International Journal of Scientific Research and Modern Technology (IJSRMT) ijsrmt.com
Volume 4, Issue 1, 2025
DOI: https://doi.org/10.5281/zenodo.14960117

Enhancing Web Security: A Comprehensive Approach to Detect and Prevent SQL Injection Attacks through Innovative Query Comparison and Encryption Algorithms

Md. Monowar Hossain[1]; Md. Abul Hasnat[2]; Md. Shahidul Islam[3]

[1] Hamdard University Bangladesh

Publication Date: 2025/02/25

Abstract

In the modern world, web apps are now essential to meeting the daily needs of every company. Databases are used by these applications to store, organize, retrieve, and process data and information. The bulk of its attacks are therefore focused on databases. The frequency of website attacks and the compromise of people's private data are rapidly rising. Since the advent of social networking and e-commerce, web security has gained popularity due to the prevalence of assaults like spam and phishing. For this reason, web applications must be securely designed to prevent unauthorized access to customer databases, bank accounts and transactions are not intercepted, and information is not destroyed or stolen. This paper presents a novel algorithm for website attacks that also stops hackers from gaining early access to databases through the web application without actually accessing the databases. The suggested algorithm uses prevention techniques, blocks the hacker's address, rejects the hacker's request when the query is executed, and updates security often to prevent unauthorized access to the web application. To ensure that everything is adequately safeguarded, this algorithm is also made to operate in many layers, working at the URL and web application levels. Research was conducted to enhance web software security, and a defense system that guards against SQL Injection was created. The developed software creates a protection mechanism using PHP, JavaScript, and regular expression, a formal language theory. This solution gives users a way to secure their web applications from potential attacks by defending against SQL Injection vulnerabilities in web resources.

Keyword: *Web Application, SQL, PHP, SQL injection Attack, Prevention, Web Security.*

I. INTRODUCTION

Web applications face growing threats, and it's important to understand common attacks like phishing and service denial. Phishing and email spamming are simple tricks used in social engineering. With the fast growth of Web 2.0 tech, network applications are now essential in daily lives. However, this progress brings more challenges to web applications. It's important to remain informed about these risks. There are many ways for constructing and executing SQL statements using different web programming languages (such as PHP, java)[1]. Developers often create SQL statements by combining strings provided by users on a web page. Because there are many different SQL languages and encoding methods, there's a risk of attacks through these statements. One serious threat is SQL Injection, a kind of code-injection attack brought on by insufficient user input validation. One of the most critical vulnerability in web security is SQL Injection (SQLIA). It happens when attackers sneak in harmful code fragments through input fields on a website. Security Mechanism like Firewall, Cryptography and Traditional Intrusion Detection Systems can be bypass using SQL Injection[10]. This can make the server execute unauthorized queries, leading to data exposure and damage to the database. it's surprising that many people, even those who are tech-savvy, struggle to identify phishing attacks and email spam. This lack of awareness is a concern because most confidential transactions occur on the web. To address these issues, it's crucial for everyone to have a basic understanding of web security. Despite efforts to find solutions, the current techniques aren't very effective in preventing SQL Injection attacks. Since SQLIA is application-level security vulnerability, it's essential to strengthen the validation of user input to avoid such risks. Awareness and education about web security

Hossain, M. M., Hasnat, M. A., & Islam, M. S. (2025). Enhancing Web Security: A Comprehensive Approach to Detect and Prevent SQL Injection Attacks through Innovative Query Comparison and Encryption Algorithms.
International Journal of Scientific Research and Modern Technology, *4*(1), 123–133.
https://doi.org/10.5281/zenodo.14960117

are key to protecting sensitive information online. SQL Injection gives access the attacker to obtain the username, password and other privacy data of from website, which posing a major risk to the data security [2-4]. The main goals of using SQL Injection attack are to gain illegal access to a database, extracting Information from the database, modifying the existing database. SQL injection attacks are the most concerning element of web applications, and their prevalence is expected to increase if the trend of providing web-based services continues[10].

This paper will concentrate on protecting the web and data before the hacker accesses the databases. So that, don't provide access the hacker to the databases before ensuring the integrity of the written query and made sure there are no commands that allow the hacker to manipulate sensitive data. Additionally the paper present different types of SQLI attack and prevention techniques against all types of SQL injection attacks and required implemented.

II. SQL INJECTION ATTACK

SQL injection is a technique used to retrieving or destroying data from a database without authorization. It is considered one of the top web application security risks. Where data fuels the core functionalities of web applications, ensuring the security of databases is of paramount importance. Web-based SQL injection occurs when an attacker manipulates input fields on a web application to inject malicious SQL code, potentially compromising the underlying database. Sometime face we Unauthorized access, Data manipulation, Information leakage, System compromise. There have been major repercussions from the exploitation of application-level vulnerabilities: hackers have deceived e-commerce sites into sending items for free, stolen usernames and passwords, and the disclosure of private data (including addresses and credit card details) [3]. Inadequate sanitization of user input can lead to a number of attacks that exploit web-based apps to undermine back-end database security [3]. The query statement is often composed of a set of words and symbols that enable it to access the web and databases. While visiting the login screen, it often compares the values stored in the database tables with a username and password. However, by using the query, no password is required. Figure 1 shows a SQL injection attack login for a website admin.

