Development of a Framework Model Using Pattern Matching Approach to Detect and Prevent SQL Injection Attacks

Khalid Mohammed Osman Saeed
Faculty of Computer Science and Information Technology
Omdurman Islamic University
Sudan
khalidmohosman@gmail.com

Mohamed Musa Ali Eisa
Faculty of Computer Science and Information Technology
Omdurman Islamic University
Sudan
Mohamed09076@hotmail.com

Tagwa Ismail Khalifa
Faculty of Computer Science and Information Technology
University of Science and Technology
Sudan
Tagwa@ust.edu.sd

Alshafie Gafaar Mahmoud Mohammed
Faculty of Computer Science and Information Technology
Omdurman Ahlia University
Sudan
Alg.mohmmed@gmail.com

Abstract

This article developed a framework model to deal with statements and SQL queries which is compared with a dataset based on pattern matching to prevent and detect whether they are containing SQL injections or not.

The proposed developed framework model exists in between the application interface and the database to verify the statements and which of them is permitted to pass or not; it works by checking the entering statement for any suspicious contents before the statement pass to the database for execution.

The proposed framework model is tested, and the findings indicate that on the false reject rate scale of a 99% of accuracy rate and on the false accept rate of a 98% of accuracy are achieved and the attacker's data is recorded too which can be used for trigger update the dataset.

Keywords: SQL Injection, Security Vulnerabilities, Security Attacks, SQL Statement

Introduction

In the light of the momentous growing of the services offered through internet by organizations and the huge relying on these services; organizations and their customers faced a big challenge to protect their valuable assets from attackers whose continue to develop new techniques for unauthorized access [1]. Therefore, the authors try in this article to develop a framework model for detecting and preventing SQL injection attack.

Data is presented in sort of files or databases, while the security that provided to them should fall into three categories as availability, integrity, and confidentiality. Taking into consideration, most hacking attempts occur through internet [2], so organizations who offered their services through internet, has to deal with their assets and users' data as a sensitive priority need a high level of protection.

There are numerous types of attacks that hackers can use to get inside systems illegally and harm people [3], but from the researcher's opinion SQL injection is the top of them. As well, the jeopardy of it is, once the attacker has illegal access to server, he has the ability to deal with data with administration rights.

If the SQL query is not secure or correct, the attacker can pass through to the database to inject the new query, then SQL vulnerability come to the picture [3], this vulnerability is considered as one of the highest vulnerabilities exists on web [4], that is why, any web application is exposed to such attacks showing how this vulnerability is serious and a huge threat.[5][6][7]

Problem Statement

Enabling users to enter data through inputs makes it easier to hack websites by entering malicious codes such as SQL injection.

Objectives

The objectives of this article can fall into two targets:

- Develop a framework to deal with SQL injection attacks based on pattern matching.
- Recording attacker's data such as the statement used to conduct the attack, time of exploitation, and the attacker's IP address.

Methodology

This article is experimental study, where the system will be developed and tested. Which is the basic idea behind the proposed developed framework model in this article is; it works based on pattern matching approach (blacklist approach), so before sending the query to the server it checks and verified the entry if it a malicious code free it passed to the server, otherwise it prevented and attacker data is recorded.

Literature Review

In order to achieve this article aims, the authors reviewed previous studies regarding prevention of SQL injection attacks and its vulnerabilities. So, to obtain permissions attackers can upload an executable script also commonly known as web shell files to the server [7][8], whereas this known as server-side dynamic scripting security issues. [9][10]

Such attacks occurred when insert a SQL query into input text box in the data form from the client side to database on the server side in order to affect the execution [11], nevertheless, they are numerous types of SQL injections attacks likewise; in band SQL injection, inferential or blind SQL injection, and out of band SQL injection [12][13].

On the contrary, numerous techniques can be used to prevent SQL injection attacks [6], like in the study [14], which explains the causes of SQL injection in PHP code and offers methods for identifying and averting such attack flaws with PHP itself, so this [15] demonstrates a comprehensive encryption strategy to safeguard web application databases by encoding the database using Elliptical Curve Cryptography (ECC) and employing Advanced Encryption Standard (AES) throughout the login process. Yet, numerous approaches are suggested or used to mitigate such attacks, most of them are presented in [16] whilst a comparison and analyzation of them are offered. Furthermore, another technique that uses words in predefine lexicon in order to detect injections by tokenizing the input query, then comparing each string token to the words in the lexicon [13].

