International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

IJISAE

ISSN:2147-6799 www.ijisae.org Original Research Paper

Prevention of Website SQL Injection Using a New Query Comparison and Encryption Algorithm

[1]Mahmoud Baklizi, [2]Issa Atoum, [3]Mohammad Al-Sheikh Hasan, [4]Nibras Abdullah, [5]Ola A. Al-Wesabi, [6]Ahmed Ali Otoom

Submitted: 30/10/2022 **Revised:** 19/12/2022 **Accepted:** 02/01/2023

Abstract*:* Nowadays, a web application has become necessary in all organizations. Which deals directly with the databases in which data and information are stored, organized, retrieved, and processed. Therefore, most of its attacks are on databases. Therefore, web applications must be secure enough to prevent access to customs databases, destruction, and theft of bank accounts and transactions. Thus, most SQL injection attacks are carried out through character spacing, as it is the tool used by hackers to find a vulnerability on the web. This paper proposes a new algorithm to prevent hackers from accessing databases early on through the web application without accessing databases. The proposed algorithm is designed to protect the web application from being voluntarily inserted by using a bind parameter, blocking the hacker's address, and rejecting his request when executing the query. Also, this algorithm is designed to work in more than one layer, as it works at the web application and URL levels so that things are sufficiently protected. The comparison was made with the algorithms SQLPMDS, SIUQAPTT, and blind SQL injection, and the results showed that the presented algorithm gave better results based on more than one measure.

Keywords :*SQL Injection, Prevention, Character Spacing, SQLPMDS, SIUQAPTT, Bind SQL Injection.*

1. Introduction

With the continuous increase in the use of the web and its spread to small or large organizations(even a minimum), it is critical to protect the website from threats. The web is a means of accessing the web server that contains any organization's databases and application systems. Therefore, it has become a subject of great interest by hackers who are always seeking to reach any weak point that may be important to them to pounce on the database and thus access the organization's data and destroy or steal data or blackmail [1]. Unfortunately, the damage of SQL injection is one of the most common threats that might jeopardize an organization's reputation and result in data loss. For example, Sony enterprise has been exposed to all its capabilities being hacked, where millions of accounts were damaged. The hack included critical assets, including credit cards, which caused significant financial losses [2-4].

[1]*Computer Science/Network Department, Faculty of Information Technology, Al-Isra University, Amman, Jordan,* mbaklizi@iu.edu.jo
[2]*Sofware Engineering Department, Faculty of Information Technology, The World Islamic Sciences and Education, Amman, Jordan,* issa.atoum@wise.edu.jo
[3]*Computer Science Department, University of Petra, Amman, Jordan,* malsheikh@uop.edu.jo
[4]*School of Computer Sciences, Universiti Sains Malaysia (USM), Penang 11800, Malaysia; Faculty of Computer Science and Engineering, Hodeidah University, Hodeidah P.O. Box 3114, Yemen,* neprasf@gmail.com
[5]*School of Computer Sciences, Universiti Sains Malaysia (USM), Penang 11800, Malaysia;Faculty of Computer Science and Engineering, Hodeidah University, Hodeidah P.O. Box 3114, Yemen,* ola.wosabi@gmail.com
[6]*Faculty of Science and Information Technology, Irbid National University, Irbid, Jordan,* aotoom@inu.edu.jo

Protecting the web from hackers is one of the significant challenges facing institutions, banks, and other essential organizations interested in protecting their data. However, there is nothing to prevent the penetration of these sites for several reasons entirely. First, the website's design differs from one institution to another. Great reliance, in this case, depends on the programmer's knowledge of the site about the threats that may result from gaps in the code. For example, if code is not written correctly, hackers might find penetration points. Next, applications are used in different implementation layers, starting from the web client to the server and then to the databases between these layers. Finally, we do not neglect the Internet, an open place for hacking if it is not adequately protected and tight [5].

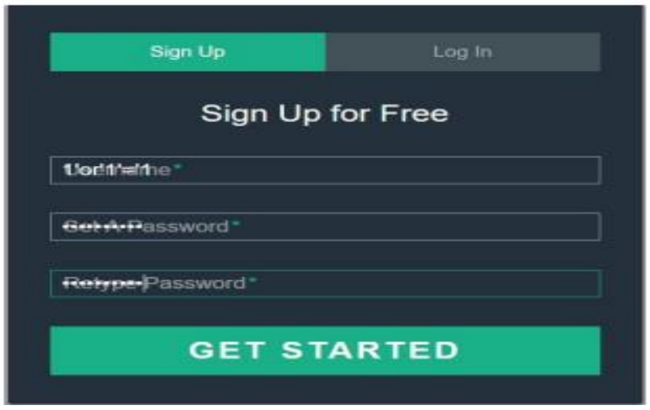
In the past, most threats were interested in authenticating the user name and password using many tools. However, we focus here on the web and databases following many previous works [6]. Vulnerabilities are known through random commands that indicate to the hacker the presence of weaknesses in the sites. These commands are called character spacing, as most hackers rely on experimenting with these commands [7]. This paper will focus on protecting the web and data before the hacker reaches the databases. Therefore we do not allow him to access the databases until after ensuring the integrity of the written query, as we ensure that there are no commands that allow the hacker to access the databases and manipulate sensitive data.

This paper is organized as follows. Section two explains the SQL Injection Attack. Section three displays the prevention of SQL injection-related work. Section four discusses the methodology of this paper. Section five discusses the implementation and evaluation results. Finally, Section six provides the conclusions.

2. SQL Injection Attack

One of the most crucial things in attacking requires finding databases than finding a loophole in web applications. Therefore, hackers often target databases with injections to steal or destroy data[8]. The principle of SQL injection is with the query. A query, in turn, deals with the databases. Therefore hackers have full knowledge of how to build the query that deals with the databases. As a result, they could find weaknesses that they can exploit to reach their aspirations [9]. This section provides a simple example of how attackers could reach a webpage as an authentic user without real credentials.

Usually, the query statement is written from a set of words and symbols that allow it to use the web and databases. For example, if we want to access the login screen, which usually uses a username and password to be compared directly to the values stored in the database tables. Figure 1 shows a login for a specific employee [10].

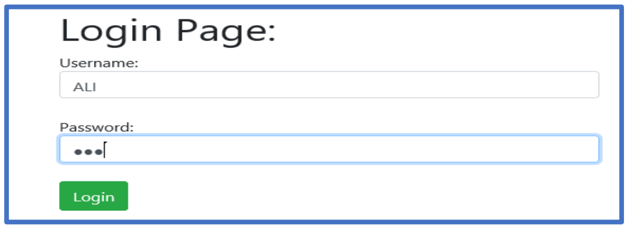


Select username from user where empn=" ALI"

Fig.1 :Login Screen

The user can enter if the user name is correct and the password is confirmed by comparing it with the values stored in the databases.