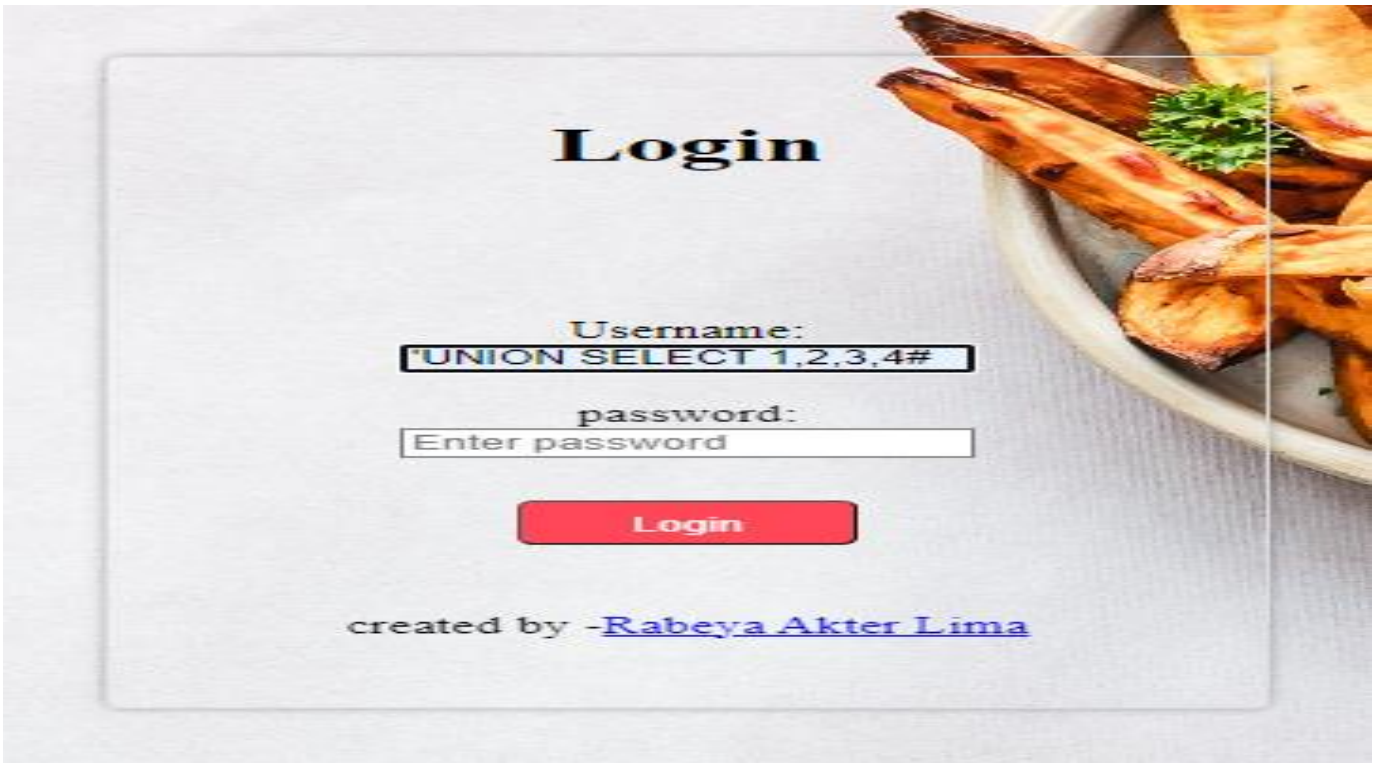

Fig 1 Attacker Login Admin Site Page

' UNION SELECT 1,2 ,3,4#

This attack called union-based SQLi. In-band SQLi, Inferential SQLi and Out-of-band SQLi are the three main categories of SQL Injection. Blind-Boolean-based SQLi and Blind-time-based SQLi are the two varieties of inferential SQL Injection. Since it relies on features that are enabled on the database server that the web application is using, out-of-band SQL injection is not very prevalent. Normally classic SQL injection bypass involves exploiting vulnerabilities in the input validation process to inject arbitrary SQL code. a login form on a website where the username and password are checked against a database:

```
SELECT * FROM users WHERE username = ' ' AND
password = ' ';
```

If the input is not properly sanitized, an attacker may input something like:

Username: admin' OR '1'='1, password: anything or blank. For example, figure 2 is that way hacker attack in website.

