-Moore string matching. Based on pre-established standards, this methodology may identify online applications vulnerable to SQL injection attacks.

Asishkumar Dalai and Sanjay Kumar Jena named their online SQL injection detection method as "Neutralising SQL Injection Attack Using Server-Side Code Modification in Web Applications." This procedure discovers the harmful item using input string extraction. The input parameters are of the appropriate type in this model; otherwise, it makes a check. It is likely that the input parameters will be rejected in case of an unsuccessful search for the match; otherwise, they will be added to the query [16].

There are numerous types of vulnerabilities that have to be detected and managed suitably to make sure a web page is secured from attacks by means of URL manipulation. In what follows is a list of the numerous types of vulnerabilities that might be generated due to the direct manipulation of URL values:

3.2 SQL INJECTION

A malicious user can, for example perform a SQL injection attack on a particular website by modifying the URL value directly. Click on the link below to sign in to the website provided your username and password have been registered with that data [17].

The query related to the above operation is given below,

```
$use = $_GET['username'];
$pas = $_GET['password'];
mysql_query("SELECT * FROM table WHERE username = $use AND password = $pas");
```

SQL injection is possible by a hacker by directly modifying the URL.

3.3 CROSS SITE SCRIPTING (XSS)

Cross-site scripting (XSS) attacks allow attackers to install malicious code into websites that appear to be trustworthy. In the following example, a URL that may be used to find a person's name using his Aadhaar number is displayed [18].

http://www.learnlive.co.in/data.php?Aadhaar □ 158978922

The query related to the above URL is given below.

```
$nam= $_GET['Aadhaar']; echo "Your Adhar is $nam\n";
```

A hacker can tamper the above URL directly by inserting a malicious script. A tampered script may look like:

http : //www.learnlive.co.in/data.php?Adhar 158978952
=<script src= "evil.com /evi.Php " />

Here the attacker crafts a URL containing a malicious string

<script src= "evil.com /evi.Php " /> and sends it to the victim.

3.4 EXECUTION OF CODE FROM A REMOTE LOCATION

Remote Code Execution (RCE) is one method by which an application can inject and run code. These types of attacks exploit people's negligence when handling private data. This is an illustration of code that could have malicious code injected into it:

```
<?php
$command=$_GET[‘c’]; System($command);
?>
```

If a user modifies the URL and enters a command for remote code execution, the attack happens. For example [19],

http://www.learnlive.co.in/data.php? arg= 1; php inf o()

The PHP information page that is saved on the server is displayed in the example above when an attacker requests that arbitrary PHP code be executed.

3.5 E-MAIL HEADER INJECTION

One kind of injection vulnerability that affects online applications is email header injection, which allows user input to be changed to create email messages. The recipient of this kind of assault thinks the email is from a legitimate user. [20]. For example,

```
<form method="get"> Name
<input type="text" name="Name" /> Reply
<input type="text" name="re" /> Message
<input type="text" name="Me" />
</td></tr>
<input type="submit" name="serch" value="send" />
</form>
</table>
<?php if(isset($_GET['serch']))
{$name = $_GET['Name'];
$re = $_GET['re'];
$me = $_GET['Me'];
$headers = "From: $name \n" ."Reply-To: $re";
$to = 'ammashaji@learnlive.co.in'; mail($to, "check", $me, $headers);
}
```

When a user enters name as josemathew and the reply field is entered as shaji, the URL looks like: http://www.learnlive.co.in/emailinjection.php?serch=send&Name=josemathew&re= shaji&Me=how+r+u+

And the Mail header is as shown below,

From : josemathew@cp-in-14.webhostbox.net To : ammashaji@learnlive.co.in
Reply-To : shaji@cp-in-14.webhostbox.net Date Today 10:13

When an intruder enters name as mymail\nbcc: intruder@spam.com and reply field as intruder@intrud.com, then the URL looks like,

http://www.learnlive.co.in/emailinjection.php? serch=send&Name = my-mail+%5Cnbcc%3A+intruder%40spam.com&re=inruder%40inrud.co m&Me=how+r+u+

The new mail header looks like,

From : intruder@spam.com
To : ammashaji@learnlive.co.in Reply-To : intruder@inrud.com

A carbon duplicate that is blind This method appends the SMTP header to the message. Additional email addresses that the SMTP server will use to forward the email to BCC are contained in this header. It can be difficult to identify SQL attack patterns from a distance in front of the web server because of their great diversity. The method we have

suggested uncovers all types of SQL injection strings. The values of the URL parameters are frequently set in a page's URL in a dynamic way to make a website dynamic. In a URL, the question mark (?) is used to indicate the domain name's end [21]. Additionally, it indicates the start of the parameter list, which is separated from the domain name by the ampersand symbol (&). For example, the variables "use" and "pas," respectively, depict the values "username" and "123" in this instance. These variables are distinguished from one another by the & symbol. The provided MSMA model entails detecting the variables and their corresponding values by scanning the string that follows the question mark. The authenticity of the input in the URL is then assessed using MSMA. A URL that has been encoded with a certain character can also be used; this technique is known as percent encoding. ASCII characters are substituted in URL encoding by a "%" followed by two hexadecimal numbers that follow the same pattern. At the same time, either + or %20 is used to alter the URL's space.. One example is,

In the above URL “user name” is replaced as ‘user+name’, i.e. space is re- placed with +. ASCII encoding is also used for encoding URL values. For example,

http://www.learnlive.co.in/data.php?use=1%27or%271%27%3D%271&pas=1%27o
r%271%27%3D%271&serch=LOGIN

In above URL, the value of the variable, use and pas are, 1' or '1'='1. These variable values are replaced with 1%27or%271%27%3D%271. Here single quotes (‘) are encoded with %27 and = is replaced with %3D. This type of URL encoding is called ASCII encoding [22].

