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

Rabeya Akter Lima, Department of CSE. Hamdard University Bangladesh, rabeyaakterlim4@gmail.com

Abstract: In today's world, web applications have become a vital necessity for the everyday requirements of all organizations. These applications interact with databases, where data and information are stored, organized, retrieved, and processed. Consequently, the majority of its attacks are directed towards databases. Attacks on websites and the compromise of individuals' secure information are increasing at a mounting rate. Since the beginning of social networking and e-commerce, web security has become popular, as attacks such as phishing and spamming have become common. 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 proposes a new algorithm to attack website and also prevent hackers from accessing databases early on through the web application without accessing databases. The proposed algorithm is designed to attack and protect the web application from being voluntarily inserted by using a Prevention Techniques, blocking the hacker's address, and rejecting hacker request when executing the query and update regularly security. 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. To improve the web software security research work developed a defense mechanism that protects web resources from SQL Injection. The developed software uses PHP, JavaScript, and a formal language theory known as regular expression to create a defense mechanism. this tool helps protect web resources from SQL Injection Vulnerabilities, Providing A Way for Users to Safeguard Their Web Applications from Potential Attacks.

Key Word: web Application, SQL, PHP, SQL injection attack, prevention, web security.

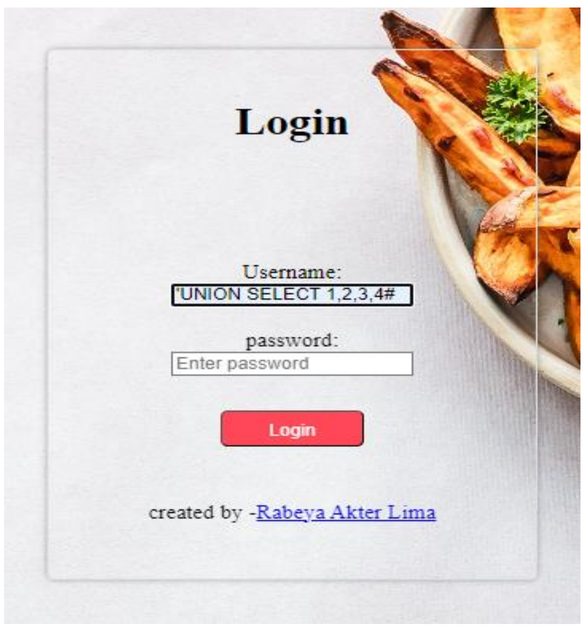
1.Introduction

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. web programming languages (such as PHP, java) provide various methods for constructing and executing SQL statements [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 significant threat is SQL Injection, a type of code-injection attack that occurs due to inadequate validation of user input. SQL Injection (SQLIA) is a critical vulnerability in web security. It happens when attackers sneak in harmful code fragments through input fields on a website. SQL Injection Attacks Can Bypass the Security Mechanism Such as Firewall, Cryptography and Traditional Intrusion Detection Systems. 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 existing methods aren't very effective in preventing SQL Injection attacks. Since SQLIA is an application-level security vulnerability, it's essential to strengthen the validation of user input to avoid such risks. Awareness and education about web security are key to protecting sensitive information online. The attacker can obtain the username, password and other privacy data of website users by SQL Injection, which seriously threatens the data security [2-4]. The main intent to use SQL Injection attack include illegal access to a database, extracting Information from the database, modifying the existing database, escalation of privileges of the user or to malfunction an application. ultimately SQLIA involves unauthorized access to a database exploiting the vulnerable parameters of a web application. If The Trend of Providing web-based Services Continues, the prevalence of SQLI as is likely to increase access to the database that underlie web applications and the most worrying aspect of SQL injection attack.

This paper will focus on protecting the web and data before the hacker reaches the databases. Therefore, don't allow hacker to access the databases until after ensuring the integrity of the written query, as ensure that there are no commands that allow the hacker to access the databases and manipulate sensitive data. And also showed different SQLI attack types. SQL Injection attack and prevention techniques against all types of SQL injection attacks and deployment requirements.