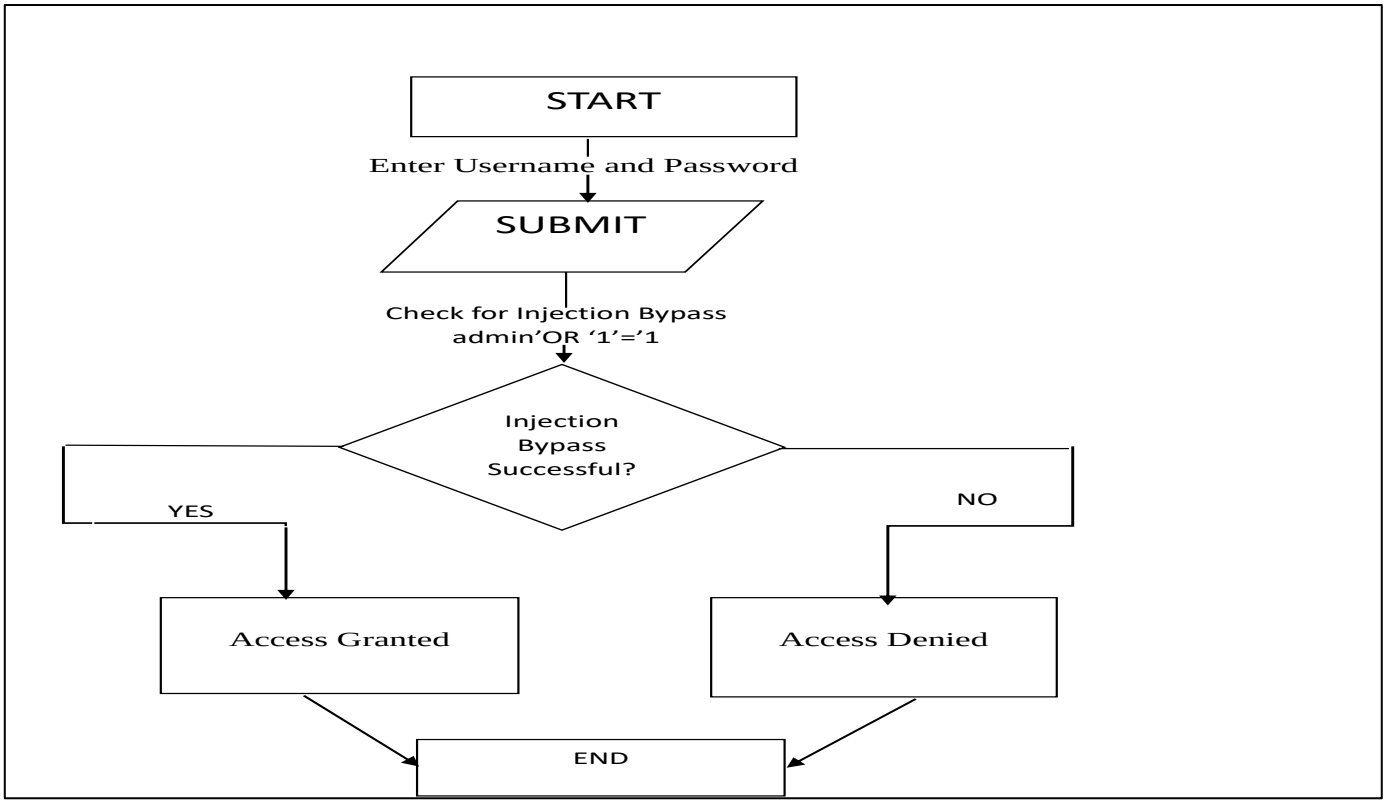

Fig 2 Classic SQL Injection Bypass Flowchart

Figure 1 actually this (figure 3) way to attack website, hacker follow this figure 3 instruction and unauthorized access in web site.

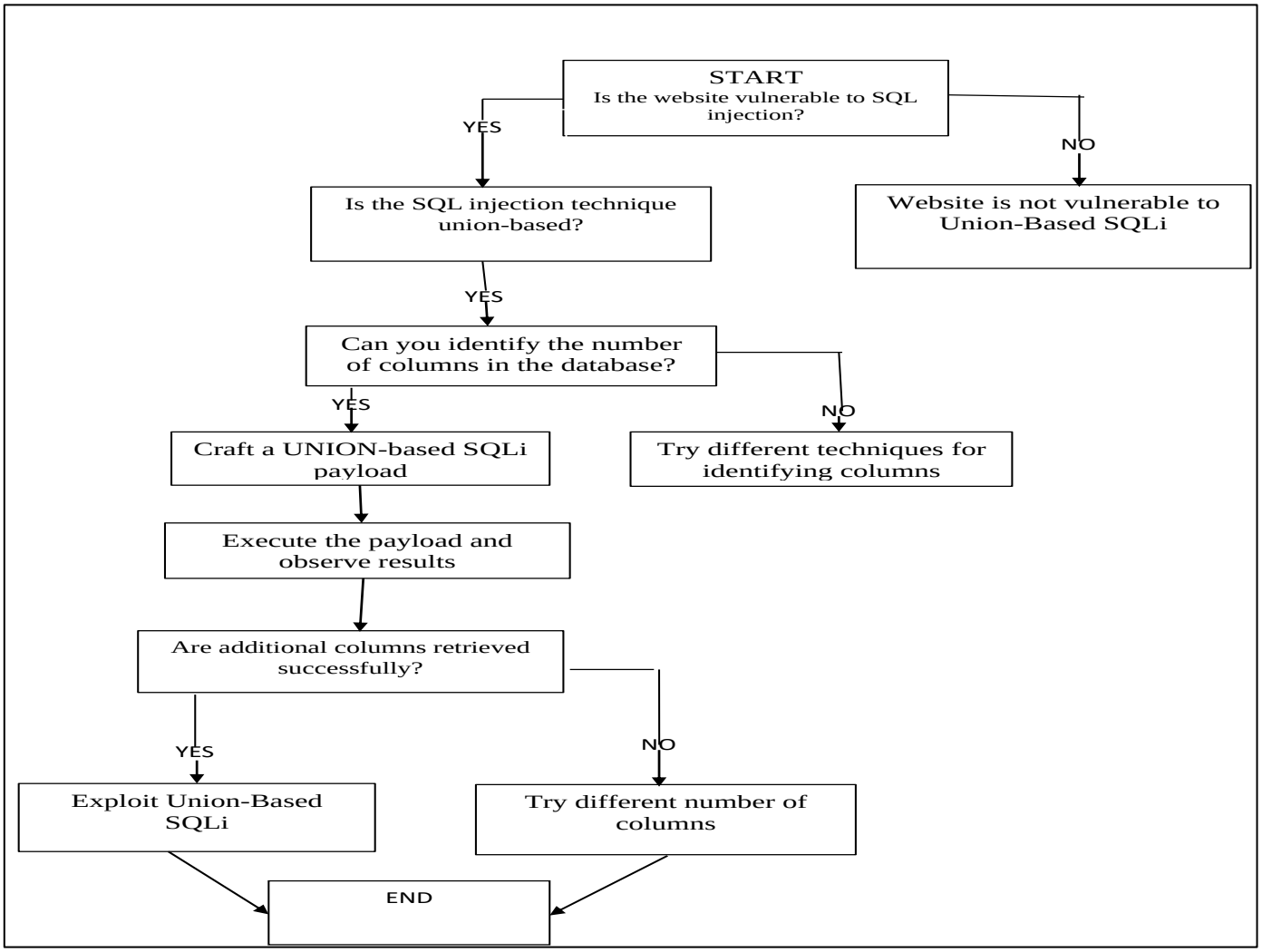

Fig 3 Union Based SQL Injection Flowchart

In a Boolean-based attack, the attacker typically exploits the application by injecting SQL code that evaluates to either true or false, allowing them to deduce information about the database.