Otherwise, the user cannot enter, showing a message to reenter the username and password. The following example shows the process of querying typically for a value stored in databases However, hackers can use the character spacing through which they always get correct results regardless of the actual query sentence if the result is wrong. The hacking activity allows hackers to access the databases and retrieve the information they want. See the following example:


Login Page:

Username: ALI

Password: •••

Select username from user where empn=" ALI1" or 1=1

Fig 1: Prevention Login Screen

Note that even if the value of 'empn' is wrong, it does not allow it to retrieve data, but the presence of the character spacing "or" made the query statement as a total result correct, and therefore it can retrieve data as an authentic user. Therefore, using SQL injection was necessary to prevent these attacks, which hackers cause. Furthermore, it protects the web and databases from any malicious attacks that may cause physical and moral damage to the web application in organizations.

3. Related Work

The danger of SQL injection that web applications are exposed to has increased rapidly and dramatically recently. Therefore, this risk had a strong motivation by researchers to protect web applications from these attacks to preserve data and databases. Researchers have found many ways to repel these attacks, such as [11, 12]. However, the question remains whether these methods and algorithms are sufficient for the purpose, which ones are best for protecting web applications, and which are suitable for application in the concerned institution. Furthermore, some attack prevention solutions of SQL injection may be expensive and need a strategy and budget for the organization's infrastructure to operate effectively. Of course, small and medium enterprises cannot bear these financial burdens [13]. Therefore, some algorithms have appeared that do not depend on high costs, and small and medium enterprises can also use and apply them.

Nikto, SQLMAP, relies on its work on the vulnerability scanner, which bears the name SQLMAP, which was initially intended for use in hacking. This method provided alerts on defects only[14]. One of the disadvantages of this algorithm is the theft and misappropriation of data and information. SQL injection with (SVM): In this algorithm, the researchers suggested a design that depends on the queries if they are natural or harmful, as they relied on trained questions to make sure whether SQL injection is present in the queries or not, and the results were satisfactory at the time. However, this algorithm was unsuitable for large data volumes [15]. The SQL Pattern Algorithm (SQLPMDS) repels SQL injection attacks. It relies in its work on storing well-known SQL statements, which according to what the algorithm was designed with, are present in the databases. When the query reaches the databases, it is checked and compared to see if this attack is malicious and is caused by a SQL injection or a harmless query. Unfortunately, if the program cannot identify the attack, this method has reached the database, and the hacker can inject SQL injection and access and destroy the databases. Also, this algorithm does not deal with cases except insert, and this feature is critical in dealing with databases [16].

SIUQAPTT is an algorithm designed to block SQL injection attacks as an acceptable solution to check only a specific query and static for all query strings. This algorithm depends in its work only on the query that contains the word "WHERE" the algorithm considers anything written after this sentence as an attempt to inject SQL and excludes it to avoid access to the databases. This algorithm is acceptable to a certain extent. However, there are more things than the word "WHERE" that hackers use to access a loophole that enables them to access databases that were not included in this algorithm. Furthermore, the algorithm did not also deal with insert cases, although most insert statements do not have a "WHERE" clause [17].

Blocking Blind SQL injection is utterly dependent on coding and prevents attacks by blocking for IP, but this algorithm is only tested on the login page; see Figure 2, which shows the prevention login screen for blind SQL injection.

A few websites were used to check their results. This algorithm displays an alert message if the hacker tries to enter incorrect values and blocks his IP address. Nevertheless, as we mentioned earlier, this algorithm relied only on coding, and many things were not taken into account because it is a powerful source for hackers to access databases or data such as union as well as insert and not a lot of character spacing. Moreover, their study covered the login screen and neglected the rest of the pages in the web application [18].

By finding a weak spot in the web, the attacker will try injecting SQL commands into the web to be executed in the particular database. As a result of the attacker's experience, it is not difficult for him to implement a random SQL query in the database through the web available to him. With SQL injection, attackers could obtain powers that entitle them to enter and deal with the databases and then change and sabotage them or retrieve information from the databases such as Credit cards, accounts, and banking transactions.

The research gap is that many proposed algorithms were designed to prevent SQL injection, but it has many limitations. For example, where it did not use all the character spacing and did not deal with the state of the insert when adding data to the databases, only one layer was taken into account to detect the query. Moreover, many researchers neglected a critical URL layer, a weak point through which hackers can access the databases. Finally, most of the mentioned algorithms only deal with the login screen, and only a few queries are checked. These parameters were a strong incentive for an algorithm to process these parameters.

4. Methodology

Protecting the web from hacker attacks has become an important matter that requires a great effort to think about how to protect data and to suggest new ways to keep pace with the massive development by attackers to access databases and hack and access sensitive data. Therefore, protecting websites and databases requires complete knowledge of how attackers work to gain access to the data.

Most hackers are interested in collecting information that they think may be valuable data for them to rely on to find a weak point before doing SQL injection. For example, the behavior of the system and how it works require time from the attacker and much analysis to eventually reach a weak point so that, in this case, he can carry out his attack through it. If an attacker finds a weak point, the attacker may be able to access the user's permissions and then obtain sensitive data and information. For example, the issue of URL attacking is explained over this site http://testphp.vulnweb.com/artists.php?artist=1 [19]. The database data was accessed, and the admin's powers were also obtained, allowing him to take sensitive data from the databases.

Therefore, this paper proposes the following model that protects the first interface of the hacker from accessing the databases and gives protection to the data stored in the databases. Look at the following figure 3, which illustrates the proposed methodology.

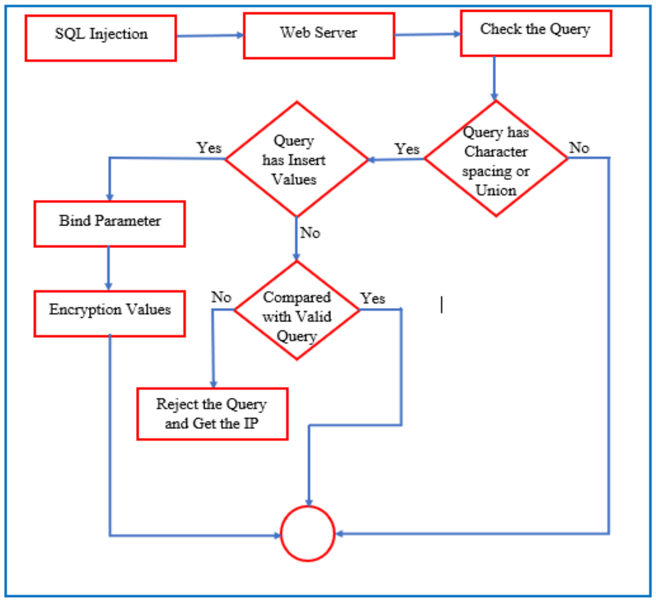


Fig 2: the proposed methodology.

