UTM UNIVERSITI TEKNOLOGI MALAYSIA
INTERNATIONAL JOURNAL OF INNOVATIVE COMPUTING
ISSN 2180-4370
Journal Homepage : https://ijic.utm.my/

An Improved SQL Injection Attack Detection Model Using Machine Learning Techniques

Yazeed Abdulmalik
School of Computing, Faculty of Engineering
Universiti Teknologi Malaysia
81310 UTM Johor Bahru, Johor, Malaysia
Email: Yz.id@yahoo.com

Submitted: 1/3/2021. Revised edition: 6/4/2021. Accepted: 6/4/2021. Published online: 24/05/2021
DOI: https://doi.org/ 10.11113/ijic.v11n1.300

Abstract—SQL Injection Attack (SQLIA) is a common cyberattack that target web application database. With the ever increasing and varying techniques to exploit web application SQLIA vulnerabilities, there is no a comprehensive method that can solve this kind of attacks. Therefore, these various of attack techniques required to establish many methods against in order to mitigate its threats. However, most of these methods have not yet been evaluated, where it is still just theories and require to implement and measure its performance and set its limitation. Moreover, most of the existing SQL injection countermeasures either used syntax-based detection methods or a list of predefined rules to detect the SQL injection, which is vulnerable in advance and sophisticated type of attacks because attackers create new ways to evade the detection utilizing their pre-knowledge. Although semantic-based features can improve the detection, up to our knowledge, no studies focused on extracting the semantic features from SQL stamens. This paper, investigates a designed model that can improve the efficacy of the SQL injection attack detection using machine learning techniques by extracting the semantic features that can effectively indicate the SQL injection attack. Also, a tenfold approach will be used to evaluate and validate the proposed detection model.

***Keywords*—Web application security, database security, SQL injection attack**

I. INTRODUCTION

SQL Injection Attack, known as SQLIA is one of the most common vulnerabilities of web applications. Although SQL Injection Attacks recently lost its first rank on the list of top 25 Most Dangerous Software Errors by the Common Weakness Enumeration (CWE™), it is still the 6th rank on the list of 2020. So far, by using these SQL query vulnerabilities, attackers can access the information on the database for many purposes such as extract, alter or delete the data. Which may cause to disrupt the system. Justin Clarke [1] described the process of SQL Inject as a weakness in the code where the program does not validate the inputs of user before passing them to SQL queries, which makes the attacker can manipulate the code to execute it on the back-end database.

Despite those obvious vulnerabilities on database, most of the prevention of injection attack methods depending on the database design phase and on the skills of the developer. So, there is no comprehensive solution will prevent directly SQL injection attacks. Steiner et al. [2] argued that withe the ever amount of research about SQLIA, the only resulting that may mitigate SQLIA is to code defensively. However, these researches not only help the developers by providing effective solutions to prevent the threats of SQLIA but also help them to understand the different types of SQL Injection Attacks to detect and avoid the vulnerabilities.

II. BACKGROUND OF THE PROBLEM

There are various types of SQL Injection Attacks, which are, including but not a limitation to Tautologies, Illegal/logically incorrect queries, Union query and Piggy-backed queries. In this section, each type will be described.

A. SQL Injection Attack based on Tautology

The tautology injection attack is to bypass the authentication and access to the database without a valid username. Also, to identify injectable parameters which means, discover an injection spot to SQL injection attack in the

database and extract the data. The following is the basic command of the Tautologies injection attack:

```
SELECT * FROM users WHERE id = 100 OR 1 = 1 ;
```

The above statement, will make the query is evaluated TRUE since the 1=1 is always true, and return all the data of the table (users), which means the attacker will bypass the user authentication, access to the table and extract the data by using this injection.

B. SQL Injection Attack based on Illegal/logically Incorrect Queries

The intent of this kind of SQL attack is trying to discover injectable parameters and extract data. Because of the ability for identifying the structure of the database and its injectable parameters, it is used in many attacks as the first step. Simply it happens when the application server returns the default error page which is contain information about the vulnerabilities that help programmers to correct their applications and sometimes return name of tables and column. So, attackers exploit this information on error pages to target the database weather by extract injectable code or data.