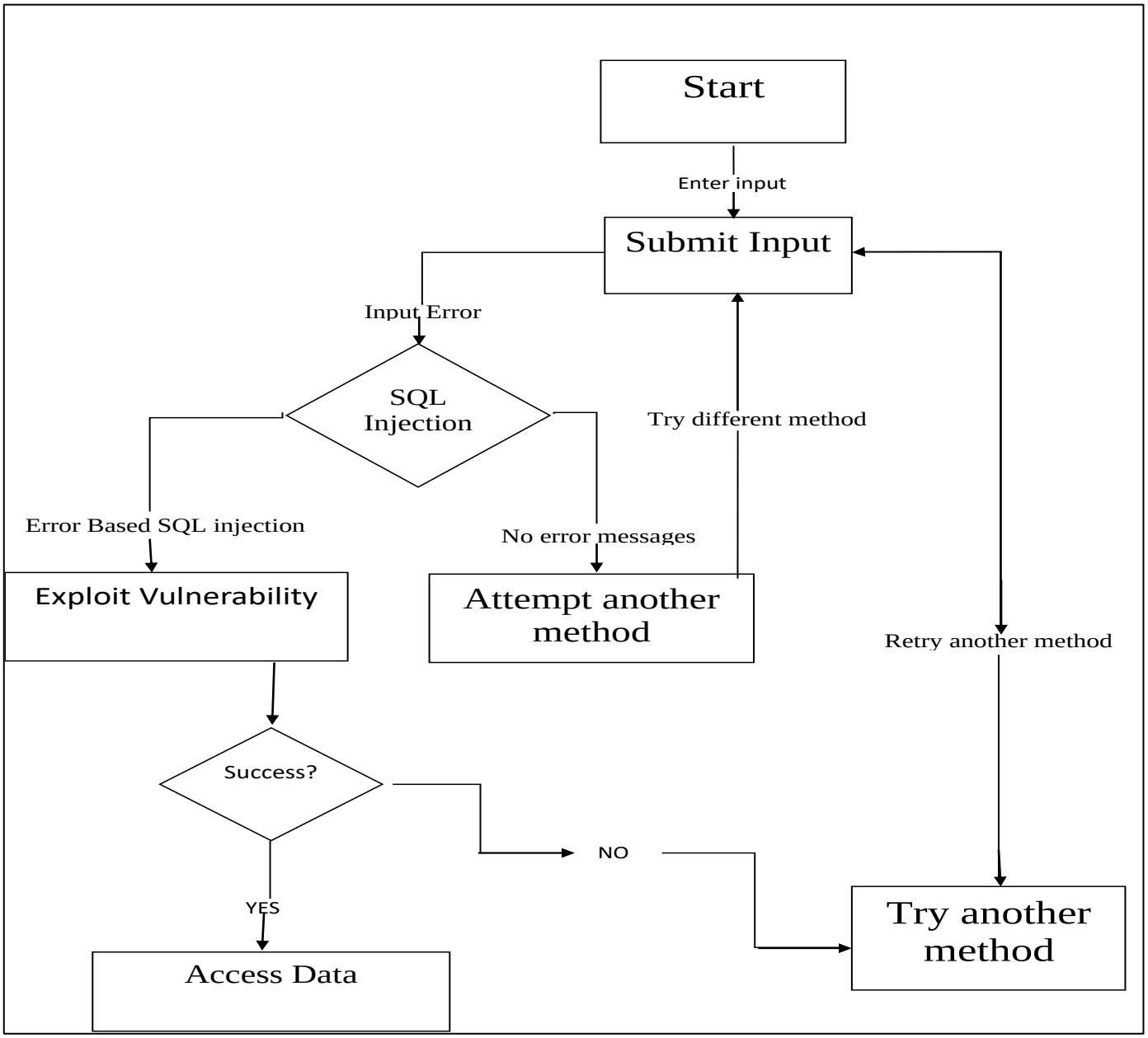

Fig 4 Boolean-based SQL Injection Flowchart

Additionally, there are kinds of SQLIAs including Tautologies, Piggy backed queries attack, Union attack, Illegal incorrect queries, Inference based attack, Alternate encodings and Stored procedures. Table 1 summarizes various types of SQLIAs, type of categories (SQL Manipulation, Code Injection, Function Call Injection, and Buffer Overflow), description and their corresponding examples. These various types of SQLIAs target different aspects of SQL query processing, ranging from manipulating logic to exploiting coding vulnerabilities. To reduce the danger of SQL Injection, developers and security experts must be aware of these attack vectors and use parameterized queries and strong input validation. Potential vulnerabilities can be found and fixed with the use of routine security audits and monitoring before they are exploited by malicious actors.

As a result, SQL injection was necessary to prevent these kind attacks. Additionally, it protects against harmful attacks that could harm web applications of organizations both physically and morally.

Table 1 Types of SQLIA

S.No	Type of Attack	Category	Working Method	Example
1	Tautologies	SQL manipulation	Tautology injects Malicious Code into the Conditional Statements so that they always evaluates to True.	select * from users where name= 'admin' OR '1' = '1'
2	Piggy backed queries attack	Code injection	An independent query is injected into the legitimate query in a piggy-backed queries attack, and it is executed after the initial query has already been run.	select * from users where username = "; INSERT INTO users VALUES('anything','1536')--
3	Union query	SQL manipulation Code injection	By joining two independent queries a UNION query allows an attacker to retrieve data from a different table.	select * from users where username = "UNION SELECT SUM (USERNAME) from users BY user id having 1 = '1' and password = 'anything'.
4	Illegal incorrect queries	SQL manipulation	When an attacker injects an improper query into a web application, they are able to obtain information about the database structure. This results in an error that contains crucial database information.	select * from users where username = Having 1 = '1' and password = 'anything'
5	Inference based attack	Code injection Buffer overflow	This kind of attack involves asking the server certain true-false queries in order to deduce important information.	select * from products where category = 'books' OR '1' = '1';

III. RELATED WORK

SQL Injection and Cross-Site Scripting Attacks are two risks to Web Application Security. SQL Injection is one of the most often exploited vulnerabilities. This form of database attack has destroyed companies, ruined careers, and is a constant challenge for security officers. Data is the greatest asset that database professionals have, and it is their duty to protect it above all else [3]. The amount of SQL injection to which online applications are subject has significantly and quickly expanded in recent years. As a result, researchers were highly motivated to defend web applications against these assaults in order to protect databases and data. Numerous strategies to fend off these attacks have been discovered by researchers [1]. It is still unclear, which of these techniques and algorithms are best suited for use in the relevant institution, which are adequate for the task and which are ideal for safeguarding web applications. In addition, some SQL Injection attack protection solutions might be costly and require a plan and budget for the infrastructure of the company to function properly. Naturally, small and medium-sized organizations are unable to handle these financial obligations [1]. Precisely tracking tainted data and specially checking for dangerous content in portions of commands and output that came from unreliable sources is the foundation of a fully automated approach to securely hardening web applications.

Since, there are many proposed algorithms were designed to SQL injection and prevent SQL injection, but they have several 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. Many researchers neglected a crucial URL layer, which is a vulnerability that hackers use to access the databases [1]. Lastly, only a few queries are examined by the majority of the algorithms listed; they merely handle the login screen. An algorithm was strongly encouraged to process these parameters.