2. SQL Injection Attack

SQL injection is a that kind of technique used to retrieve or destroy data from a database without permission. 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. Application-Level Vulnerabilities Have Been Exploited With Serious Consequences: hackers have tricked e-commerce sites into shipping goods for no charge, usernames and passwords have been harvested and confidential information (such as addresses and credit-card numbers) has been leaked [5]. If user input is not sanitized correctly, it is possible to mount a variety of attacks that leverage web-based applications to compromise the security of back-end databases [6]. 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 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. But if use the query don't needed password. Figure 1 shows a SQL injection attack login for a website admin.


Login

Username: 'UNION SELECT 1,2,3,4#

password: Enter password

created by -Rabeya Akter Lima

Figure 1: attacker login admin site page

```
' UNION SELECT 1,2 ,3,4#
```

This attack called union-based SQLi. for kind information SQLi can be classified into three major categories – In-band SQLi, Inferential SQLi and Out-of-band SQLi. The two most common types of in-band SQL Injection are Error-based SQLi and Union-based SQLi. The two types of inferential SQL Injection are Blind-Boolean-based SQLi and Blind-time-based SQLi. Out-of-band SQL Injection is not very common, mostly because it depends on features being enabled on the database server being used by the web application. 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 might input something like:

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

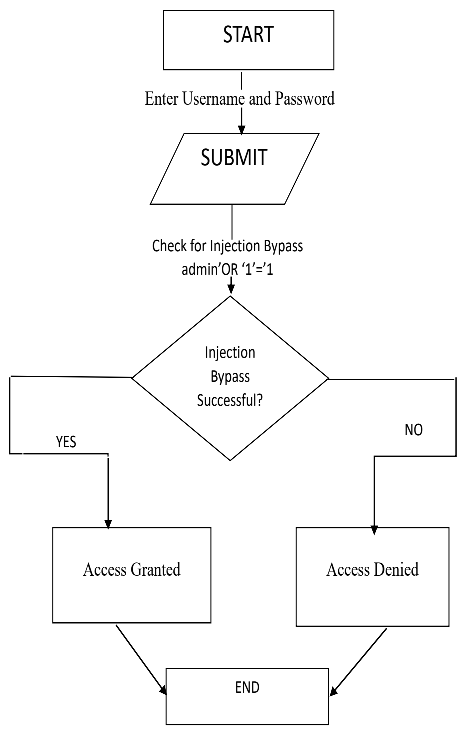

START
Enter Username and Password
SUBMIT
Check for Injection Bypass
admin'OR '1'='1
Injection Bypass Successful?
YES
NO
Access Granted
Access Denied
END

Figure 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.