4. PERFORMANCE ANALYSIS AND RESULTS

4.1 MALICIOUS STRING-MATCHING ALGORITHM (MSMA)

In the context of URL Scanning, an attacker can perform injection by directly changing the URL and using encoded data. This is the type of direct assault that the proposed MSMA seeks to prevent. In particular, it compares a string to a malicious string contained within the URL. The process involves tokenizing the URL and using an indexed searching technique. Reading the URL and figuring out how many variables are contained inside it is the first stage in the process. It then finds the user-supplied values and checks them for any potentially harmful content [23]. Microsoft Security Management Assistant (MSMA) defends against SQL injection attacks as well as email header injection, cross-sitescripting, and remote code execution attacks. Figure 3 below shows a representation of the proposed model's structure.

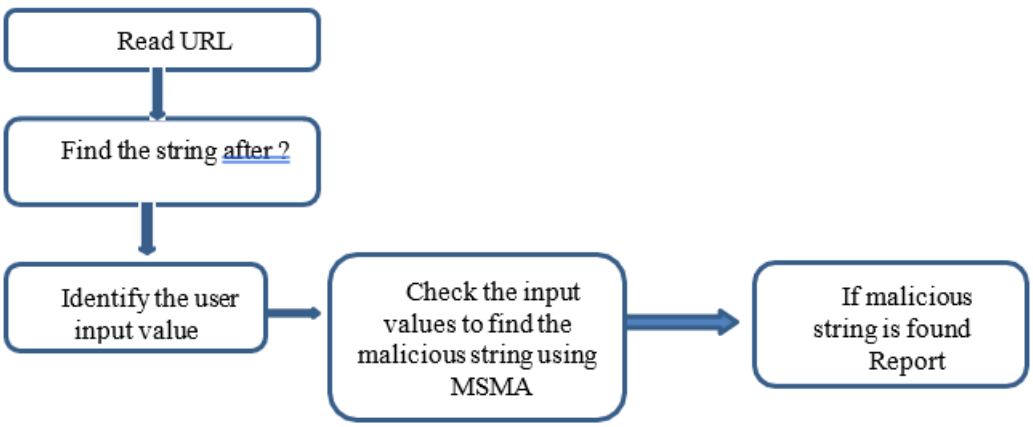


FIGURE 3. Workflow of URL scanning using MSMA

The algorithmic steps involved in the proposed model is given as:

Table 1. Algorithm MSMA

	Algorithm: MSMA
3	Read the URL
4	Find the string after the question mark
5	Find the number of ampersand symbol
6	Calculate the number of variables
7	Identify the input values
8	Find all the substrings in the URL variables
9	Extract the first letter in the substring
10	If the first letter is not present in the indexed table
11	The URL is good
12	Otherwise, check for all the indexed string in the table
13	If the match is found, the URL is malicious, report an injection attack
14	Otherwise, URL is good

To find a URL that contains valuable information mixed with hazardous data, MSMA is utilized. This algorithm verifies the authenticity of both the encoded URL and the conventional URL. The MSMA method will find the maliciously encoded URL value if the input text is encoded. Only when the string is encoded does this happen. Every hazardous string has an index in one database that matches its initial letter. All of the harmful strings are contained in this table. When a single quotation marks the end of a value that enters the system, it is considered that an injection attack has taken place [24].

Finding the total number of variables in the URL is the first step in this technique. Next, extract each value from the URL that was entered, and match the results with malicious data that was recorded in the database. The database is used to store the damaging values in an indexed manner. The approach used by the MSMA is called index-based mapping.

Consider the input string IS,

IS = mymail\nbcc: intruder@spam.com

The above input is encoded and, in the URL, it will appear as shown below.

IS=mymail+%5Cnbcc%3A+intruder%40spam.com

Included in the string that was just displayed is the harmful string %5Cnbcc. Initially, every substring included within the input value will be extracted by the algorithm. Comparing these substrings with malicious data that is kept in the database is the final stage. The columns that correspond to those malicious strings are kept, and the initial letter of every piece of destructive data that is entered into the database is given a primary key. The harmful table is made using this technique. Figure 4, which is shown below, shows how malicious string is kept in the storage area [25].