IV. METHODOLOGY

Securing websites and databases is crucial due to the constant threat of hacker attacks. Efforts are needed to devise effective strategies in response to the evolving techniques employed by attackers aiming to breach databases and access sensitive information. Understanding attackers' methods is essential for safeguarding data. Hackers typically seek valuable information and conduct thorough analyses, focusing on system behavior and vulnerabilities. An example is SQL injection, where attackers exploit weak points after investing time and effort in understanding system operations. Once a vulnerability is identified, hackers can exploit it to gain access to user permissions and subsequently acquire sensitive data.

This paper introduces a model designed to safeguard the initial interface from unauthorized access by hackers and to provide security for the data stored within databases. Refer to Figure 5 for a visual representation of the proposed methodology.

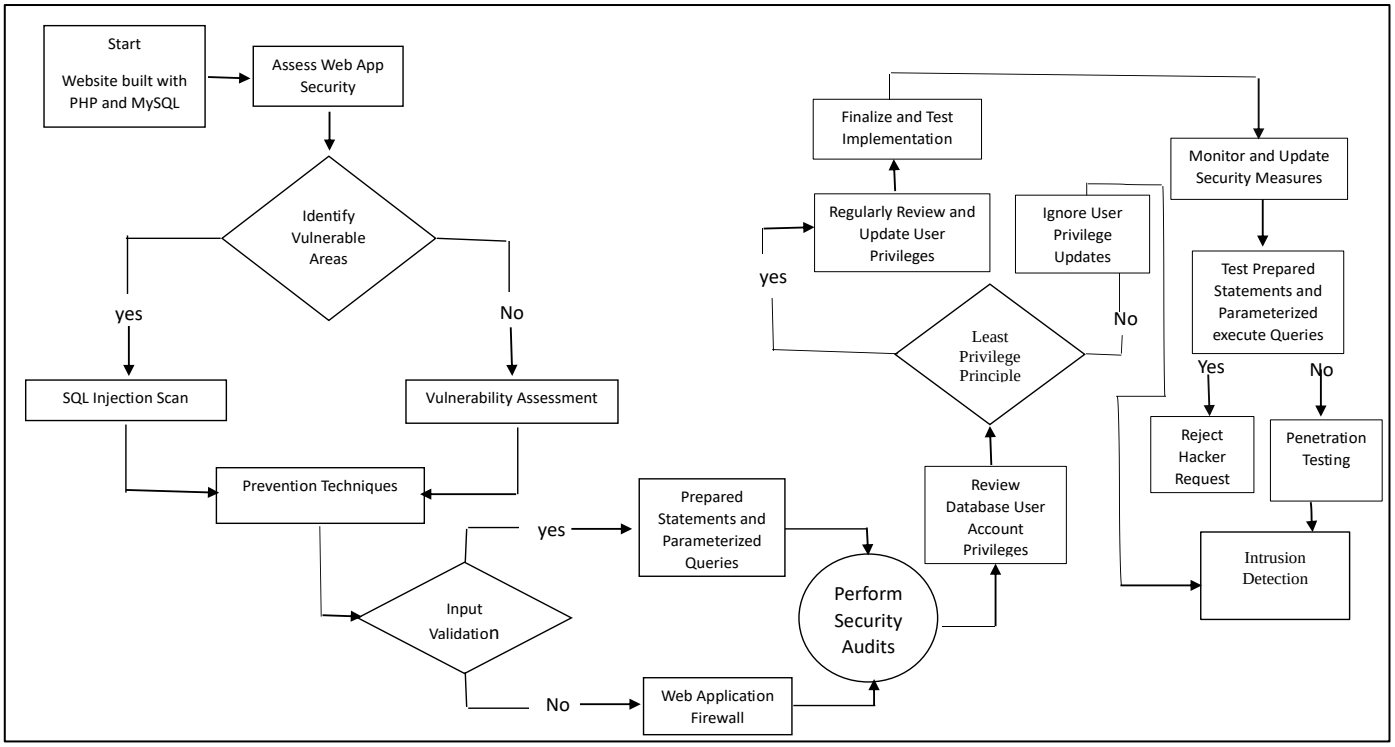

Fig 5 Process Diagram

➢ *Process Diagram for Building a Secure PHP and MySQL Web Application:*

- Website Development: Start by building the website using PHP and MySQL as the primary technologies.
- SQL Injection Scan: Perform an initial scan for SQL Injection vulnerabilities to identify potential weaknesses in the application.
- Security Assessment: Examine the web application's overall security to find any weak points that could be attacked.
- Vulnerability Identification: Determine which particular application components are at risk of attack, such as input fields that could be subject to injection attacks.
- Prevention Techniques: To stop attacks like prepared statements, parameterized queries and input validation.
- Input Validation: To make sure that all user input is properly sanitized and validated.
- Use Prepared Statements and Parameterized Queries: Employing prepared statements and utilizing parameterized queries is vital to minimize the chances of being hit by SQL Injection attacks.
- Consider a Web Application Firewall Use a web application firewall as an extra security measure.
- Finalize and Test Implementation: Finalize the implementation of security measures and thoroughly test the application to ensure that all security features are functioning properly.
- Regularly Review and Update User Privileges: Make sure people have access only the resources they require by routinely reviewing and updating their needs [20].
- Perform Security Audits: Use to find and proactively fix any possible security flaws.
- Implement Least Privilege Principle: Grant users the minimal amount of access necessary to carry out their duties [12].
- Ignore User Privilege Updates: Do not ignore updates to user privileges, ensuring that all changes are properly reviewed and approved.
- Review Database User Account Privileges: Regularly review the privileges assigned to database user accounts to ensure that they are appropriate and necessary.
- Monitor and Update Security Measures: Continuously monitor security measures and update them as needed to respond to new threats and vulnerabilities.
- Test Prepared Statements and Parameterized Queries: Make sure that prepared statements and parameterized queries are appropriately rejecting hacker requests by testing them on a regular basis [15].
- Penetration Testing: Perform regular penetration testing to identify any potential security weaknesses and address them proactively.
- Intrusion Detection: Implement intrusion detection measures to alert you of any suspicious activity on your web application.
- By following this process diagram, you can build a secure PHP and MySQL web application that is resistant to attack and protects sensitive data.