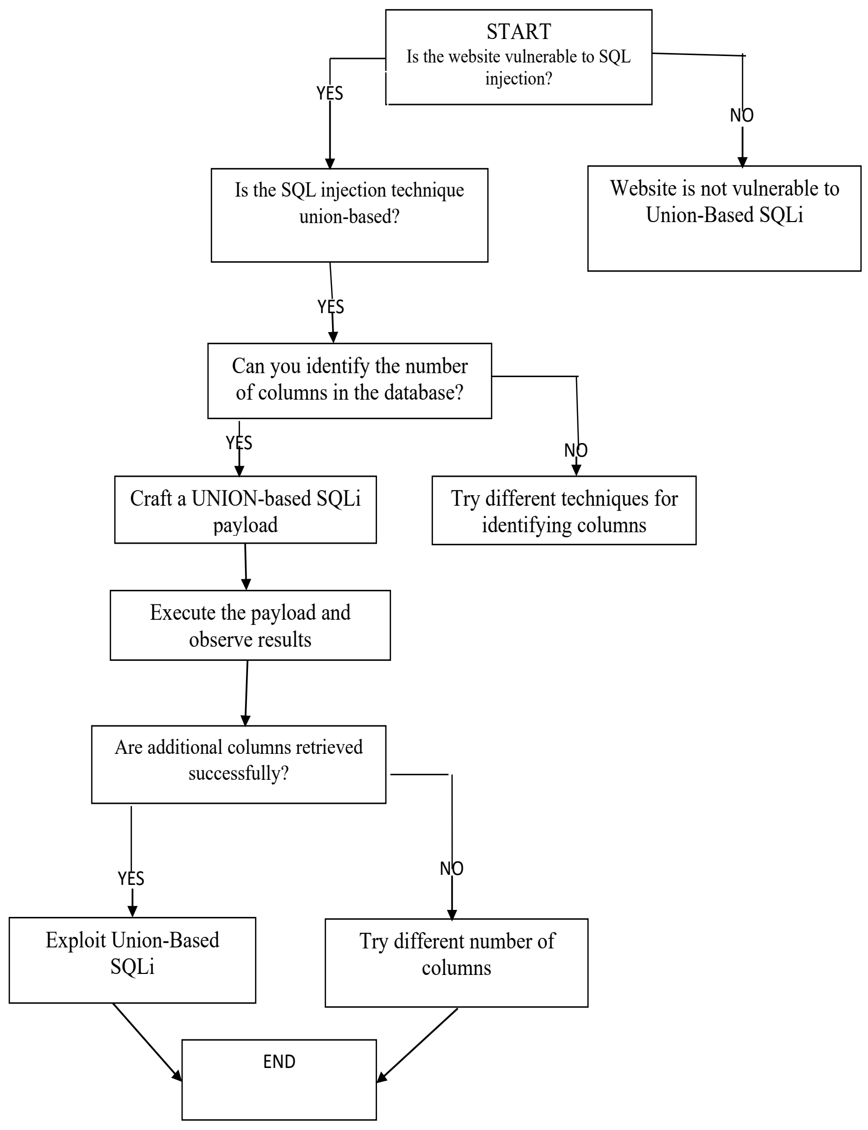

START
Is the website vulnerable to SQL injection?
YES
NO
Is the SQL injection technique union-based?
Website is not vulnerable to Union-Based SQLi
YES
Can you identify the number of columns in the database?
YES
NO
Craft a UNION-based SQLi payload
Try different techniques for identifying columns
Execute the payload and observe results
Are additional columns retrieved successfully?
YES
NO
Exploit Union-Based SQLi
Try different number of columns
END

Figure 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.

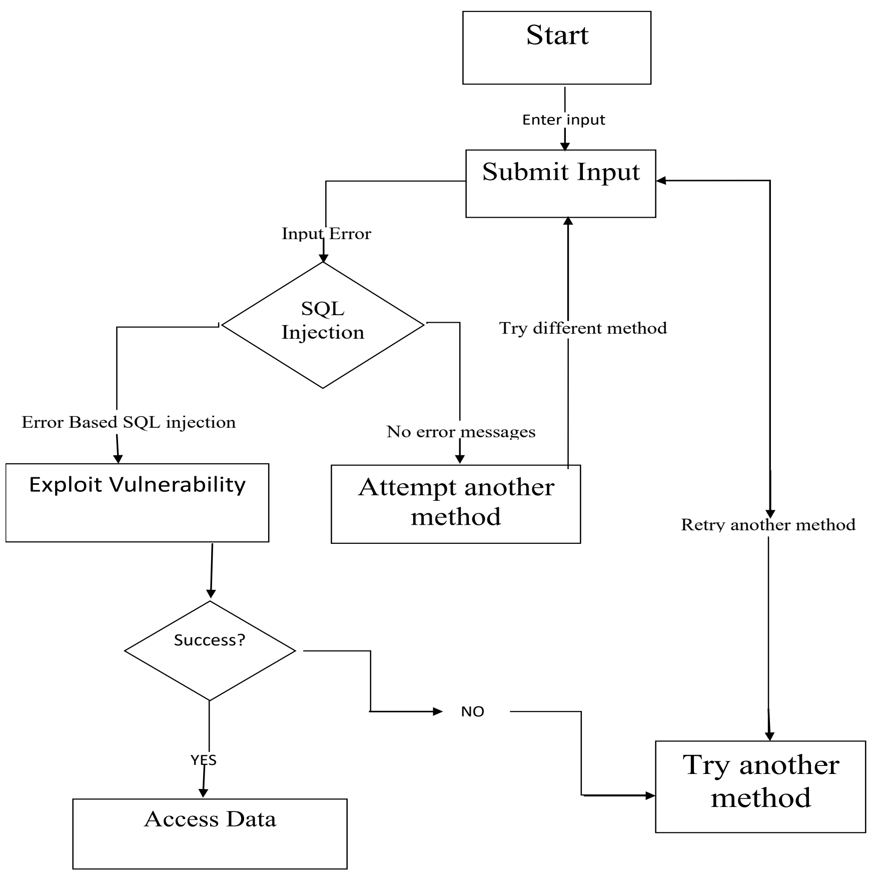


Figure 4: Boolean-based SQL injection flowchart

Also Various types of SQLIAs are Tautologies, Piggy backed queries attack, Union attack, Illegal incorrect queries, Inference based attack, Alternate encodings and Stored procedures5. 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. It is crucial for developers and security professionals to be aware of these attack vectors and implement robust input validation and parameterized queries to mitigate the risk of SQL Injection. Regular security audits and monitoring can help identify and address potential vulnerabilities before they are exploited by malicious actors.