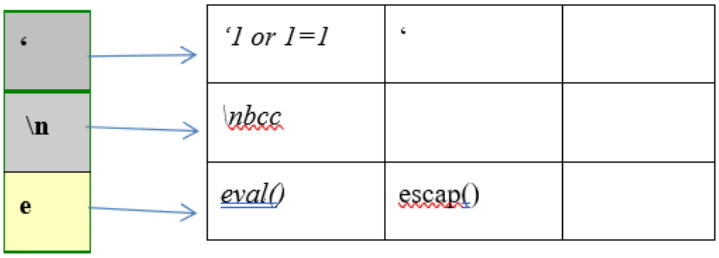
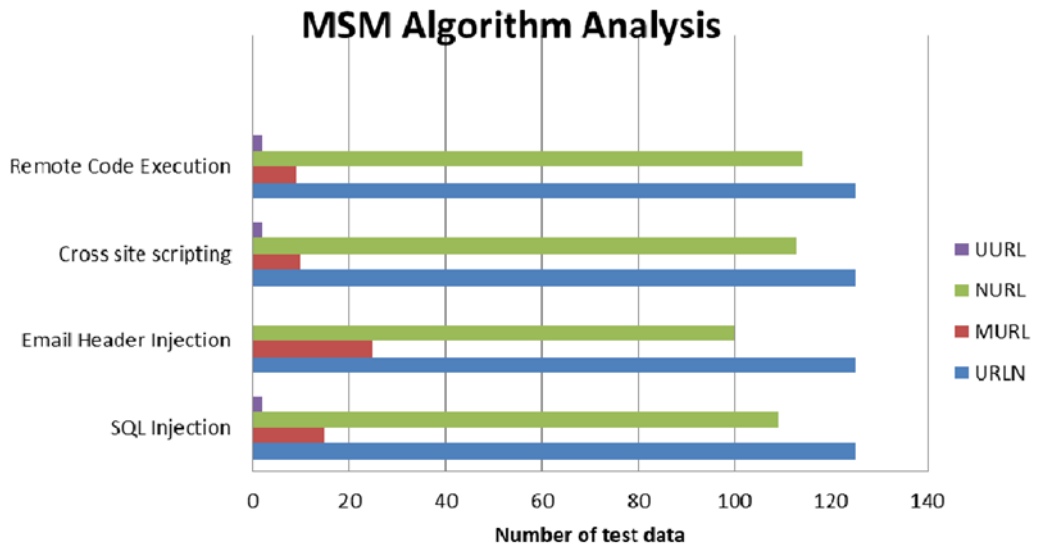


FIGURE 4. Storing malicious string

4.1.1 PERFORMANCE ANALYSIS AND RESULTS

This system is created by developing an application that is made with ASP.net. This system has a feature that lets it accept URLs, and a user can modify it if a new malicious string is found. To investigate the possibility of injection attacks, one can experiment by using both a malicious URL and a URL that contains data. Malicious URLs that are present online can be found using the method that has been proposed. In over 200 URLs that were analyzed in order to conduct experiments, SQL Injection, cross-site scripting, email headers, and remote code execution strings were found.

Many URLs with vulnerable strings for SQL injection, cross-site scripting, and remote code execution are made by scientists for testing purposes. Additionally, they create URLs devoid of attack strings. Following the extraction of each string from the URL, the algorithm checks it by comparing it to the harmful strings that are kept in the database. "MURL" denotes a malicious URL discovered during testing, "URLN" denotes the quantity of URLs used to carry out the test, "NURL" denotes a normal URL, and "UURL" denotes an unrecognizable URL. It is evident from the experimental results that the accuracy of this suggested model was 95.54%. In reality, it employs ASP.net to create this application, which makes up this system. This system has the ability to take in URLs, and the user may simply replace any newly discovered dangerous strings. A malicious URL and a URL with data to test the feasibility of injection attacks can be used to conduct the experiment. One can find the malicious URLs on the Internet by using the method that has been described. We discovered instances of SQL Injection, cross-site scripting, email headers, and remote code execution strings among the 200 or so URLs that were evaluated for testing purposes [26]. Several URLs are created in order to do experiments; some of these may contain attack strings that are susceptible to remote code execution, SQL injection, cross-site scripting, email header injection, and other vulnerabilities. Following the extraction of every string from the URL, the algorithm checks it by comparing it to the malicious strings that are kept in the database. "MURL" denotes the malicious URL that was found, "URLN" denotes the number of URLs utilized for the experiment, "NURL" denotes the Normal URL, and "UURL" denotes the URL that was not recognized. The experiments' findings indicate that the proposed model has an accuracy of 95.54%.



FIFURE 5. Test result of MSMA

4.2 A RESILIENT IMMUNE VULNERABILITY REDUCTION SYSTEM USING MACHINE LEARNING (RISecure-web)

The ability of human systems to adapt dynamically and protect themselves against biological threats is truly amazing. Computer security is becoming more and more dangerous, and research is still ongoing to design a solution that can stop viruses from infiltrating the system. Some of the first research in cyber security may adjust to changing environments, like the human body's immune system. According to Jamie Paul's thesis, "Integrated Innate and Adaptive Artificial Immune Systems for Anomaly Detection," an immune system for network security has been modeled after the biological adaptive immune system. The development of a strong and flexible cyber security system is motivated by the human immune system's remarkable adaptability and resilience. Similar to the proactive actions of the human immune system, this type of technology avoids infections [27].