In web applications, SQL Injection attacks are a common and serious security risk. A weakly built web application and inadequate input validation are two advantages that hackers can use in SQL Injection attack [2]. Data confidentiality and integrity are compromised as a result of a successful SQLIA, which degrades the market value of the organization. This paper presents a useful analysis of several kinds of SQLIAs, methods and processes. It also explores different methods of detection and prevention.

V. IMPLEMENTATION

The proposed algorithm can be implemented as part of a comprehensive web security framework. It seamlessly integrates with existing systems and requires minimal adjustments to deploy. Its simplicity ensures that organizations of varying sizes and technical expertise can adopt and benefit from enhanced security measures.

➢ *Benefits:*

- *Cost-Effective:*

The algorithm leverages existing technologies and focuses on simplicity, making it a cost-effective solution for organizations with budget constraints.

- *User-Friendly:*

Its straightforward design facilitates easy implementation and maintenance, reducing the need for extensive training or specialized expertise.

- *Scalable:*

The algorithm can scale to accommodate the evolving needs of growing databases and web applications.

A web application connected to the database for accessing data. Take a look at Figure 5 below, outlining how prevent unwanted data insertions.

Fig 6 Use of Username and Password

Using the interface of web application, the input fields will be filled in. When a user selects "insert data," the information will be saved in the MySQL database. The database structure is illustrated in the accompanying figure 6.

➢ *Below is the pseudo-code outlining the measures to prevent insertion:*

- The (Host, Database User, Database Password, And Database Name) Have Been Initialized.
- //Connect Web Server to Database Using Mysql Method.Mysqli($Host, $Dbuser, $Dbpass, $Dbname);
- The Username Is Initialized and Waits to Execute
- //The Password Is Initialized and Encrypted Using the Md5 Method $ POST['username']);$raw_password = md5($ POST['password']); $password = mysqli_real_escape_string($conn, $raw_password);
- The Query Is Waiting to Be Prepared
- “Insert Into Users (Username, Password) Values (??)”);
- The Query Is Prepared
- // Bind Parameter Is Invoked Bind_Param (‘Ss’, $User, $Password);
- Execute (); \\Execute Method Is Run

In this case, stopping unauthorized persons from getting into the database to sabotage or steal data has been accomplished using a straightforward algorithm. The process involves several stages and primarily focuses on preventing SQL injection, a common method used by hackers to gain access. The key strategy is to block a typical tactic employed by malicious individuals: messing with character spacing.

Table 2 Character Spacing

Character	Task
--	Line Comment
“or”	String
+ ,||	Concatenate
/*---*/	Many lines comment
’0:0:22’	Wait for the time delay
?php1=abc&ad=mar	URL

Also, when comparing detection and prevention methods, it becomes evident that detection focuses on identifying security breaches and unauthorized access after they occur, while prevention is centered around implementing measures to stop these incidents from happening in the first place. Detection often involves monitoring and analyzing system logs and network activities to catch anomalies, whereas prevention relies on robust security protocols, encryption, access controls, and other proactive measures to fortify the system against potential threats. Striking a balance between effective detection and prevention strategies is crucial for comprehensive cybersecurity. In table 3 also compare normal and intruders.

Table 3 Comparison between Different Detection and Prevention Methods [6]

	Normal	Intruders
SQL injection query	Can access rows in table	Can access rows and all tables
Prepared Statement	Can access one row	Can access one row at a time
Stored procedures	Can insert, delete, update values in the table	Database admin can revolve execution
White list input validation	A special query has to be passed	A special query has to be passed

The number of Internet application attacks decreased by 69% in the third quarter of 20217 compared to the third quarter of 2016, according to Akamai. Just 9% of these attacks were caused by XSS, compared to 85% by SQL Injection and Local File Incorporation Attacks. According to Verizon’s data breach study [14], web application attacks accounted for 29.5% of breaches, whereas only 15.4 percent of recorded events involved Web Application Assaults. SQL Injection is the most prevalent attack on the Internet, 85% of web application attacks in Q3 2017 were caused by this according to Research Done by US-Based Cloud Service Provider. From November 2017 to March 2019, SQL Injection attacks accounted for 65% of web-based attack vectors. Additionally, some websites continue to be attacked in 2024 even if the majority of online apps receive at least four web attack campaigns per month [9].

In a single day, 94,057 SQL injection attack requests are sent to each website. SQL injection attacks are twice as common in e-commerce as in other sectors, 98% of the time, 176 days out of 180, an observed website was attacked. On average, 94,057 equates to 26 attack requests each minute or 1,567 SQLI attacks per hour.