Table 1. Types of SQLIA

S.No	Type of Attack	Category	Working Method	Example
1	Tautologies	SQL manipulation	In Tautology, Malicious Code Is Injected into The Conditional Statements So That It Always Evaluates to True.	select * from users where name= 'admin' OR '1' = '1'
2	Piggy backed queries attack	Code injection	In piggy backed queries attack, attacker injects an independent query to the legitimate query and it gets executed after the already run first query.	select * from users where username = "; INSERT INTO users VALUES('anything','1536')--
3	Union query	SQL manipulation Code injection	Union query basically joins two independents' queries and attacker use UNION to join these queries and extract the data from another 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	In Illegal incorrect queries attacker collects the information about the schema of database by injecting the wrong query into the web application due to which some error is generated and this error contains important information about the database.	select * from users where username = Having 1 = '1' and password = 'anything'
5	Inference based attack	Code injection Buffer overflow	In this type of attack useful information is inferred by asking the server some true false questions.	select * from products where category = 'books' OR '1' = '1';

Therefore, using SQL injection was necessary to prevent these kind 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

Web Application Security Threats Such as SQL Injection and Cross-Site Scripting Attacks [7]. SQL vulnerabilities as commonly exploited as SQL Injection. This form of database attack has destroyed companies, ruined careers, and is a constant challenge for security officers. As Database Professionals, Data Is Greatest Asset, And It Is Responsibility to Guard It Above All

Else.[1] 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 [8, 9]. 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[10]. A Fully Automated Approach to Securely Hardening Web Applications. It Is Based on Precisely Tracking Tainted Ness of Data and Checking Specifically for Dangerous Content Only in Parts of Commands and Output That Came from Untrustworthy Sources. Unlike Previous Work in Which Everything That Is Derived from Tainted Input Is Tainted, Approach Precisely Tracks Tainted Within Data Values [11] 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 [12].

The research gap is that many proposed algorithms were designed to SQL injection and 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 acritical URL layer, a weak point through which hacker scan 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

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.