Any injected query passes through the web server, an intermediary between the user and the databases, and commands are executed directly on the underlying databases. Of course, the web server receives any query, regardless of its source, from a real user to carry out a routine, harmless process or through an attacker whose goal is to sabotage and access the database.

The proposed approach connects the webserver to ensure an established connection exists as the request to execute a query is made from the private server on the web. Figure 4, explains the client and server communication process.

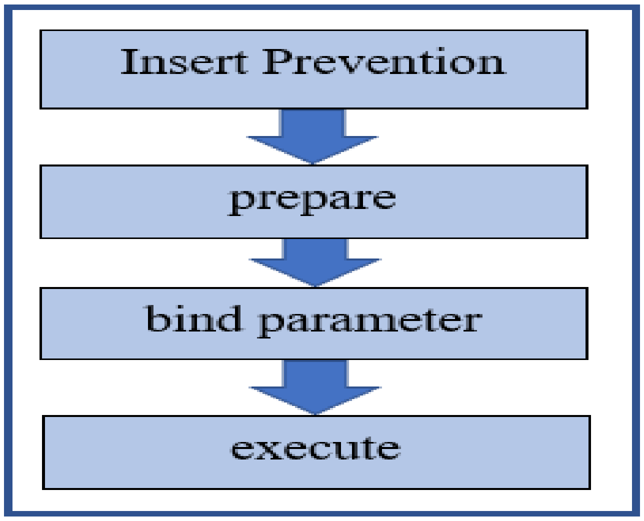


Fig. 3:communication between the client and the server.

After the connection between the client and the server, the server is ready to receive the query sent from the web after identifying the socket address, which in turn contains the IP address for the server- side side side and port number. See the following figure that shows the pseudo code to connect the client to the server and send data between clients. and server

1- //server side
2- initialize the Server Socket //ServerSocket Socket = new ServerSocket (80);
// server will listen for a connection from a client on port 80
3- the Server relies into a Waiting State// link = Socket.accept();
4- prepare Input and Output Streams// getIStream(),getOStream
5- Send and Receive the query// input.nextLine();
6- After completing Close the Connection
7- //Client side
8- Create a Connection to Server side // Socket (host, PORT_number)
9- prepare Input and Output Streams // link. getIStream;
10- Step 3: Send and Receive query// input. nextLine ();
11- Close the Connection// link. close ();

We refer to what was previously mentioned: the hacker relies on character spacing randomly in his attacks to find a loophole that helps him in the penetration process and access to databases. Therefore, this paper aims to prevent any query from executing except after ensuring that it is free of this character to ensure that the hacker never accesses the databases.

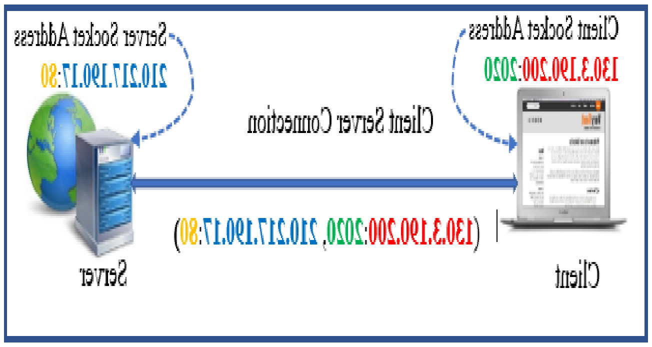


Fig. 4:insertion prevention

When a server receives queries from a web application, the proposed algorithm compares the received queries with those stored in the web server. Stored queries are nonmalicious and authentic, recognized by the web or network administrator, and allowed to access databases and retrieve data. Such equerries are executed directly and are considered non-harmless. However, it is neglected if it does not exist and contains the texts that the hacker usually uses. It is also made sure that they do not contain any insert clause. The query is canceled if it contains the insert command, and the IP is sent to the malicious block list. Look at the following that shows the comparison process.

```
1-  Query in server-side //
2-  Check the query
3-  if the query does not have a character spacing
4-  the query is executed, and it accesses the database
5-  if the query has a character spacing
6-  the query is not allowed to execute and reject
7-  if the query has an insert process
8-  the bind parameter is implemented
```

Finally, if the query contains an insert, a Bind parameter is applied to protect the data entered into the databases. Then, a Mark is placed to protect data from hackers. The bind parameter is received in the second stage, where the real value is entered into the databases. The last stage is the query execution, where it will be entered at this stage to the database and work. However, usefully when entering a single quote, for example. This scenario does not receive it and transfer it to a back-slash. Therefore we know that a hacker is attempting to access the databases.
Consequently, the process is canceled, and the IP address of the sending party is taken. The proposed Biber is designed as a web application connected to the database to access the database. Look at the following figure 5, which shows insertion prevention