While, the paper [17] gives a demonstration of SQL injection attacks and classified them into order wise, blind, and against database, depending on the characteristics of the attack. As well, [12] presents techniques written in PHP to detect and prevent SQL injection vulnerabilities.

In line with [18], analyzes various existing techniques in relation to cost estimation and protection provided. Nevertheless, the study examined each suggested strategy, highlighting its characteristics based on the technology used to detect or block injection attempts on a very small or restricted scale concluding that in spite of the protection they offered but one method for thwarting SQL injection assaults is not enough. Furthermore, and in order to portray reality, the document [19] compares the data collected at the time of the attack, citing examples of such attacks, talking about their repercussions and suggests possible mitigating techniques.

Finally, a well investigation of frequent SQL injection attacks is depicted in [20]. In addition, it proposes a framework to examine the proposed techniques efficiency depending on the type of attack using a statistical, dynamical and machine learning with a small data set for training and testing. See table 1.

Table (1): Summary of Previous Studies

Author/ Year	**Techniques Used to Prevent SQL Injection**	**Limitation**
Yunus, M. A. M. 2018., et al.	Blockchain concept is used on DBMS via IP	Depends on IP address in the recognition
Zhang, H., and Zhang, X. 2018	The techniques used are using php language	Depend on checking the values passed to the URL by get or post request
Gupta, H. 2019., et al.	Use (AES) at login phase and (ECC) to encrypt the database	This technology depends on the encryption and decryption process which it takes more time to implement

Alsahafi, R. 2019	Different techniques such as AMNESIA, SQLrand, SQL-IDS, Sania, SDrive and A heuristic-based approach are used	The techniques advantages are mentioned and Limitations, and did not explain with examples
Hlaing, Z. C. S. S., and Khaing, M. 2020	Different types of attacks are presented as dangers web applications threats	The techniques used are mentioned with an example of each type given
Joe, F. Y., and Selvarajah, V. 2021	The techniques used have been categorized as Order wise, Blind, and Against DB	The techniques used are not mentioned
Kareem, F. Q.2021., et al.	The techniques used depend on post, get, and cookie	It only depends on the input passed on the URL
Alenezi, M.2021., et al.	Relied on hybrid analysis	No example used was mentioned
Oreku, G. S. 2022	Different techniques are used	Did not mention the techniques used
Deriba, F. 2022., et al.	Machine learning to develop a framework	The input goes through a lot of checking stages causes a delay in the process

The Proposed Model

The proposed framework model consists of four steps. The first step begins with reading the input from the input form, the second step in which the entry is classified as number, string, or email, then passed to the third step for checking for injections with the dataset relied on pattern matching, if the statements free of injections it passed to the fourth step for execution but if it contains injections it prevented and the attacker data is recorded, the fourth step, the statement is free of injections and will be executed.

In order to check for the existence of an injections in the query or statement, the authors have developed a tool that has the ability to compare the inserted statement with the dataset which is read it from the file “listSqli.txt” which contains different types of injected statements into the array and compare them all based on pattern matching (see pseudo code 1).

```
$listSqli = array('select *','select','union all
','union','where','and 1','\'and','\'1','\'or','1=1','2=2',';',"'",'"','--
','/','\\','%00','/*','+','%','false','true','order','by','group','#','/*','v
ersion','into','sum','1,2','--+','1,2,3--+','2--+','3--
+','rows','admin\' or \'1\' = \'1','admin','admin--',';--
','\"','<','>','\';','create','shutdown','restart','sleep','delete','drop','
insert','order by','sum');
```

Pseudo Code (1): Validate String

Then, if the entry is number to check it the authors has used “intval” function in PHP language (see pseudo code 2).

```
if(intval($_POST['number']))
        echo 'yes number';
else
        echo 'not number';
```

Pseudo Code (2): Validate Number Using intval Function

Hereafter, to check if the entry is email first it needs to verify that it is in the correct format whereas PHP language has three styles:
Define the input type as Email (see pseudo code 3).

```
<input   type="email"   name="email"    value="" />
```