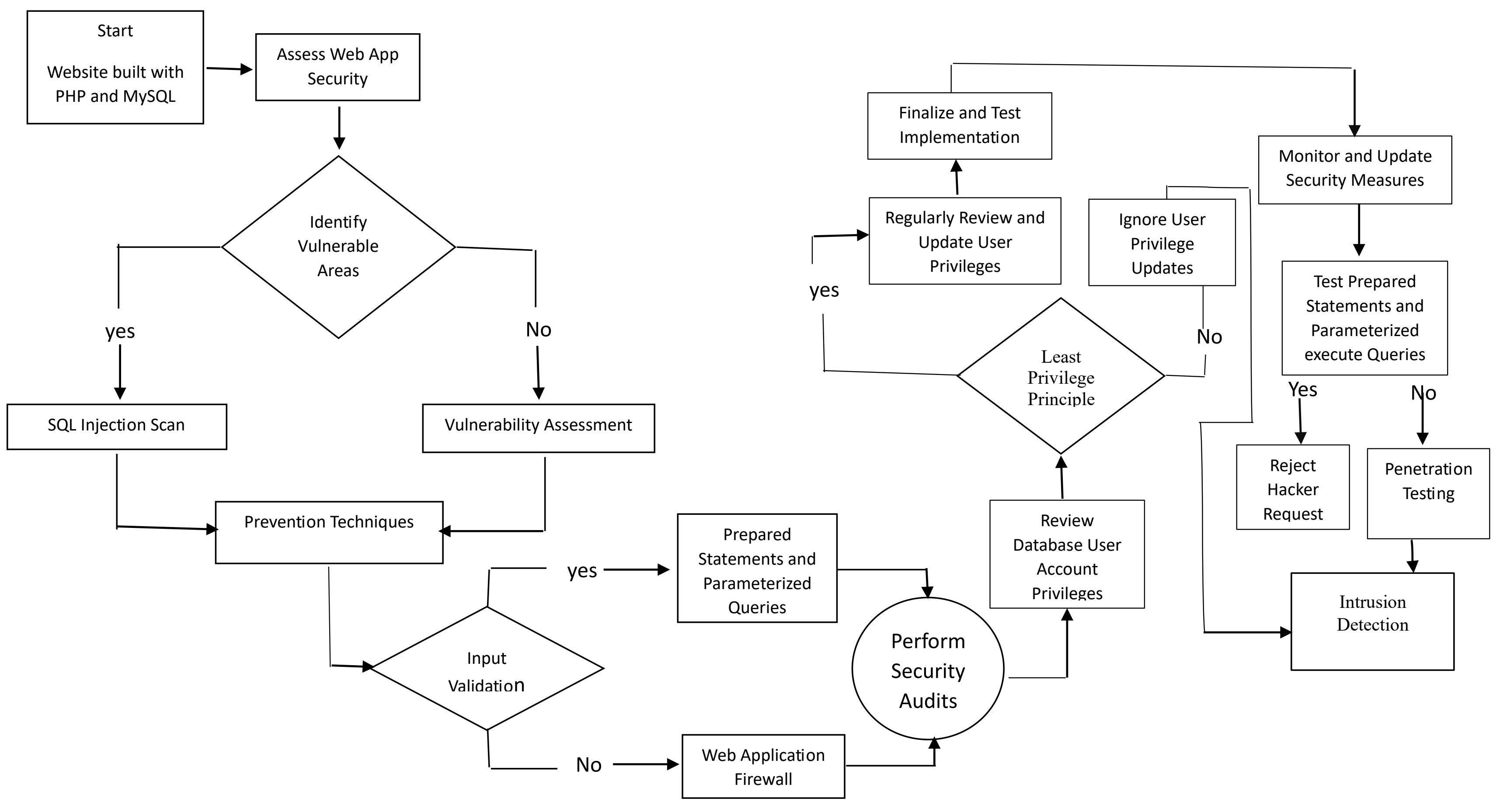

Start
Website built with PHP and MySQL
Assess Web App Security
Identify Vulnerable Areas
yes
No
SQL Injection Scan
Vulnerability Assessment
Prevention Techniques
Input Validation
yes
No
Prepared Statements and Parameterized Queries
Web Application Firewall
Perform Security Audits
Review Database User Account Privileges
Least Privilege Principle
yes
No
Regularly Review and Update User Privileges
Ignore User Privilege Updates
Finalize and Test Implementation
Monitor and Update Security Measures
Test Prepared Statements and Parameterized execute Queries
Yes
No
Reject Hacker Request
Penetration Testing
Intrusion Detection

Figure 5: Process Diagram

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

1. Website Development: Start by building the website using PHP and MySQL as the primary technologies.
2. SQL Injection Scan: Perform an initial scan for SQL Injection vulnerabilities to identify potential weaknesses in the application.
3. Security Assessment: Assess the overall security of the web application to identify any areas that may be vulnerable to attack.

4. Vulnerability Identification: Identify specific areas of the application that are vulnerable to attack, such as input fields that may be susceptible to injection attacks.
5. Prevention Techniques: Use best practices to prevent attacks, such as input validation, prepared statements, and parameterized queries.
6. Input Validation: Implement input validation to ensure that all user inputs are properly sanitized and validated before being processed by the application.
7. Use Prepared Statements and Parameterized Queries: Use prepared statements and parameterized queries to prevent SQL Injection attacks.
8. Consider a Web Application Firewall: Evaluate the use of a web application firewall to provide an additional layer of security.
9. Finalize and Test Implementation: Finalize the implementation of security measures and thoroughly test the application to ensure that all security features are functioning properly.
10. Regularly Review and Update User Privileges: Regularly review and update user privileges to ensure that users only have access to the resources they need.
11. Perform Security Audits: Perform regular security audits to identify any potential security weaknesses and address them proactively.
12. Implement Least Privilege Principle: Follow the principle of least privilege, giving users the minimum level of access required to perform their job functions.
13. Ignore User Privilege Updates: Do not ignore updates to user privileges, ensuring that all changes are properly reviewed and approved.
14. Review Database User Account Privileges: Regularly review the privileges assigned to database user accounts to ensure that they are appropriate and necessary.
15. Monitor and Update Security Measures: Continuously monitor security measures and update them as needed to respond to new threats and vulnerabilities.
16. Test Prepared Statements and Parameterized Queries: Regularly test prepared statements and parameterized queries to ensure that they are properly rejecting hacker requests.
17. Penetration Testing: Perform regular penetration testing to identify any potential security weaknesses and address them proactively.
18. 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.