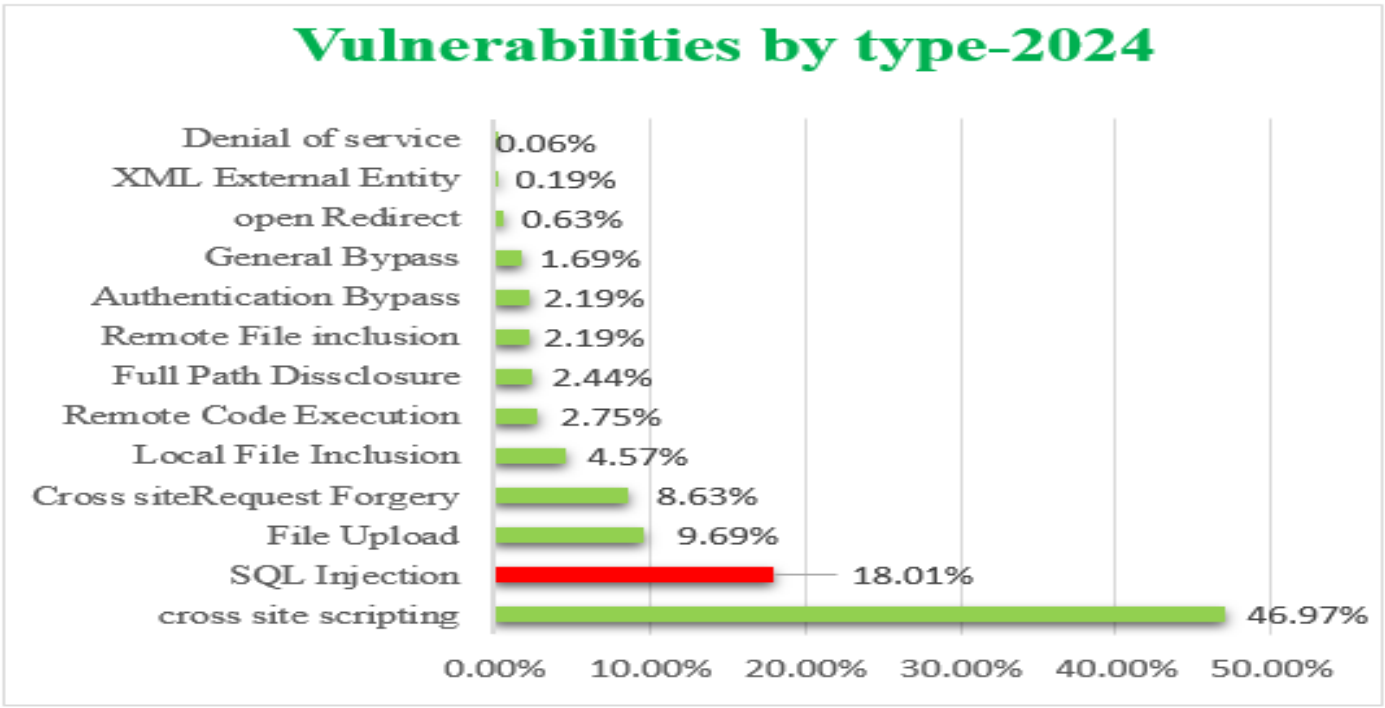

Fig 7 Vulnerabilities by Type-2024 [6]

According to WordPress sites 2024 vulnerabilities by type SQL injection 2nd most widely used exploit.

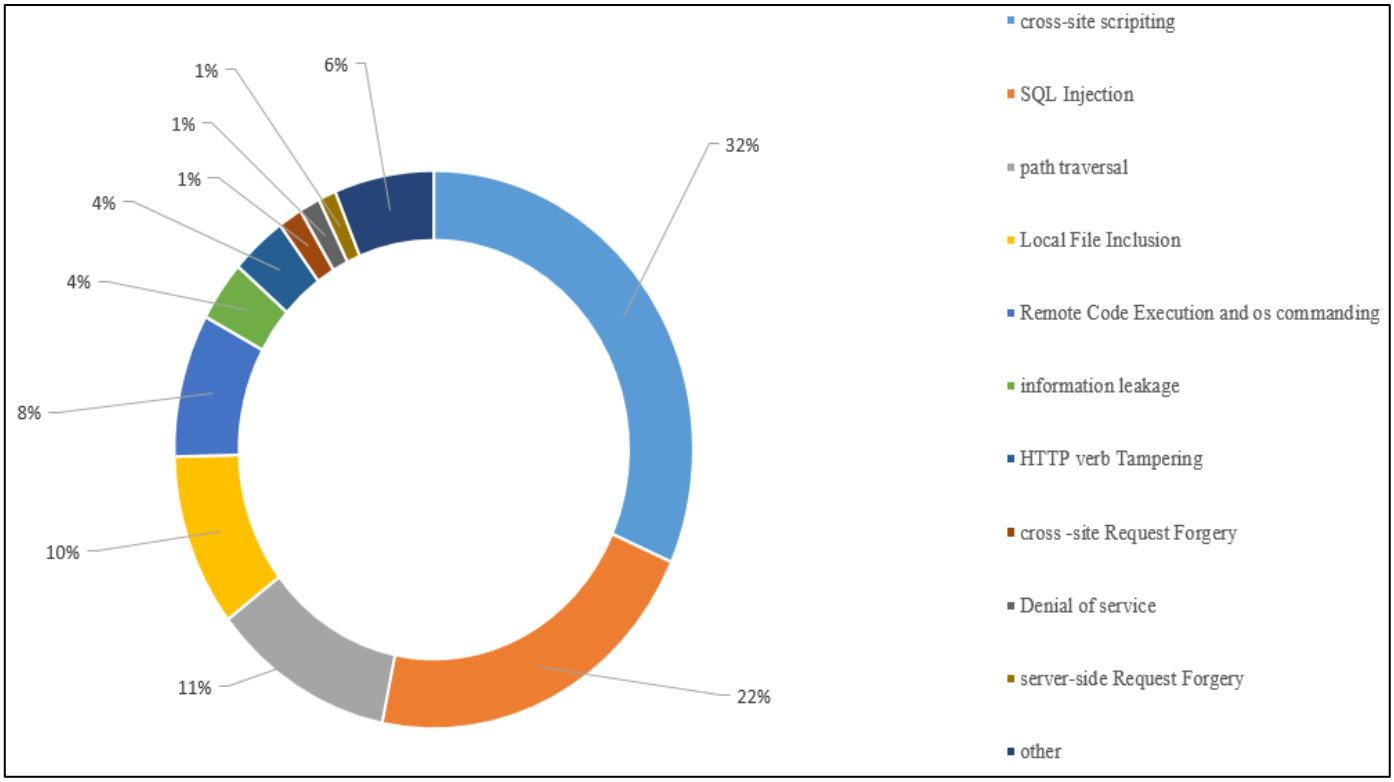

Fig 8 Vulnerabilities by Sites-2024

This chart shows that while SQL injection is the second most common vulnerability in WordPress sites. which is 22% Assuming that the list provided contains different types of web application vulnerabilities, and create a chart focusing on SQL Injection. Here's a simple chart illustrating SQL Injection and its related information:

Table 4 SQL Injection Vulnerabilities

Category	Vulnerability Type	Description
Injection	SQL Injection	Unauthorized SQL queries are injected into a web application's input fields, allowing unintended access to the database.
Techniques	Error-based SQLI	Exploiting SQL errors to extract information or perform unauthorized actions.
	Union-based SQLI	Combining query results from multiple tables to extract sensitive data.
	Blind SQLI	Inferring database content using true/false responses or time delays.
Prevention	Input Validation	Validating and sanitizing user inputs to prevent malicious SQL queries.
	Prepared Statements	Using parameterized queries to separate code from data, reducing the risk of SQLI.
	Least Privilege Principle	Granting the web application the minimum database permissions necessary to function.
Detection	Automated Scanning	Using automated tools to identify SQL Injection vulnerabilities in web applications.
	Manual Testing	Expert security testers manually attempting to exploit SQL Injection vulnerabilities.