Pseudo Code (3): Validate Email Using Field Data Type

Use (filter_var) function (see pseudo code 4).

```
$email = test_input($_POST["email"]);
if (! filter_var ($email, FILTER_VALIDATE_EMAIL)) {
$emailErr = "Invalid email format";}
```

Pseudo Code (4): Validate Email Using filter_var Function

Once again, the authors used a function named (preg_match) to check the entry with regular expressions and return a value true or false (see pseudo code 5).

```
function emailValid ($str) {
return (!preg_match("^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-
9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3})$^", $str))
? FALSE: TRUE;}
```

Pseudo Code (5): Validate Email Using preg_match Function

The suggested framework model steps (see figures 1 and 2) could be summaries as follows:

- Filling out the form with the entry if its value or sentence is the first step.
- The second step, entry will be classified as either numeric, email, or a string of letters.
- The entry will be checked based on pattern matching against a dataset containing a comprehensive set of SQL injection statements and indicators such as select, -, union, sleep, etc., the framework tokenizes the entry and matches each token with the items in the dataset, if the entry contains SQL injection, the third step will prevent it and record

the attacker's information (statement used, time of exploitation, and attacker's IP address); if not, the statement moves on to the fourth step.

- The query will be executed directly in the fourth step. Refer to pseudo code (6).

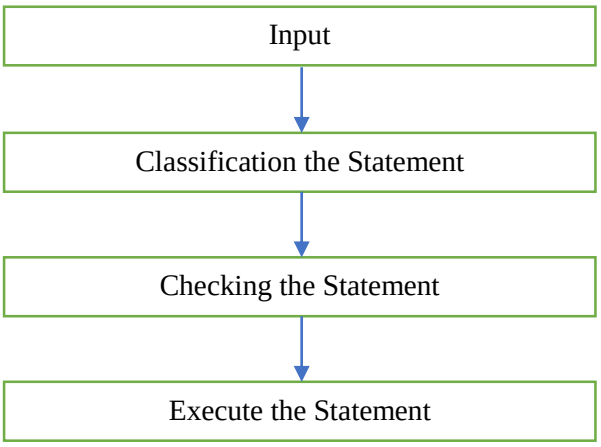


Figure (1): Proposed Framework Model Steps

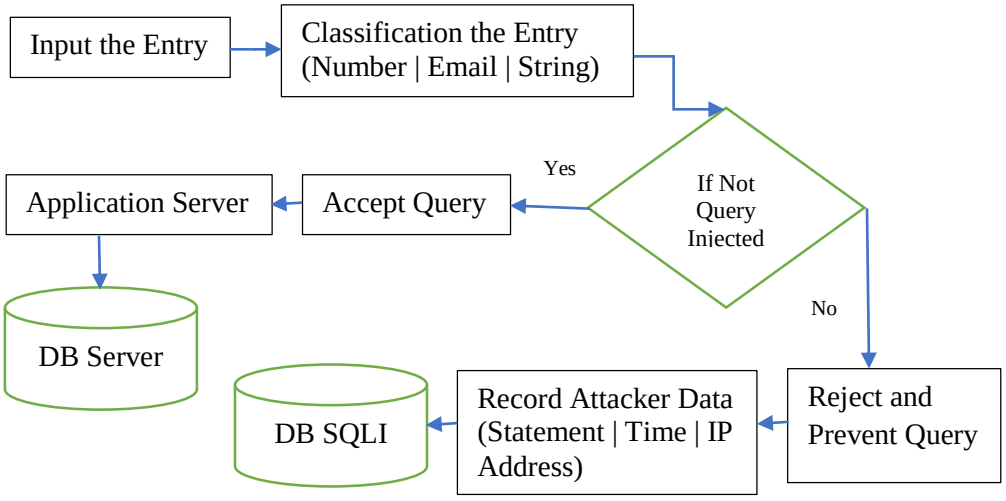


Figure (2): Proposed Framework Model Flaw Chart

```
Start
Input Accept the Entry
Entry Classification as (Number | E-mail | String)
Read the Dataset File
If SQL Injection (Pattern Matching)
        Prevent and Record Attacker Information
Else
        Pass to the Database for Execution
End
```