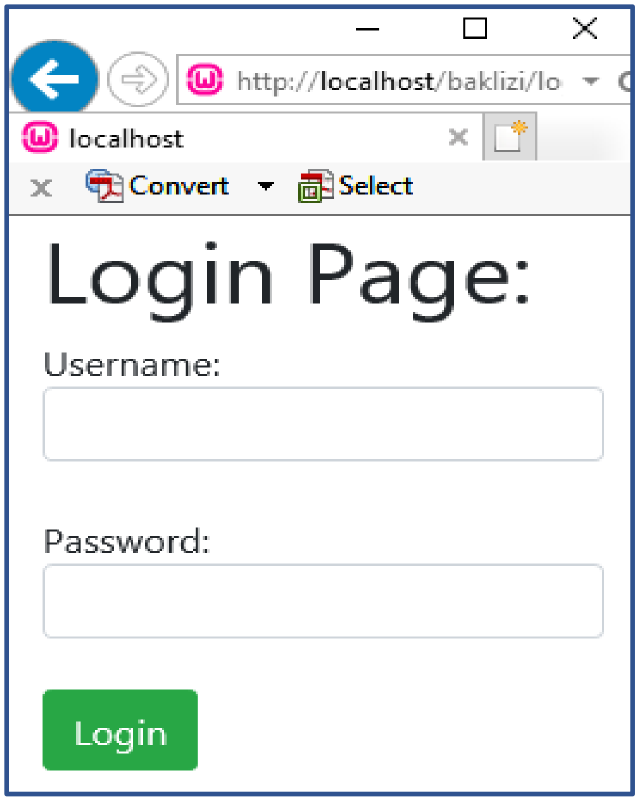


Fig 5:insertion prevention

The fields will be entered through the web application, and when a user clicks on insert data, it will be stored in the database where the database used is Mysql. The following figure shows the structure of the database used. The following is pseudo-code for insert prevention.

```
1-  The (host, database user, database password, and
    database name) have been initialized.
2-  //Connect web server to baklizi database using
    MySQL method.
    Mysqli($host,$dbuser,$dbpass,$dbname);
3-  The username is initialized and waits to execute
4-  //The password is initialized and encrypted using the
    md5 method
    $password= (md5($_POST['password']));
5-  The query is waiting to be prepared
6-  "Insert into users (username, password) values
    (??)");
7-  The query is prepared
8-  // bind parameter is invoked
    bind_param ('ss', $user, $password);
9-  execute (); \\ execute method is run
```

In this case, any unauthorized person has been prevented from accessing database data if it aims to sabotage or steal, using the previous algorithm that goes through more than one stage to preserve the data and prevent SQL injection. Before accessing databases, it is based on preventing the most common type used by hackers to access databases: character spacing.

5. Implementation, Evaluation, and Discussion

Protecting the web from hacker attacks has become necessary to protect databases and sensitive data. So it was necessary to propose an algorithm to protect the data and databases before the hacker accessed them. Figure 5 shows the algorithm's work to prevent SQL injection before accessing the databases using the character spacing mentioned in Table 1, which shows the character spacing [12].

Table1 : Character Spacing

character	Task
--	Line comment
"or"	String
/*---*/	Many lines comment
+ ,||	Concatenate
?php1=abc&ad=mar	URL
'0:0:10'	Wait for the time delay

xIn this part of the paper, we will show the implementation of the algorithm in detail, which shows the process of filtering a query to accept or prevent a query from executing. Many tools were used to help us build the application to fully implement the algorithm, including MySQL databases, PHP, and PhpMyAdmin [20].

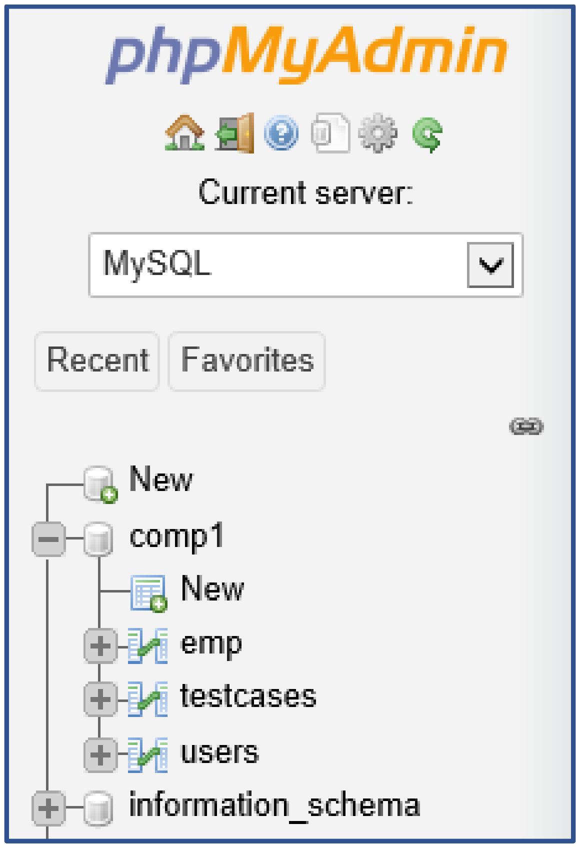
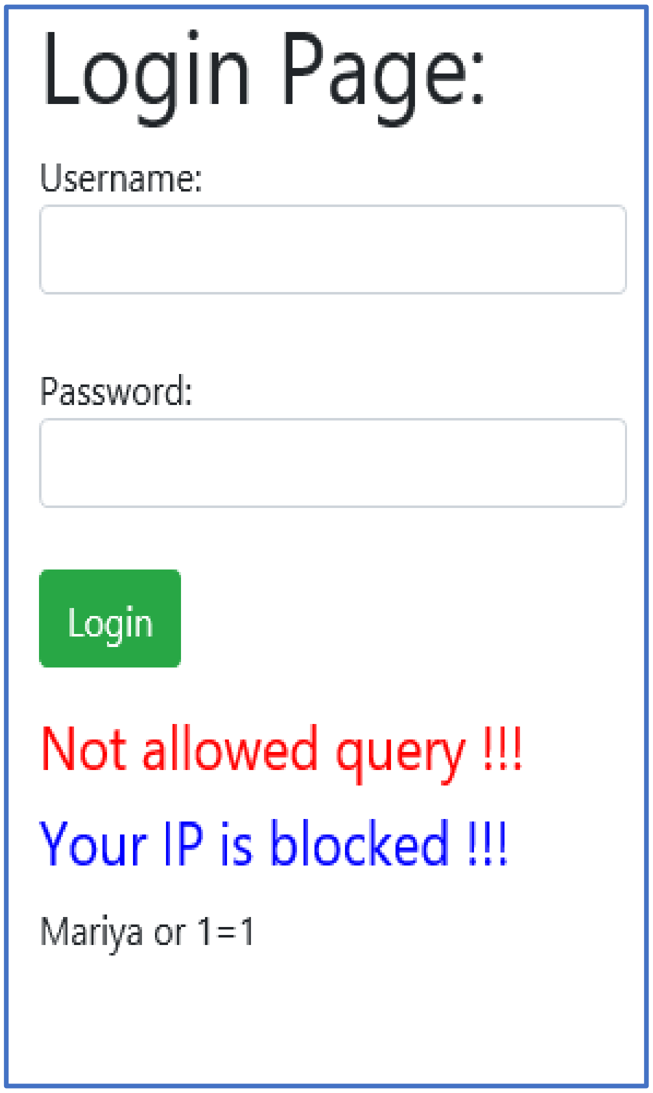


Fig. 6:the structure of the database.