An evaluation of various techniques, including AMNESIA, SQLCHECK, CANDID, Automated approach, Tautology checker, SQLrand, SQLDOM, CSSE, and WebSSARI, against a variety of attacks, including tautology, logically incorrect queries, union queries, stored procedures, piggybacked queries, inference attacks, and alternate encodings, is provided in Table 5. The '√' symbol indicates whether a technique can detect or prevent the corresponding attack, while '×' denotes its inability to do so. According to the analysis, the WebSSARI technique emerges as the most effective, successfully preventing all types of SQL injection attacks. In contrast, CANDID and the tautology checker only demonstrate prevention capabilities for one specific attack type, namely tautology. This underscores the superiority of the WebSSARI technique over other approaches in thwarting a comprehensive range of SQL injection threats. Additionally, recent studies, including Puneet's analysis of classical and modern approaches [16], and another study comparing various detection and prevention techniques [17], highlight the effectiveness of SQL IDS, SQL Checker, and SQL Prevent in detecting diverse SQL injection attacks.

Table 5 Analysis of the Differences in Attack Types, Protection Strategies, and Detection

Types of Attacks →	**Tautology**	**Logically incorrect queries**	**Union query**	**Stored procedure**	**Piggy backed queries**	**Inference attack**	**Alternate encoding**
Detection and Prevention Techniques ↓							
AMNESIA	√	√	√	×	√	√	√
SQLCHECK	√	√	√	×	√	√	√
CANDID	√	×	×	×	×	×	×
Automated approach	√	√	√	×	√	√	×
Tautology checker	√	×	×	×	×	×	×
SQLrand	√	×	√	×	√	√	×
SQLDOM	√	√	√	×	√	√	√
CSSE	√	√	√	×	√	√	
WebSSARI	√	√	√	√	√	√	√

√ This indicator shows whether the strategy can identify or stop the attack.

× This sign denotes if technique is unable to detect or stop the attack.

SQLIA detection can be done by checking anomalous SQL query structure using string matching, pattern matching and query processing.

VI. CONCLUSIONS

Website-based SQL injection represents a persistent threat in the realm of cybersecurity. Its existence underscores the critical importance of strong web application security measures. This research paper has introduced innovative algorithms useful for addressing and mitigate the risks associated with website-based SQL injection. The proposed methodologies demonstrate a robust approach to identifying and preventing SQL injection vulnerabilities in web applications. The current research thoroughly examines different types of SQL Injection Attacks (SQLIA) and conducts a critical analysis. The focus is on understanding and addressing SQLIA through the investigation and evaluation of various detection and prevention techniques. The research involves a comparative analysis of different attack types and the corresponding mitigation techniques. By investigating the efficacy of pattern matching algorithms like Brute-force, Rabin-Karp, Boyer-Moore, Knuth-Morris-Pratt, and Aho-Corasick for identifying and thwarting various assaults, the ultimate objective is to eradicate significant SQLIA. The algorithm has been implemented for the main login screen of the website and other screens related to the same online application. Over 250 requests from the web and URL layers were used to test the system. Comparing the results with SQLPMDS and SIUQAPTT shown that the suggested technique provided greater speed and scalability to defend against and identify different types of attacks.

REFERENCES

[1]. E. Pollack, "Protecting against SQL injection: Applications performance and security in microsoft SQL server", Proc. Dyn. SQL, pp. 31-60, 2019.

[2]. A. A. Sarhan, S. A. Farhan And F. M. Al-Harby, "Understanding and Discovering SQL Injection Vulnerabilities", Proc. Int. Conf. Appl. Hum. Factors Ergonom., Pp.1063-1075, 2017

[3]. A. Maraj, E. Rogova, G. Jakupi and X. Grajcevci, "Testing Techniques and Analysis of SQL Injection Attacks", Proc. Int. Conf. Knowl. Eng. Appl. (ICKEA), Pp. 1-11, 2017

[4]. D. Das, U. Sharma and D. K. Bhattacharyya, "Defeating SQL Injection Attack in Authentication Security: An Experimental Study", Int. J. Inf. Secur., Vol. 18, No. 1, Pp. 1-22, 2017

[5]. D. Scott and R. Sharp. Abstracting Application-level Web Security. In Proceedings of the 11th International Conference on the World Wide Web (WWW 2002), pages 396–407, 2002

[6]. F. Valeur, D. Mutz, and G. Vigna. A Learning-Based Approach to the Detection of SQL Attacks. In Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA), Vienna, Austria, July 2005

[7]. H.-C. Huang, Z.-K. Zhang, H.-W. Cheng and S. W. Shieh, "Web application security: Threats countermeasures and pitfalls", Computer, vol. 50, no. 6, pp. 81-85, 2017.

[8]. 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

[9]. 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.

[10]. 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.

[11]. A. Nguyen-Tuong, S. Guarnieri, D. Greene, J. Shirley, and D. Evans. Automatically Hardening Web Applications Using Precise Tainting

Information. In Twentieth IFIP International Information Security Conference (SEC 2005), May 2005
[12]. Nikita, P., Fahim, and S. Soni, SQL Injection Attacks: Techniques and Protection Mechanisms. International Journal on Computer Science and Engineering, 2011. 3
[13]. Akamai, State of the Internet/Security, Q3 2017 Report
[14]. Verizon, 2017 Data breach investigations report, 10th edition.
[15]. Web Attacks and Gaming Abuse Report: Volume 5, Issue 3
[16]. Singh JP. Analysis of SQL injection detection techniques[Internet]. 2016 [updated 2016 Dec 15; cited 2016 May 9]. Available from: Crossref.
[17]. Dehariya H, Shukla PK, Ahirwar M. A survey on detection and prevention techniques of SQL injection attacks. International Journal of Computer Applications. 2016 Mar; 137(5):9–15. Crossref.