Pseudo Code (6): Proposed Framework Model Steps

Comparative Analysis

The proposed framework model employs a custom keyword/symbol list to validate input data, dividing it into three types (text, email, number) and utilizing dedicated functions (sqlValid, emailValid, numValid) for each type. Nevertheless, the attack keyword or symbol list dataset contains a comprehensive set of SQL injection statements and indicators such as select, -, union, sleep, etc., the framework tokenizes the input string and matches each token with the items in the dataset. This proposed framework model approach differs from the traditional methods that rely on manual or simple output-based verification [21], so the key differences between the two approaches can be summarized as:

- The proposed framework model has specialized list improves detection accuracy of multiple SQL injection patterns, while simple validation often fails in unexpected or encoded input cases.
- Allowing dynamic updates to the injection pattern list through a user interface increasing updatability of the proposed framework model and making it adaptable to new attack techniques without deep code modification.
- Logging attacker information by the proposed framework model, providing a valuable audit trail for forensic investigation and scientific analysis.

The proposed framework model validates inputs at the application layer before forming SQL queries, representing an application-level defensive approach. In contrast, other approaches such as prepared statements and/or stored procedures their defenses mechanism at the database layer, thereby the key differences between the two approaches here can be summarized as:

- The proposed framework model provides a solid first line of defense in depth, while prepared statements offer another security level at the database layer.
- Integration into existing applications as a validation package, making the proposed framework model easy for implementation and beneficial for systems where secure query rewriting is not possible.

Install the Framework

To use and install the framework model, transfer it to the web server and set it alongside the site files to check for SQL injection statements based on pattern matching against a dataset containing the injection statements before passing the free statement to the database for execution.

Testing and Results

With the purpose of testing the framework model the authors were used a dataset file from [11] containing different types of injection statements or injected queries (see table (1)), then the framework is tested with 150 regular accepted query statements.

Table (1): Regular Expressions for SQL Injections

Number	SQL Inj. RegExps
1	AND
2	ORDER BY
3	SLEEP
4	*
5	'1' = '1'
6	MAKE_SET
7	SELECT
8	DROP
9	OR
10	EXTRACTVALUE
11	UNION
12	WAITFOR
13	INSERT
14	SHUTDOWN
15	UPDATE
16	CREATE
…….	…….

The findings indicate that 147 of the statements were reported as SQL injection free, however 3 were identified as SQL injection statements with a 98% of non-false reject rate as accuracy rate and a 2% of false reject rate [22]. See figure (3).

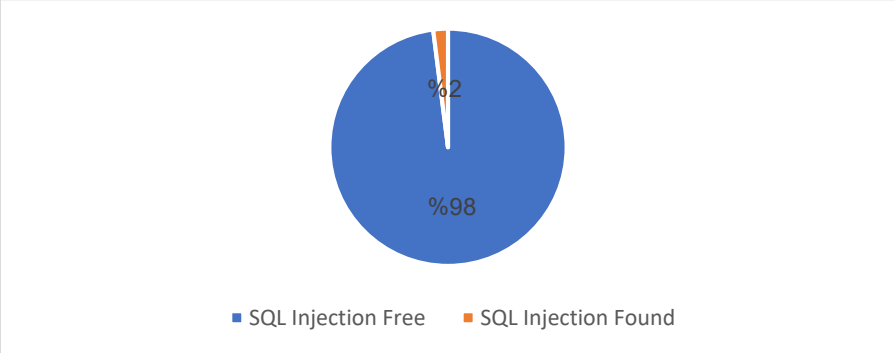


Figure (3): False Accept Rate

Conversely, when 150 SQL injected statements are utilized once more, the findings indicate that a 149 of them include SQL injections that are prevented and attacker's data are reported, while 1 statements are classed as SQL injection free with a false accept rate [22] of 1% and an accuracy rate of 99% as non-false accept rate (see figure (4)). See table (2).