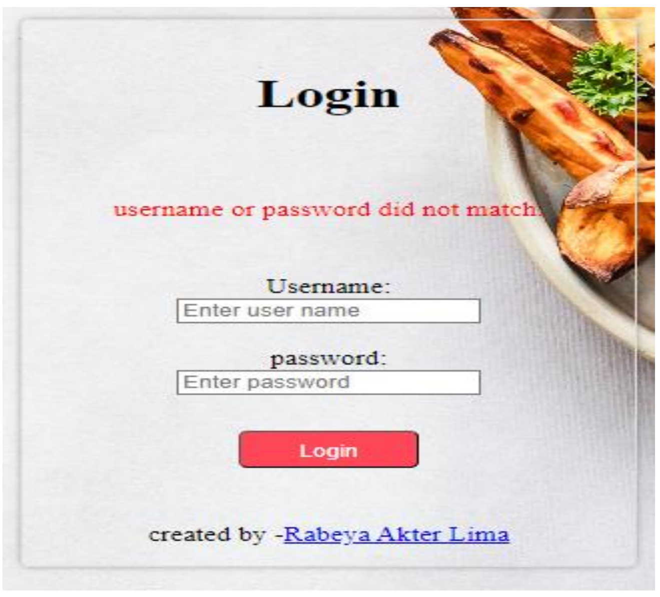
SQL Injection Attack (SQLIA) is a frequent and a severe security issue in the web applications. In SQLIA, hacker can obtain the benefit of poor input validation and weak coded web application. Due to the successful execution of a SQLIA, integrity and confidentiality of data are lost which results in the degrading organization's market value. This paper gives a valuable analysis of various types of SQLIAs, methods and mechanisms. It also explores various detection and prevention techniques.

5. 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:

1. Cost-effective: The algorithm leverages existing technologies and focuses on simplicity, making it a cost-effective solution for organizations with budget constraints.
2. User-friendly: Its straightforward design facilitates easy implementation and maintenance, reducing the need for extensive training or specialized expertise.
3. 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.



username or password did not match

created by -Rabeya Akter Lima

Figure 6: insertion prevention

The input fields will be filled via the web application. 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:

1. The (Host, Database User, Database Password, And Database Name) Have Been Initialized.

2. //Connect Web Server to 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 $_POST['username']);$raw_password = md5($_POST['password']); $password = mysqli_real_escape_string($conn, $raw_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, 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.

Table2: 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

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 norma and intruders.

Table 3: Comparison of detection and prevention methods

	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

According To Akamai [13], However, In The 3rd Quarter Of 2017, The Amount of Internet Application Attacks Improved By 69% Compared to Q3 2016. Whereas SQL Injection and Local File Incorporation Attacks Accounted For 85% Of All of These Assaults, XSS Only Accounted For 9%. For The 2017 Data Breach Study of Verizon [14], Just 15.4 Percent of Recorded Events Involving Web Application Assaults, while 29.5 Percent of Breaches Were Suffered by Web Application Attacks. SQL Injection Is the Most Prevalent Attack on The Internet. Accounted For 85% Of Web Application Attacks in Q3 2017 [13]. As Per the Studies Conducted By US-Based Cloud Service Vendor Akamai, Who Discovered in His Internet State of Report [15]. That SQL injection and local filer inclusion attacks accounted for more than 85 percent of the vectors of assault reported. From November 2017 to March 2019, SQL Injection attacks accounted for 65% of web-based attack vectors. and also, in 2024 while most web applications receive at least 4 web attack campaigns per month, some sites are still under attack.

Each website receives 94,057 SQL injection attack requests in one day. E-commerce experiences twice as many SQL injection attacks as other industries. An observed website was attacked 176 days out of 180, 98% of the time. 94,057 is equivalent to 1,567 SQLi attacks per hour or 26 attack requests per minute, on average.