A database has been created on the server to contain all the tables needed to implement the algorithm called Comp1. Look at figure 6 shows the structure of the database.
The proposed paper is concerned with preventing SQL injection through the web application form and the URL of the same site so that the protection is on more than one layer to ensure that the hacker does not reach in any way he thinks. In this part, we will show protection from the same web application and the URL layer.
When the database is ready to receive data from any source, the user can start from the well-known login screen, which is the most crucial part of any web application. For example, look at Figure 7 which shows the home login screen.
n the typical situation, the entry process should be with the user name and password so that if the information is correct, the web and its web-related pages are entered after accessing the databases to ensure that the user name matches to allow him to enter. However, if the hacker uses character spacing, the algorithm stops and never allows

Look at the following example; if the value entered in the password is Mariya, if it is compared to the databases and exists, it is executed typically. On the contrary, the password is incorrect if it is not present. However, if a hacker writes a query from two or more parts; for example, one of the character spacing, let it be OR, then if the result of the Or is healthy, this is enough to execute the query.



Not allowed query !!!

Your IP is blocked !!!

Mariya or 1=1

Fig. 7:home login screen

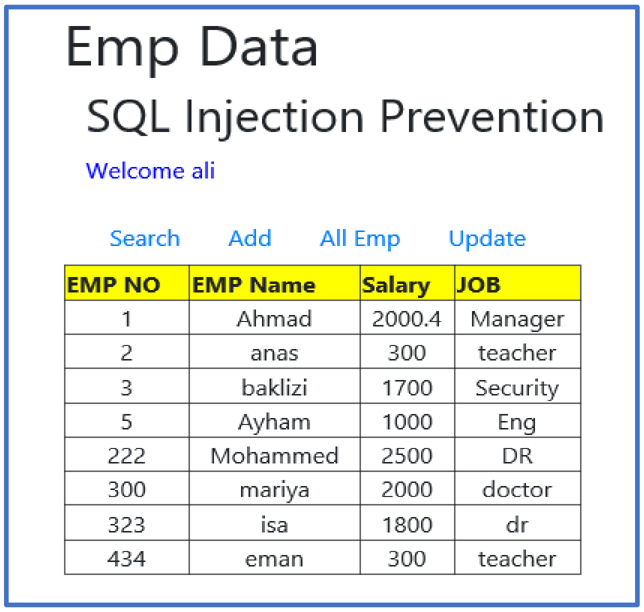
it to operate. What distinguishes character spacing is that it is mainly used to give a valid result regardless of the first part of the query.
See the following example, which shows that the service gives true even if the password is wrong.
Look at Figure 8, which shows the process of preventing SQL injection using the password field

Mariya or 1=1

Fig. 8: preventing SQL injection using the password field

It is not allowed to access because it contains character spacing of type OR because it is considered SQL injection. The algorithm is also concerned with the letter case of writing the command. It can prevent the query from being executed, whether it is a capital letter or a small letter, for example, "or" or "Or." In this case, the hacker was wholly prevented from accessing the data.

If the user name and password are correct, the site enters the user into the web to perform the operations of viewing, searching, adding, and modifying. These operations are considered the central part of any site. For example, look at figure 9, which shows what basic operations there are on the web.



EMP NO	EMP Name	Salary	JOB
1	Ahmad	2000.4	Manager
2	anas	300	teacher
3	baklizi	1700	Security
5	Ayham	1000	Eng
222	Mohammed	2500	DR
300	mariya	2000	doctor
323	isa	1800	dr
434	eman	300	teacher

Fig. 9:basic operations on the web

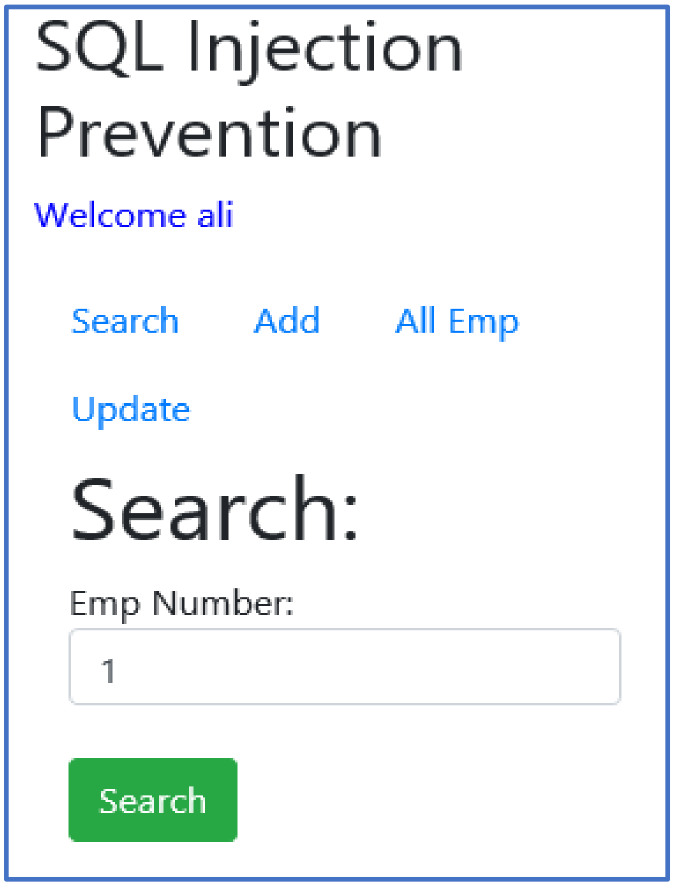
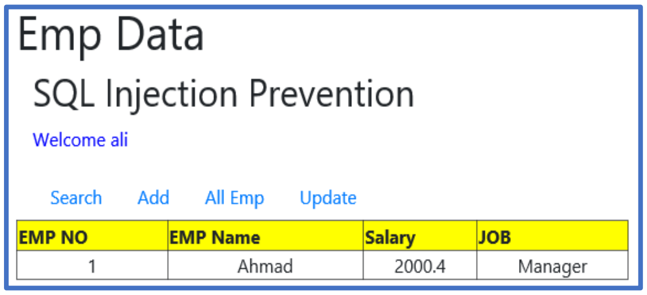


Fig 10:search page

This part shows examples of the process of preventing SQL injection for search, modification, and addition.

In the search process, the employee who holds the number one when inquiring about typically shows all its contents if his number is correct; it is in the databases, where the result appears as figure 10 from the search page.

When a user clicks on search, the employee's information with the number 1 appears, as shown in Figure 11.