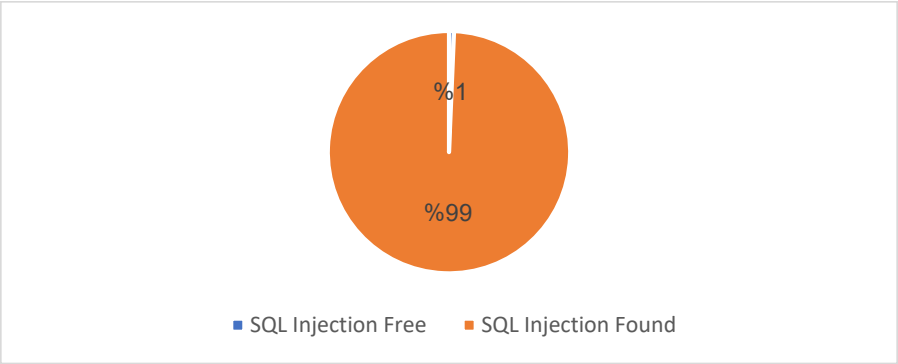


Figure (4): False Reject Rate

Table (2): FRR, NFRR, FAR, and NFAR

False Reject Rate: Incorrectly Classified as SQL Injected Statement	**False Accept Rate:** Incorrectly Classified as Non-Injected SQL Statement
Non-False Reject Rate: Correctly Classified as Injected SQL Statement	**Non-False Accept Rate:** Correctly Classified as Injected SQL Statement

Finally, the lower the values of false accept rate and false reject rate the more accurate the proposed framework model is.

Discussion the Results

The proposed framework model demonstrates a high level of effectiveness in detecting and preventing SQL injection attacks. The results show that the model achieves a 99% accuracy rate in rejecting injected SQL statements and a 98% accuracy rate in accepting legitimate SQL statements. These findings suggest that the pattern matching approach used in the framework model is robust and efficient in identifying potential SQL injection attacks.

The proposed framework model ability to record attacker data, including the statement used, time of exploitation, and IP address, provides valuable visions for security analysts to track and mitigate potential threats. This feature can also be used to update the dataset of injection statements which is improving the model's detection capabilities over time, moreover, the use of a dataset-based approach enables the model to learn from new attack patterns and improve its detection capabilities.

One of the key advantages of the proposed framework model is its flexibility and scalability; however, it can be easily integrated with existing web applications, and its modular design allows for straightforward modifications and updates.

Overall, the proposed framework model offers a promising solution for detecting and preventing SQL injection attacks and its high accuracy rates, flexibility, and scalability make it a valuable tool for web application security.

Conclusion

This article developed a framework model that provides a practical and layered security solution for SQL injection prevention and detection, particularly for legacy codebases using pattern matching approach, whereas the entries are checked against a dataset containing different statements containing injections; besides, the framework is capable to capture the attacker's information, including the statement he/she used, the time he/she attempted the exploitation, and his/her IP address.

In addition to provide a thorough report on the attacker's data, the authors advise including alternatives in the future to retrieve further information on them, such as the device name, operating system, and browser type.

Furthermore, any improvements need to be done to the proposed framework model can be done easily with slight modifications, likewise to improve the matching process the dataset that containing the injection statements can and needs to be updated on a regular basis, which the attacker's data that is recorded by the proposed framework model could be in line with this purpose.

Future Work

While the proposed framework model shows promising results, there are some limitations and areas for future research. One potential limitation is the need for regular updates to the dataset of injection statements to ensure the model remains effective against new types of attacks.

Despite the fact that, the proposed framework model reliance on pattern matching may not be effective against highly sophisticated or novel attacks; future research could focus on improving the model's ability to detect and prevent SQL injection attacks by incorporating machine learning techniques, using more advanced pattern matching algorithms, exploring new approaches to updating the dataset, and integrating the model with other security measures. Additionally, the model's ability to record attacker data could be expanded to include more detailed information.

Reference

[1] Samonas, S., & Coss, D. (2014). The CIA strikes back: Redefining Confidentiality, Integrity and Availability in Security. Journal of Information System Security, 10(3). Information Institute Publishing. Pp. 21-45.

[2] Kumar, K., Jena, D., & Kumar, R. (2013). "A Novel Approach to Detect SQL Injection in WEB Applications", International Journal of Application or Innovation in Engineering & Management (IJAIEM), 2(6), Pp. 37-48.