This section attempts to provide an architectural view of a system that defends against cyberattacks by reducing application-level vulnerabilities, particularly those associated with the injection method. One major challenge for Intrusion Detection Systems (IDS) is maintaining real-time monitoring of network traffic, virus attacks, and log files. The time it takes to identify attacks and put countermeasures in place is a significant problem with many intrusion detection systems. Even worse, the majority of live systems are incapable of learning from their design. In this paper, we suggest a robust system architecture for creating a safe cyber defense system. To create a comprehensive security system, Muhammad Awais Shibili and associates created an immune system for network security. Sensors and mobile agents would be employed in this configuration. Among the many subsystems that comprise this system are the Intrusion Detection System, Vulnerability Analysis System, Security Management System, and Intrusion Response System. This technique uses clonally and negatively selected covert mobile agents. By examining the system's capacity

for learning and adapting to changing environments, the article goes even farther. Nevertheless, it is unable to combat the mystery virus that is attacking through learning techniques.
In order to defend online applications against SQL injection and URL manipulation injection attacks, we provide RISecure-online, a novel model that makes use of an agent-based machine learning system. Figure 5 depicts the architecture of the aforementioned system. Additionally, the technology allows for easy connection with any intrusion detection system that is already in place [28].

There are primarily three parts to RISecure-Web:
• AVRS, or Agent-Based Vulnerability Response System
• Web Security Exposure Identification (VI-Web)
• ML Anti-Malware Detection System

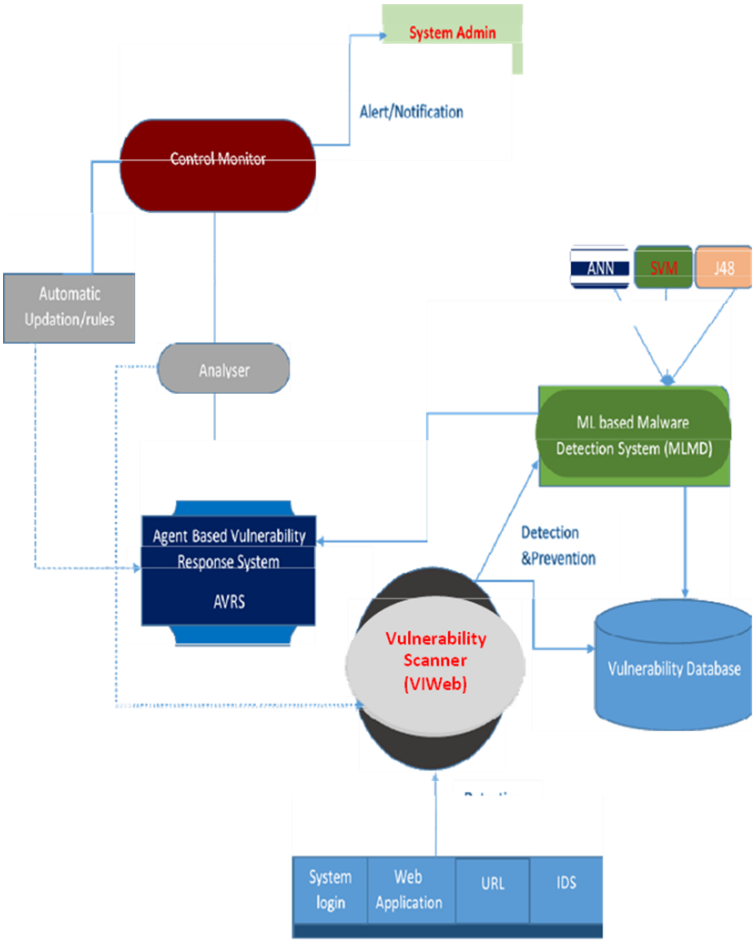


FIGURE 6. The RISecure-Web Architecture

4.2.1 AVRS, NAMELY AN ATTACK RESPONSE SYSTEM BASED ON AGENTS

The "Agent" is one type of autonomous computer program that can meet deadlines and objectives by itself. In addition to exhibiting dynamic behavior, a large number of agents can be configured or executed during runtime without human intervention. "Agents" are defined by their capacity for adaptation, independence, and cooperation. Agent-based intrusion detection systems collect information about the intrusion and the intruders, then respond dynamically and automatically block or alter sites [29]. Additionally, as a precaution, attackers try to turn off firewalls, intrusion detection systems, and other anti-virus software in order to implant infections that agents can also stop. This enables them to protect the machine and evade their attacks.

As each virus behaves differently, it might be challenging to establish an effective intrusion detection system. Because of the complex behavior and the increasingly intrusive infections and attacks, security software researchers and developers are having trouble sleeping. Since a single technique won't be enough to achieve the objective, tremendous

efforts must be made to establish a strong and inherent cyber security system. Research in these areas is ongoing in order to keep up with the ever-increasing complexity of assaults.

Due to its mobility and autonomy, the agent-based system can get around some of the drawbacks of traditional intrusion detection systems. Even if mobile agents are already familiar with this application domain, careful design decisions are still needed to properly utilize their capabilities. It is extremely difficult and complex to build an immune system that uses multiple substances [30].

An explanation of the AVRS approach is provided below.

Connecting the vulnerability scanner and machine learning system to the agent-based response system is feasible. All of the vulnerability information is in the vulnerability database. If an intrusion or assault is detected, the Central Manager is notified and checks the analyzer. This happens each time the analyzer receives a notification. Following a thorough study of the problem, the control manager will recommend system safeguards. The system will either immediately shut down or, in the event of a more serious attack, deactivate the cookies when a web application notifies ARS of an attack. ARS will also communicate the answers and changes to the system administrator. Future research will focus on the implementation of the AVRS concept, which is only briefly discussed in this study.