Emp Data

SQL Injection Prevention

Welcome ali

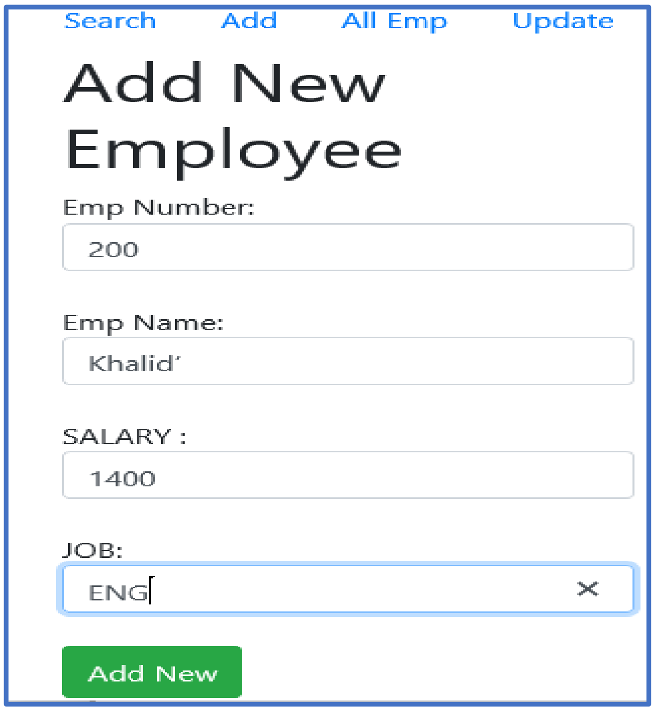
Search Add All Emp Update

EMP NO	EMP Name	Salary	JOB
1	Ahmad	2000.4	Manager

Fig 11: Search Results

As shown, all data related to employee number 1 appeared. However, if there was a breach, the web did not allow it to perform the query we mentioned earlier. For example, look at Figure 12 when using character spacing –.

The insertion process is considered one of the dangerous processes through which the hacker can exploit it in the penetration process. Therefore, what we talked about was applied in the methodology part, which is the bind parameter, where this process is a source of strength for data when it is added to databases. For example, see Figure 13, which explains the process of insertion prevention.



Search Add All Emp Update

Add New Employee

Emp Number: 200

Emp Name: Khalid'

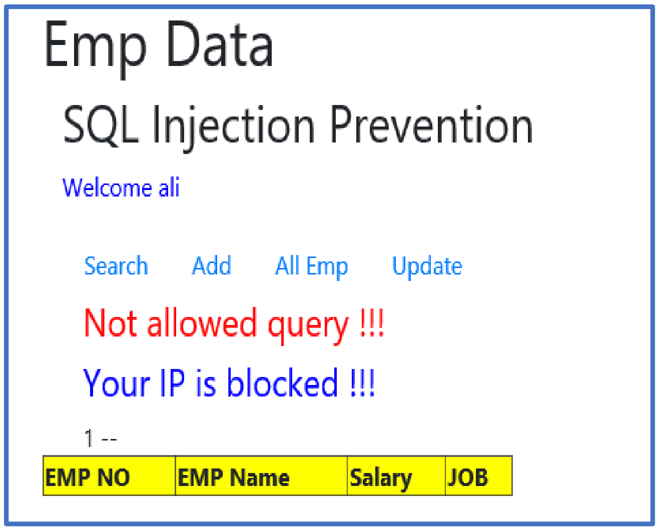
SALARY : 1400

JOB: ENG

Fig 12: Injection Character Spacing

We note from the previous figure that the insert operation of the four fields will be stored inside the databases, but after applying the bind parameter

On the four fields, look at the emp Name field, where a single quote was added, and it is one of the codes used in the insertion process by hackers. In this case, the web will allow it to be stored, but at the same time, the form of content in the databases will change to indicate the presence of a hacker. At the same time, if he wanted shows the data in the databases after applying the bind parameter



Emp Data

SQL Injection Prevention

Welcome ali

Search Add All Emp Update

Not allowed query !!!

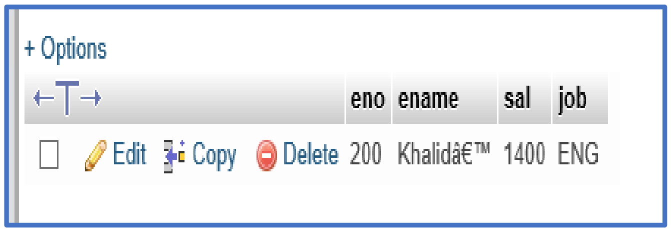
Your IP is blocked !!!

1 --

EMP NO	EMP Name	Salary	JOB

Fig. 13: Insertion Prevention

to retrieve this information, it would not appear to him in the future because the input value is different from the stored value. Figure 14

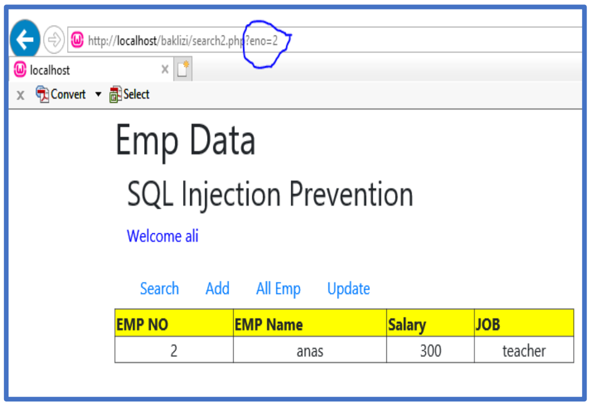


	eno	ename	sal	job
Edit Copy Delete	200	Khalidâ€™	1400	ENG

Fig. 14: Results after Binding

We note from the above that the algorithm was able with high efficiency to prevent SQL injection during the use of the web application form, as well as from the hacker from accessing the databases. The algorithm is also designed and implemented to work not only in one layer. It works in the URL layer, which is available to everyone, where any hacker can access the URL if he knows it or searches for it, so this layer is vital to protect it and prevent SQL injection through it. For example, see Figure 15, which shows the process of using a query through the URL so that the result appears naturally and correctly as if it was executed through the web application form

.