[3] Halfond, W. G., Viegas, J., & Orso, A. (2006). A Classification Of SQL-Injection Attacks and Countermeasures. In Proceedings of The IEEE International Symposium on Secure Software Engineering. 11(4). IEEE. Pp. 13-15.

[4] "Top 10 Web Application Security Risks", https://owasp.org/www-project-top-ten/, visited on 9-October-2024

[5] Wei, K., Muthuprasanna, M., & Kothari, S. (2006). Preventing SQL Injection Attacks in Stored Procedures. In Australian Software Engineering Conference (ASWEC'06). IEEE. Pp. 191–198.

[6] Yunus, M. A. M., Brohan, M. Z., Nawi, N. M., Surin, E. S. M., Najib, N. A. M., & Liang, C. W. (2018). Review of SQL Injection: Problems and Prevention. JOIV: International Journal on Informatics Visualization, 2(3-2), Pp. 215-219.
[7] Wu, H., & Zhao, L. (2015). Web Security: A White Hat Perspective. CRC Press.
[8] "Types of Cyber Security Threats", https://onlinedegrees.und.edu/blog/types-of-cyber-security-threats/, visited on 20-October-2024
[9] "Web Security Threats in Network Security", https://www.fortinet.com/resources/cyberglossary/web-security-threats, visited on 28-November-2024
[10] "What Is Spyware", https://www.veracode.com/security/spyware, visited on 28-November-2024
[11] "Common Types of Website Vulnerabilities", https://www.indusface.com/blog/what-is-a-website-vulnerability-and-how-can-it-be-exploited/, visited on 30-November-2024
[12] Kareem, F. Q., Ameen, S. Y., Salih, A. A., Ahmed, D. M., Kak, S. F., Yasin, H. M., ... & Omar, N. (2021). SQL Injection Attacks Prevention System Technology. Asian Journal of Research in Computer Science, 6(15), Pp. 13-32.
[13] Hlaing, Z. C. S. S., & Khaing, M. (2020). A Detection and Prevention Technique on SQL Injection Attacks. In 2020 IEEE Conference on Computer Applications (ICCA). IEEE. Pp. 1-6.
[14] Zhang, H., & Zhang, X. (2018). SQL Injection Attack Principles and Preventive Techniques for Php Site. In Proceedings of the 2nd International Conference on Computer Science and Application Engineering. Pp. 1-9.
[15] Gupta, H., Mondal, S., Ray, S., Giri, B., Majumdar, R., & Mishra, V. P. (2019). Impact of SQL Injection in Database Security. In 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE). IEEE. Pp. 296-299.
[16] Alsahafi, R. (2019). SQL Injection Attacks: Detection and Prevention Techniques. Int J Sci Technol Res, 8(1).
[17] Joe, F. Y., & Selvarajah, V. (2021). A Study of SQL Injection Hacking Techniques. In 3rd International Conference on Integrated Intelligent Computing Communication & Security (ICIIC 2021). Atlantis Press. Pp. 531-539.
[18] Alenezi, M., Nadeem, M., & Asif, R. (2021). SQL Injection Attacks Countermeasures Assessments. Indonesian Journal of Electrical Engineering and Computer Science, 21(2), Pp. 1121-1131.
[19] Oreku, G. S. (2022). A Study of Online Database Servers: The Case Of SQL-Injection, How Evil That Could Be? Asian Journal of Research in Computer Science, 14(4), Pp. 198-211.
[20] Deriba, F., Salau, A. O., Mohammed, S. H., Kassa, T. M., & Demilie, W. B. (2022). Development of A Compressive Framework Using Machine Learning Approaches for SQL Injection Attacks. PRZEGLĄD ELEKTROTECHNICZNY, 1(7), Pp. 183-189.
[21] C. Bockermann, M. Apel, and M. Meier. (2009). Learning SQL for Database Intrusion Detection Using Context-Sensitive Modelling (Extended Abstract). In Detection of Intrusions and Malware, and Vulnerability Assessment, Edited by Udo Flegel and Davide Burschi, Lecture Notes in Computer Science, Vol. 5587, Springer. Pp. 116–129.
[22] Harold F. Tipton and Micki Krause, Information Security Management Handbook, 6th Edition, CRC Press, 2007.