4.2.2 VULNERABILITY IDENTIFICATION FOR WEB (VIWeb)

Figure 6 illustrates the VIWeb vulnerability scanner that takes as inputs the Malicious String-Matching Algorithm and the Reverse Resemblance Algorithm. The information will be stored in a vulnerability-specific database, which may also contain the outputs of other intrusion detection systems.

4.2.3 MACHINE LEARNING-BASED MALWARE DETECTION SYSTEM (MLMD)

MLMD machine-learning algorithms have been created utilizing datasets retrieved from vulnerability databases maintained in vulnerability scanners. A brief description of three machine-learning techniques will be provided here: decision trees, support vector machines (SVMs), and artificial neural networks (ANNs)[31].

4.2.4UTILISING DECISION TREES, SVMS, AND ANNS

While evaluating the injection technique, URL lexical aspects were taken into account. Each attack has unique URL properties, therefore selecting the criteria is essential to accurate classification. On the Internet, rogue URLs and domains are known to be different from legitimate ones. In order to classify URLs as either benign or dangerous, this experiment uses URL attributes associated with the injection mechanism. A URL parser is used to extract tokens from strings delimited by ('/', '?', '.', '-', '_', and '=') in order to extract the remaining lexical features from the strings gathered by the Malicious String-Matching Algorithm. 89 examples were tested using nine criteria: identified name or phrase, encoded special character or operator, left-right equality, repeated numbers, SQL keywords, logical operators, special characters (e.g., apostrophe, parenthesis, etc.), and logical data structures [36]. To assess the results, 10-fold cross validation is used. We trained our models on this dataset using ANN, SVM, and Decision Tree with J48. Accuracy of generalization of test set algorithms is used to compare performance. Kappa statistics, mean absolute error, root mean square error, and the proportion of correctly classified occurrences are computed. With 86.50% accuracy, ANN is the most accurate of the three classifiers. It was a 9×3×2 ANN with a 0.3 learning rate and 0.2 momentum.

5. EVALUATION METRICS

Evaluation metrics form a critical foundation for assessing the effectiveness of machine learning classifiers in detecting SQL injection attacks. Accuracy, or the ratio of correctly classified incidents to the total number of cases, is the most basic measure of the overall reliability of a classifier.

Tables 1, 2, 3, and 4 show the outcomes of the performance comparison

Table 1.–Comparison of Results from ANN, SVM, and Decision Tree Approaches

Computer program	Accurately categorized incidents	Cases that were misclassified	Accuracy
ANN	78	12	86.50%
SVM	73	16	82.00%
J48	72	17	80.90%

Table 1 Comparison of ANN, SVM, and J48 in terms of well-classified incidents, misclassification, and overall accuracy. Based on the results, it is evident that ANN has performed best with an accuracy level of 86.50% with 78 correct classified incidents and only 12 misclassifications, while SVM and J48 show accuracies of 82.00% and 80.90%, respectively. This would emphasize the better capability of ANN to process and classify data more accurately, hence it's an excellent candidate for SQL injection attacks.

Table 2. – Mean Absolute Error, RMS Error, and Kappa Statistics of Artificial Neural Network, Support Vector Machine, and Decision Tree Approach.

Computer program	Mean Absolute error	Squared Root Mean Error	The Kappa Statistic
J48	0.2143	0.3709	0.5696
SVM	0.1798	0.424	0.5749
ANN	01.402	0.2971	0.6748

These metrics in table 2 reveal variability, consistency, and classification agreement in the predictions. ANN has the lowest RMSE at 0.2971, meaning its predictions are very consistent, but the MAE of ANN (1.402) is relatively higher than that of SVM (0.1798) and J48 (0.2143). Kappa statistic is the agreement beyond chance. ANN had the highest alignment with the actual classifications with a kappa statistic of 0.6748 compared to SVM at 0.5749 and J48 at 0.5696.These results validate ANN's capacity for robust classification, albeit with room for improvement in reducing prediction variability.

Table 3. – Classifiers' TP and FP Rates, as well as Their Recall, Accuracy, AND F-Measure, are Compared.

Computer program	TP Rate	FP Rate	Precision	Recall	F-Measure	Class
ANN	0.934	0.286	0.877	0.934	0.905	SQL Injection
	0.714	0.066	0.833	0.714	0.769	Valid
SVM	0.885	0.321	0.857	0.885	0.871	SQL Injection
	0.679	0.115	0.731	0.679	0.704	Valid
J48	0.836	0.250	0.879	0.836	0.857	SQL Injection
	0.750	0.164	0.677	0.750	0.712	Valid

Table 3 provides a comparative analysis of precision, recall, F-measure, TP Rate, and FP Rate for SQL injection and valid data classes. ANN outperforms in detecting SQL injection attacks, achieving the highest recall (0.934), precision (0.877), and F-measure (0.905). These metrics confirm ANN's ability to identify SQL injection cases with minimal false negatives and high accuracy. For valid data classification, J48 exhibits the highest recall (0.750), demonstrating its effectiveness in recognizing non-malicious instances, though its precision (0.677) is lower than ANN. SVM, while balanced, does not surpass ANN in any metric, showcasing ANN's dominance in critical detection scenarios.