Emp Data

SQL Injection Prevention

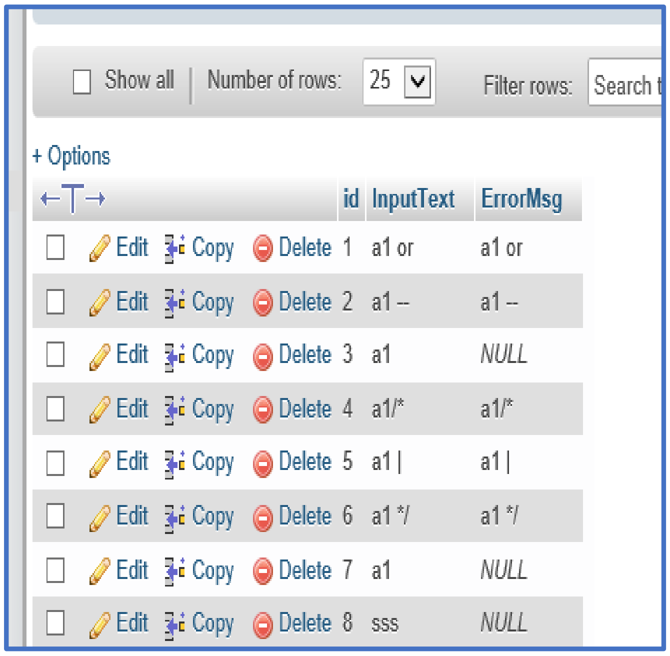
Welcome ali

Search Add All Emp Update

EMP NO	EMP Name	Salary	JOB
2	anas	300	teacher

Fig. 15: the process of using query through the URL

after typing the query in the URL, it is clear that ‘eno’ = 2, and all the fields related to the employee with the number 2 will appear. However, if the query contains the following character spacing, e.g., ‘Or,’ the query will not allow execution after filtering and comparison. See Figure 16, which shows SQL injection prevention via a URL.



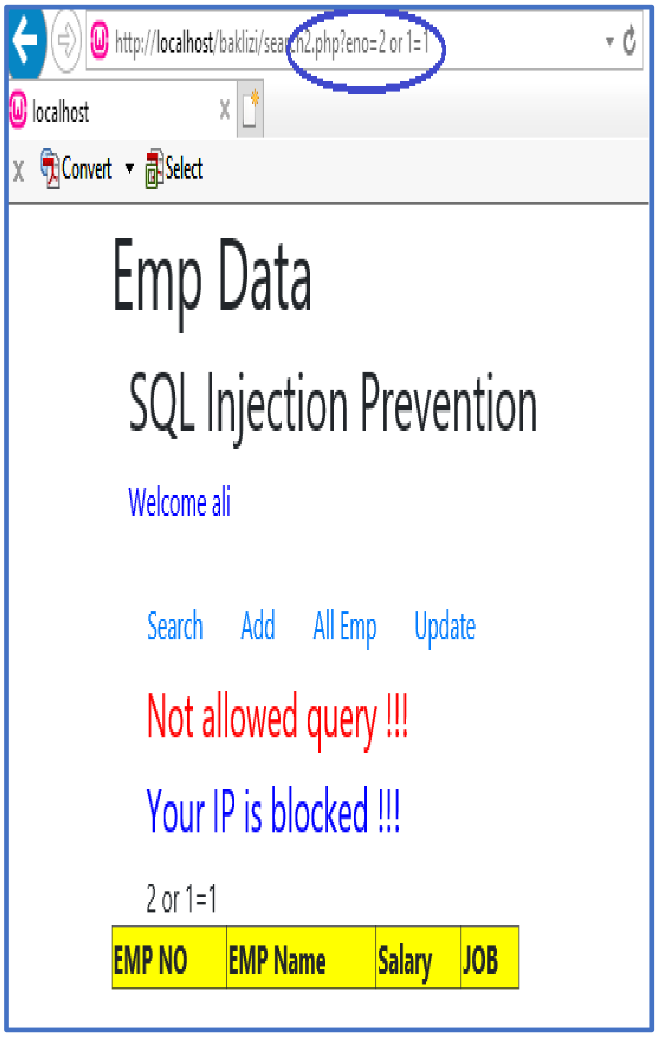
	id	InputText	ErrorMsg
Edit Copy Delete	1	a1 or	a1 or
Edit Copy Delete	2	a1 --	a1 --
Edit Copy Delete	3	a1	NULL
Edit Copy Delete	4	a1/*	a1/*
Edit Copy Delete	5	a1 |	a1 |
Edit Copy Delete	6	a1 */	a1 */
Edit Copy Delete	7	a1	NULL
Edit Copy Delete	8	sss	NULL

Fig 16: URL prevention Results

We note that the query was not allowed to execute or access the databases, although the query is correct in the two parts. The first part has an employee with the number 2, and the second part is correct. Thus, the algorithm prevents SQL injection by more than one layer to protect the databases before accessing them.

In this paper, 200 queries were examined to evaluate the algorithm's results; 140 queries contained SQL injection, and 60 queries did not contain SQL injection. Where it is stored in the query databases that contain SQL injection by showing the entire query clause for future use by similar systems to benefit significantly from the hackers' thinking in SQL injection

Also, queries not containing SQL injection are stored in the databases and marked as Null. The following Figure 17 shows samples of the results that are stored in the databases for both cases.



Emp Data

SQL Injection Prevention

Welcome ali

Search Add All Emp Update

Not allowed query !!!

Your IP is blocked !!!

2 or 1=1

EMP NO	EMP Name	Salary	JOB

Fig 17: samples of the results that are stored in the databases

Figure 18 shows the number of queries used in parsing and evaluating the algorithm and the number of malicious queries out of the total number recognized by the algorithm.

The table compares the proposed algorithm with SQLPMDS, SIUQAPTT, and Blind SQL algorithms. We note that the proposed Maria algorithm deals with all character spacing, giving a solid rudder line from any expected attack by hackers using one character spacing. Blind SQL Injection and the proposed algorithm are concerned with its work to resist the hacker and prevent him from accessing the database before reaching it. This technique dramatically reduces the chance of injection.

All of the mentioned algorithms do not treat the insert case except the proposed algorithm, which deals with the case of the insert as a layer alone due to the Bind Parameter's design. All algorithms alert the hacker that the query is not allowed.

Blind SQL Injection and the proposed method block the IP address coming from the hacker. Finally, the proposed algorithm was used in the screening of most of the web application screens.

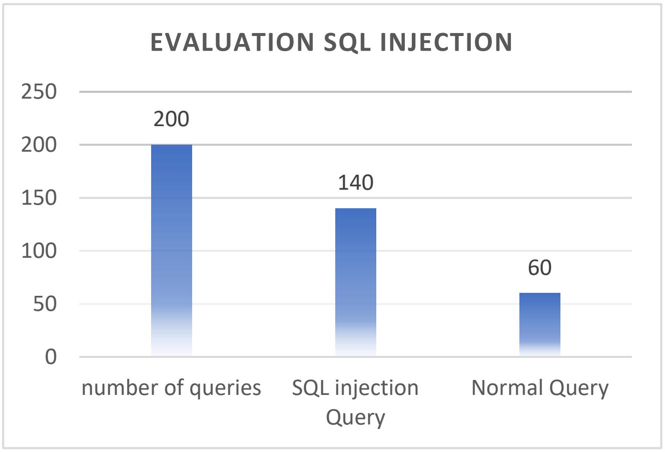


Fig. 18: The proposed Algorithm Result

We note from the above that the proposed algorithm achieved better results than the comparison algorithms. See Table 2, which shows the proposed and current algorithms' comparison results.