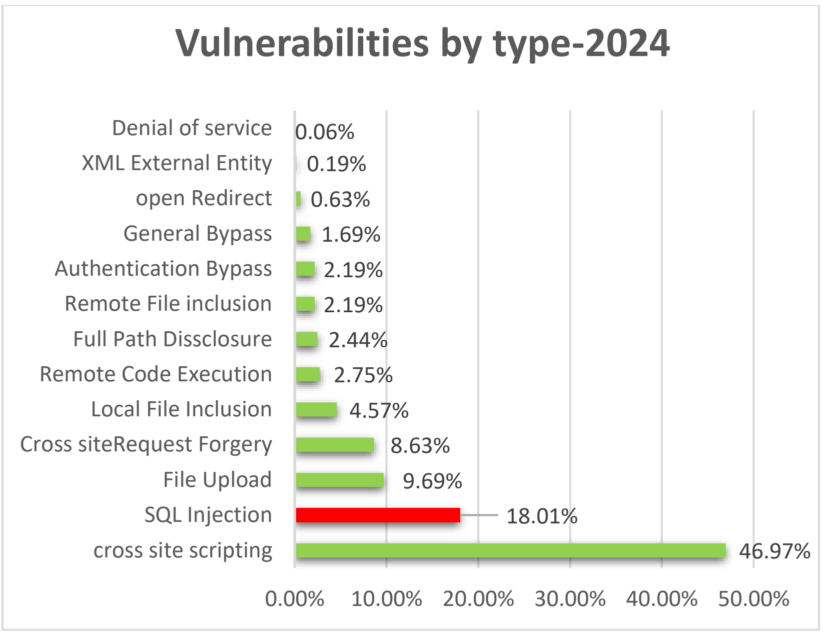
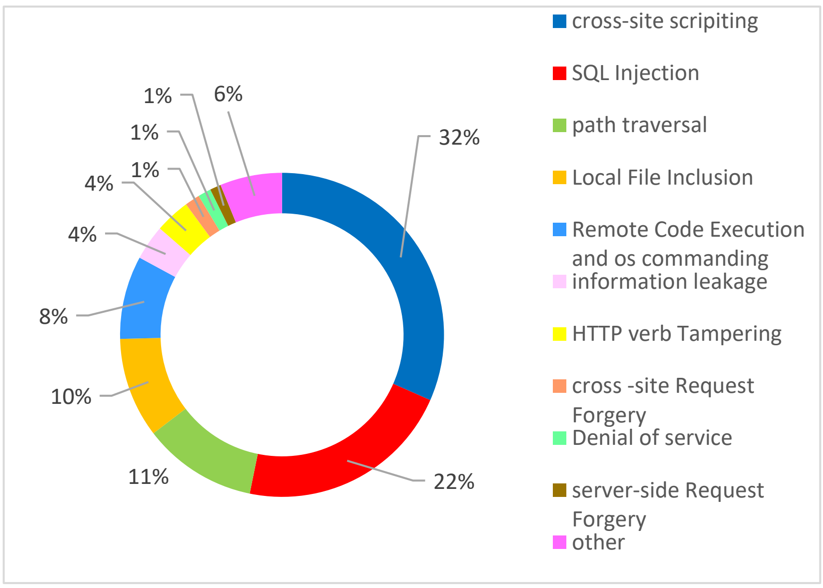


Figure 7: Vulnerabilities by type-2024

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



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.

In Table 5, a comparative assessment is presented, evaluating different techniques like AMNESIA, SQLCHECK, CANDID, Automated approach, Tautology checker, SQLrand, SQLDOM, CSSE, and WebSSARI, against various types of attacks such as tautology, logically incorrect queries, union query, stored procedure, piggybacked queries, inference attack, and alternate encodings. 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: Comparative analysis among detection, prevention techniques and types of attacks

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 sign denotes if technique is able to detect or prevent the attack.
× This sign denotes if technique is not able to detect or prevent the attack.

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

6. 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. The ultimate goal is to eliminate major SQLIA by exploring the effectiveness of pattern matching algorithms such as Brute-force, Rabin-Karp, Boyer-Moore, Knuth-Morris-Pratt, and Aho-Corasick for detecting and preventing these attacks. 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 250 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] 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.

Note: This work was originally written by me back in 2023. While some modifications and line changes have been made, the core content and ideas remain the same. Unfortunately, it appears that someone else has published it under their own name, along with their colleagues, without my consent or proper acknowledgment.