Table 4. – Classifiers' Area Under Curves (AUCs)

	Algorithm		
CLASS	J48	SVM	ANN
SQL Injection	0.867	0.782	0.937
Valid	0.867	0.782	0.937

Table 4 above compares the AUC values of the three classifiers for SQL injection and valid data classes. AUC is a critical metric for evaluating a model's discriminatory power across various thresholds. ANN achieves the highest AUC (0.937), indicating its superior ability to distinguish between SQL injection and valid instances. J48 follows with an AUC of 0.867, while SVM scores the lowest at 0.782. ANN's strong AUC performance highlights its effectiveness in managing classification challenges in SQL injection detection scenarios.

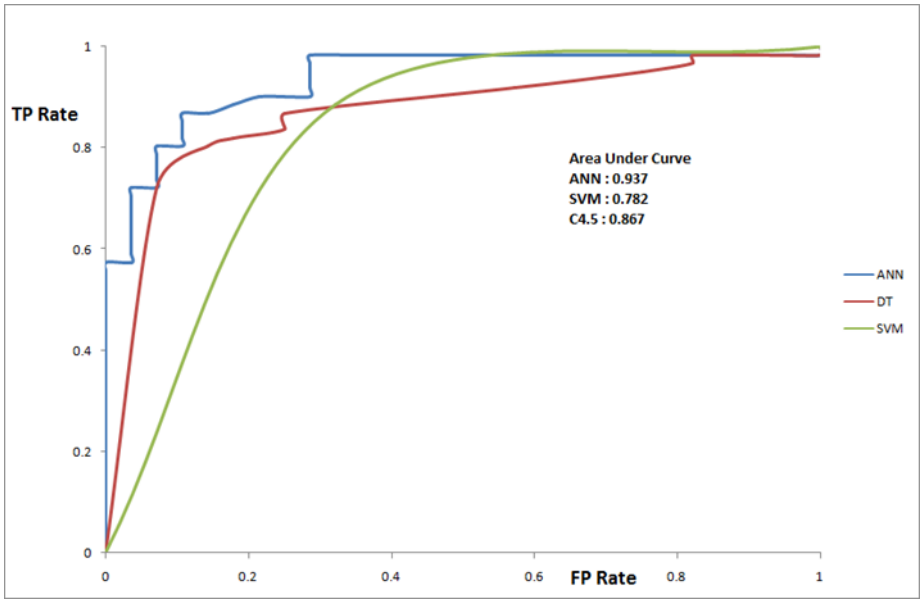


FIGURE 10. The J48 Algorthim, SVM, and ANN ROC Curve

5.1 SCALABILITY ANALYSIS

The scalability of the system was tested under varying traffic loads to evaluate its performance in real-world high-traffic environments. Three traffic scenarios were analyzed:

1. **Low Traffic (100 requests/min):** The system processed requests efficiently with minimal latency (average response time: 50 ms) and no degradation in accuracy.
2. **Moderate Traffic (500 requests/min):** Performance remained stable with a slight increase in latency (average response time: 120 ms). Detection accuracy remained consistent across models.
3. **High Traffic (1000 requests/min):** The system maintained an accuracy above 95%, but the response time increased to 300 ms due to a higher load on the model prediction and database lookup processes.

The results demonstrate that the proposed system is capable of scaling to handle high traffic while maintaining robust detection accuracy and responsiveness.

5.3 COMPARATIVE ANALYSIS

Of the models assessed, ANN proved to be the most effective, achieving the highest classification accuracy of 86.50%, as detailed in Table 1. This result underscores ANN's ability to accurately classify incidents with minimal error, making it highly suitable for high-traffic applications where precision is critical. Although ANN exhibits a higher Mean Absolute Error (MAE) of 1.402, it shows consistency with a Root Mean Squared Error (RMSE) of 0.2971 and a Kappa statistic of 0.6748, as shown in Table 2. These metrics indicate a strong alignment between the model's predictions and the actual outcomes, further establishing ANN as a reliable tool for real-time SQL injection detection. Its performance is especially critical in situations where false negatives may result in serious security breaches.

SVM has an accuracy of 82.00%, although its performance is well-balanced; however, when precision or recall are of primary concern, this model does not support it very well. The moderate computational requirement allows the model to be applicable in medium-scale web applications or systems that have resource constraints. J48, with a precision of 80.90%, is outstanding at recalling valid data at 0.750 and, therefore, can be applied to applications where validating legitimate transactions is essential. Its precision of 0.677 is, however, relatively lower, and this will give a higher rate of false positives that may not be so suitable for high-risk environments.

ANN performs similarly or even better than other models like Random Forests or GBMs in comparison, at least in situations that demand adaptability and scalability. GBMs are very accurate but prone to computational inefficiencies in the high-traffic environment. Meanwhile, ANN processes large amounts of data efficiently, hence it is more suitable for real-world applications.