Table 2:comparison results.

Algorithm name	SQLPMDS	SIUQAPTT	Blind SQL injection	MARIA
All character spacing	NO	NO	NO	yes
Protect before Access Database	NO	NO	YES	yes
Cover Insert function	NO	NO	NO	Yes
Notify the hacker	YES	YES	YES	yes
Get the IP Address	NO	NO	YES	yes
Number of screen test	Command screen	Command screen	One test screen	Most web application screen

Figure 19 shows the number of queries used in the check between the proposed and the comparison algorithms—the all-out. For example, the attack was 30 in SQLPMDS, while the proposed algorithm used 200.

Based on the apparent results, the number of tests performed on all the algorithms compared them. According to the chart shown.

We note that the Maria algorithm is the best among them as an algorithm to protect the web application from SQL injection

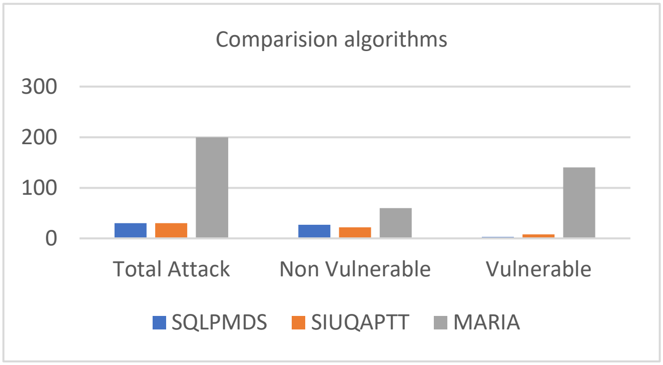


Fig 19: number of queries used in the check between the proposed algorithm and the comparison algorithms

6. Conclusion

SQL injection is one of the most scathing attacks issued by hackers, harming web applications and databases. An SQL attack is a significant cause of data destruction and bank and credit card transaction theft. Therefore, protecting the web and databases from SQL injection has become a big challenge for large, medium, and small organizations. Therefore, it was necessary to think of ways to protect the web from these attacks to protect the data of any organization that uses the web. This paper proposes an approach to protect the web inexpensively that is suitable for organizations regardless of their size. This algorithm depends in its work on the principle of preventing SQL injection at an early stage before accessing the databases and on two layers; the web layer and the URL layer. After the query reaches the server, the query is filtered based on the presence of character spacing to be ignored, rejected, and blocked for any query containing this character. Also, if the query contains the insert command, a bind parameter is set to protect the data entered into the databases. The algorithm has been implemented for the main login screen of the site and other screens related to the same web application. The algorithm was examined by more than 200 queries spread over the web and URL layers. Results were compared with SQLPMDS and SIUQAPTT, proving that the proposed algorithm gave more scalability and speed to protect and detect various attacks.

References

[1] Bayyapu, N., *SQL Injection Attacks and Mitigation Strategies: The Latest Comprehension*, in *Advances in Cybersecurity Management*, K. Daimi and C. Peoples,

Editors. 2021, Springer International Publishing: Cham. p. 199-220.

[2] Chen, D., et al., *SQL Injection Attack Detection and Prevention Techniques Using Deep Learning.* Journal of Physics: Conference Series, 2021. **1757**(1): p. 012055.

[3] Marashdeh, Z., K. Suwais, and M. Alia. *A Survey on SQL Injection Attack: Detection and Challenges*. in *2021 International Conference on Information Technology (ICIT)*. 2021.

[4] Latchoumi, T.P., M.S. Reddy, and K. Balamurugan, *Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention.* European Journal of Molecular & Clinical Medicine, 2020. **7**(2): p. 3543-3553.

[5] Voitovych, O.P., O.S. Yuvkovetskyi, and L.M. Kupershtein. *SQL injection prevention system*. in *2016 International Conference Radio Electronics & Info Communications (UkrMiCo)*. 2016.

[6] Shanmughaneethi, V., et al., *SQLIVD - AOP: Preventing SQL injection vulnerabilities using aspect oriented programming through web services*. Vol. 169. 2011. 327-337.

[7] Lu, D., et al. *A GAN-based Method for Generating SQL Injection Attack Samples*. in *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*. 2022.

[8] Nikita, P., Fahim, and S. Soni, *SQL Injection Attacks: Techniques and Protection Mechanisms.* International Journal on Computer Science and Engineering, 2011. **3**.

[9] Singh, N. and P. Tiwari. *SQL Injection Attacks, Detection Techniques on Web Application Databases*. in *Rising Threats in Expert Applications and Solutions*. 2022. Singapore: Springer Nature Singapore.

[10] Kar, D. and S. Panigrahi, *Prevention of SQL Injection attack using query transformation and hashing*. 2013. 1317-1323.

[11] Raut, S., et al., *A Review on Methods for Prevention of SQL Injection Attack.* International Journal of Scientific Research in Science and Technology, 2019: p. 463-470.

[12] Kini, S., et al. *SQL Injection Detection and Prevention using Aho-Corasick Pattern Matching Algorithm*. in *2022 3rd International Conference for Emerging Technology (INCET)*. 2022.

[13] Harefa, J., et al., *SEA WAF: The Prevention of SQL Injection Attacks on Web Applications.* Advances in Science, Technology and Engineering Systems Journal, 2021. **6**: p. 405-411.

[14] Ojagbule, O., H. Wimmer, and R.J. Haddad. *Vulnerability Analysis of Content Management Systems to SQL Injection Using SQLMAP*. in *SoutheastCon 2018*. 2018.

[15] McWhirter, P.R., et al., *SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel.* Journal of Information Security and Applications, 2018. **40**: p. 199-216.

[16] Chaki, S.M.H., M. Mat Din, and M. Md Siraj, *Integration of SQL Injection Prevention Methods.* International Journal of Innovative Computing, 2019. **9**(2).

[17] Maheshwarkar, B. and N. Maheshwarkar, *SIUQAPTT: SQL Injection Union Query Attacks Prevention Using Tokenization Technique*. 2016. 1-4.

[18] Binu, S. and A. Albert, *Proposed Method for SQL Injection Detection and its Prevention*. 2018.

[19] Aljebry, A.F., Y.M. Alqahtani, and N. Sulaiman. *Analyzing Security Testing Tools for Web Applications*. in *International Conference on Innovative Computing and Communications*. 2022. Singapore: Springer Singapore.

[20] Yenduri, R. and M. Al-khassaweneh. *PHP: Vulnerabilities and Solutions*. in *2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*. 2022.