Practical implications for this study are important in high-traffic web services such as e-commerce platforms, online banking systems, and enterprise resource management tools. ANN, with a high recall of 0.934 and precision of 0.877 in detecting SQL injection, offers robust protection against malicious attacks, thus reducing the risks of financial loss and reputational damage. It also possesses a high AUC value of 0.937, signifying that the model remains stable in detecting threats for most scenarios, and is very effective in dynamic environments of threats.

6. DISCUSSION

Performance analysis using the proposed system that leverages the Malicious String-Matching Algorithm, with ASP.NET, can identify the potential effectiveness in detection and prevention of injection attacks, such as SQL injection, cross-site scripting, and other web-based vulnerabilities. The MSMA has a method of tokenizing the URL, indexing input variables, and then comparing the substrings to malicious data in a pre-stored database. The algorithm first evaluates the URL for variables and input values, extracting and then analyzing substrings for the potential threats. If any match is found with a malicious string known in the indexed database, the URL will be marked as malicious. On the performance side, it showed an accuracy of detecting harmful URLs at 95.54%. This high accuracy results from the strength of MSMA, which checks both encoded and standard URLs, with the protection it provides while the malicious data is obfuscated. Additionally, the types of attacks the system have identified, such as SQL injection, cross-site scripting, remote code execution, and email header injections, improve its versatility and effectiveness.
The indexing mechanism in the database allows for rapid comparison of the input data with the stored malicious patterns, thus making the detection fast and reliable.
The comparison was done between various machine learning models that were being used for SQL injection attacks, and ANNs performed the best with an accuracy of 86.50%. Even though the classification performance was very strong by the ANN model, the overall protection provided by MSMA combined with ASP.NET was significantly better in comparison to others against most of the types of injection attacks, whether encoded or modified input string. It ensured high accuracy of detection with acceptable response times across different variations of traffic loads, and therefore is apt for deployment in the real world, where threats evolve in an ever-changing nature. Scalability also helps it deploy in high-traffic environments, where timely detection and prevention of injection attacks are very crucial for security assurance of a web application.

7. CONCLUSION

This paper introduces two experiments in the context of injection methods through the utilization of URL scanners. Testing the proposed Malicious String-Matching algorithm through the usage of the URL scanning for injection approach achieved a level of accuracy of 95.54%. ML approaches are integrated into an immune vulnerability reduction system to create stronger immune systems that can adjust to constantly changing threats. Lexical features of injection approaches are used for adaptive training to conditions [37]. An Agent-based Vulnerability Response System may be designed to automatically identify, notify, and respond to changes or reconfigurations. In this study, we have concentrated on using the three machine learning techniques—ANN, SVM, and J48—to create a strong security system. The best accuracy rate of 86.50% has been attained by ANN.

REFERENCES

[1] C. Sharma and S. Jain, "SQL injection attacks on web applications. Int. J. Adv. Res. Comput. Sci. Softw. Eng. 7, 24–26 (2017).
[2] B. Mukhtar and M. Azer, "Evaluating themodsecurity web application firewall against SQL injection attacks. In: 2020 15th International Conference on Computer Engineering and Systems (ICCES), pp. 2–7 (2020).
[3] MAlenezi,M., Nadeem, Asif, R.: SQL injection attacks countermeasures assessments. Indon. J. Electr. Eng. Comput. Sci. 21, 1121–1131 (2020).
[4] L., Qian. M. Zhu, Z., Hu, J., Liu, S.: Research of SQL Injection Attack and Prevention Technology. In: 2015 International Conference on Estimation, Detection and Information Fusion (ICEDIF), pp. 303–306 (2015)
[5] M. Horner.,T. Hyslip., SQL injection: the longest running sequel in programming history. J. Digit. Forensics Secur. Law 12, 10 (2017.
[6] L. Ma and C. Gao, “Research on SQL injection attack and prevention technology based on web. In: 2019 International Conference on Computer Network, Electronic and Automation (ICCNEA), pp. 176–179 (2019).
[7] Z. Alwan., and M., Younis, "Detection and prevention of SQL injection attack: a survey”. Int. J. Comput. Sci. Mob. Comput. 6, 5–17 (2017).
[8] C. Ping, "A second-order SQL injection detection method. In: Proceedings of 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), vol. 2018-January, pp. 1792–1796 (2018)," PACIS 2009 Proceedings, p. 16, 2009.
[9] N. Karthikeyan and R. Vivekanandan, "A novel technique to detect and prevent SQL injection attacks using bitap string matching algorithm. High Technol. Lett. J. 27, 252–264 (2021).

[10] J.,Abirami,R., Devakunchari., and C.,Valliyammai, "A top web security vulnerability SQL injection attack—Survey. In 2015 Seventh International Conference on Advanced Computing (ICoAC) (pp. 1-9). IEEE.
[11] C. Sharma. S., C., Jain "Explorative study of SQL injection attacks and mechanisms to secure web application database-A. Int J Adv Comput Sci Appl, 7(3), 79-87.
[12] A., Pramod., A., Ghosh. And A., Mohan,"SQLI detection system for a safer web application. In 2015 IEEE International Advance Computing Conference (IACC) (pp. 237-240). IEEE.
[13] N.,.Antunes, and M. Vieira, "Detecting SQL injection vulnerabilities in web services. In 2009 Fourth Latin-American Symposium on Dependable Computing (pp. 17-24). IEEE, 2009.
[14] T. Pattewar, and H., Patil "Detection of SQL injection using machine learning: a survey. Int. Res. J. Eng. Technol. (IRJET) 6, 239–246 (2019).
[15] M.,Reddy,T., Balamurugan., "Applied machine learning predictive analytics to SQL injection attack detection and prevention. Eur. J. Mol. Clin. Med. 7, 3543–3553 (2020).
[16] F. Hernawan. S., C.,Hidayatulloh, "Hybrid method integrating SQL-IF and Naïve Bayes for SQL injection attack avoidance. J. Eng. Appl. Technol. 1, 85–96 (2020).
[17] D., Chen., Q., Yan. And C., Zhao,".: SQL injection attack detection and prevention rechniques using deep learning. J. Phys: Conf. Ser. 1757, 012055 (2021).
[18] R., K.,Sonakshi, and G. Gopal, "Prevention of SQL injection attacks using RC4 and blowfish encryption techniques. Int. J. Eng. Res. V5, 25–29 (2016).
[19] M. Sood, and S., Singh "SQL injection prevention technique using encryption. Int. J. Adv. Comput. Eng. Netw. 5, 5–8 (2017).
[20] K. Sharma and S. Bhatt, "Efficient method to prevent SQL injection attacks using password encryption. IAETSD J. Adv. Res. Appl. Sci. 5, 90–96 (2018).
[21] M.,Muttaqin,R., "Implementation of AES-128 and token-base64 to prevent SQL injection attacks via HTTP. Int. J. Adv. Trends Comput. Sci. Eng. 9, 2876–2882 (2020).
[22] P. Jayali. S., V.,Chougule "SQL injection detection and prevention using pattern matching algorithm. Int. J. Adv. Res. Comput. Commun. Eng. 5, 145–147 (2016).
[23] M., Kashyape., A., Agrawal. And A., Gahlod,"A hybrid approach for prevention of SQL injection attack using pattern matching Mitali. Int. Res. J. Adv. Eng. Sci. 2, 194–197 (2017).
[24] O.C.,Abikoye,A., Abubakar. A., H., Dokoro, "A novel technique to prevent SQL injection and cross-site scripting attacks using Knuth-Morris-Pratt string match algorithm. EURASIP J. Inf. Secur. 2020(1), 1–14 (2020). https://doi.org/10.1186/s13 635-020-00113-y.
[25] N., Karthikeyan. R. Vivekanandan,"Detection of SQL injection using machine learning: a survey. Int. Res. J. Eng. Technol. (IRJET) 6, 239–246 (2019).
[26] K.,Poulsen, "Guesswork Plagues Web Hole Reporting. https://www.securityfocus.com/new s/346.
[27] O. Shezaf., "Russian hackers broke into a RI GOV website. https://web.archive.org/web/201 10213051033/http://www.xiom.com/whid-2006-3.
[28] K., Ward., Redmond channel partner online: hacker defaces Microsoft U.K. web page. https:// web.archive.org/web/20071223181645/http://rcpmag.com/news/article.aspx?editorialsid= 8762.
[29] P.,McDougall, and G. Gopal, "Prevention of SQL injection attacks using RC4 and blowfish encryption techniques. Int. J. Eng. Res. V5, 25–29 (2016).
[30] S. Lemon, "Mass SQL injection attack hits Chinese websites. https://www.computerworld. com/article/2536020/mass-sql-injection-attack-hits-chinese-web-sites.html.
[31] D., Danchev.,"Kaspersky's Malaysian site hacked by Turkish hacker | ZDNet. https://www. zdnet.com/article/kasperskys-malaysian-site-hacked-by-turkish-hacker/.
[32] BBC NEWS | Business | US man 'stole 130m card numbers'. http://news.bbc.co.uk/2/hi/ame ricas/8206305.stm.
[33] Yap, J.: 450,000 user passwords leaked in Yahoo breach | ZDNet. https://www.zdnet.com/art icle/450000-user-passwords-leaked-in-yahoo-breach/.
[34] TalkTalk gets record £400,000 fine for failing to prevent October 2015 attack. https://web. archive.org/web/20161024090111/https://ico.org.uk/about-the-ico/news-and-events/newsand-blogs/2016/10/talktalk-gets-record-400-000-fine-for-failing-to-prevent-october-2015- attack/.
[35] S. Khandelwal., "Fortnite flaws allowed hackers to takeover gamers' accounts. https://thehac kernews.com/2019/01/fortnite-account-hacked.html.
[36] K., Hu., A survey on SQL injection attacks, detection and prevention. In: CMLC 2020: 2020 12th International Conference on Machine Learning and Computing, pp. 483–488. Association for Computing Machinery, New York (2020).
[37] M., Althunayyan, and M. Saxena, "Evaluation of black-box web application security scanners in detecting injection vulnerabilities. Electronics, 11(13), 2022.
[30] S. Lemon, "Mass SQL injection attack hits Chinese websites. https://www.computerworld. com/article/2536020/mass-sql-injection-attack-hits-chinese-web-sites.html.

Wasit Journal for Pure Science

Journal Homepage: https://wjps.uowasit.edu.iq/index.php/wjps/index
e-ISSN: 2790-5241 p-ISSN: 2790-5233

*Corresponding author: hanoomo9393@gmail .com