C. SQL Injection Attack-based UNION QUERY

SQL injection attack-based UNION QUERY is another type of SQL attack that is used for bypassing authentication and extracting data. In this type, attacker exploits the UNION SELECT statement to inject his query after the correct query. The result is, return both of correct and injected query results from a database. Following is example of a Union Query statement:

```
SELECT accounts FROM users WHERE login='' UNION
SELECT cardNo from CreditCards where
acctNo=10032 -- AND pass='' AND pin=
```

Assuming that there is no login equal to'' and this is the fake query that trick the application and continue to execute the second query which is the data of the Credit Cards table, the column "cardNo" of account "10032". [3].

D. SQL Injection Attack based on Piggy Backed Queries

The intent of this type is to extract data, add or modify data, perform denial of service, execute remote commands. In this type, attacker injects the malicious with the correct query that may involve lNSERT, UPDATE and DELETE clause for modifying a record. For example:

```
SELECT * FROM User_info WHERE UserName = n;
lNSERT
lNTO User_info VALUES ('Pknapit','123')
```

When just the table name is known, this statement will insert a record into the table. Also DELETE, UPDTE clauses can be executed by using the same above query. [4].

III. Literature Review

In this section, a review of three common techniques that used for detecting SQL Injection Attack and related to our model proposed has been presented. These detection techniques are, Static Analysis, Dynamic Analysis and combined both them.

A. Static Analysis

Static Analysis is an approach that is used to detect many vulnerabilities not only SQL injection attacks. For SQL injection attack, it analyses the SQL query statement on the web application for detecting and preventing SQL Injection attacks. There are numerous researches that had proposed based on this approach. Wassermann and Su [5] have proposed using the combined automated reasoning and static analysis method in order to prevent SQLIA. The method verifies that there is no tautology in the real-time SQL Query. However, this method only detects the attack-based tautology and cannot detect another type of injection attack. Another method is proposed by Gould et al. [6] is the JDBC checker. This method by using the Java String Analysis library is verifying the inputs in real time to check if there SQL Injection Attack and prevent it. Because of using the Java Script Analysis library in this method, so it is only supported Java. Kosuga et al. [7] proposed a technique that is named SANIA based on static analysis. SANIA analyses the syntax of SQL queries between web application and database to detect where are the SQL injection vulnerable spots, then generate automatically attacks based on this vulnerability and compare the parse trees of SQL queries with the results of that automated attack to validate the portability of attack. The limitation of SANIA that is needed the developers to know the all-SQL queries and HTTP requests on the web application. Fu et al. [8] proposed a static analysis framework called SAFELI for identifying SQLIA vulnerabilities at compile-time, SAFELI can identify the vulnerabilities of the corresponding user input that could lead to a breach of information security. The limitation of this method that it is only detecting SQL injection attack in Microsoft based product.

B. Dynamic Analysis

Dynamic Analysis is another approach used for detecting SQL injection attacks, it detects the vulnerabilities during program execution. Appelt et al. [9] proposed an automated technique called µ4SQLi to detect SQL injection attack based on dynamic analysis, the technique based on a set of mutation operators that manipulate inputs to create new test inputs to trigger SQL injection attacks. Which may generate inputs that contain new attack patterns, and increasing the likelihood of

detecting vulnerabilities. Ciampa et al. [10] proposed an approach and tool called "viper". The approach matching pattern the outputs and error messages, and use the information to craft attack inputs that are more likely to be successful at revealing vulnerabilities. In this method, error messages of web application may lead to not be able to prevent SQL injection attack. Shin [11] proposed the white-box security testing framework that verifying and the inputs in order to detect SQL injection attack. The main limitation in dynamic analysis that the vulnerabilities of web application can be located depending only on previous attacks.

C. Combined Analysis

Combined Static and Dynamic Analysis is the approach that collects the advantages of both static and dynamic approaches and using them together in order to detect and prevent SQL injection attacks. Halfond and Orso [12] proposed a technique called AMNESIA to detecting and preventing SQL injection attacks based on static analysis and dynamic analysis. The static analysis, creating a model of the legitimate queries of the application, and the dynamic analysis by using runtime monitoring check the dynamically-generated queries and match them with the static model. However, some a legitimate query that is like the structure of SQL attack may lead to produce false negatives with this method. ASSIST is another technique that is proposed by Mui and Frankl [13] which using a combination of static analysis and program transformation for automatic query sanitization and prevent SQL injection in code. But this technique also as AMNESIA result a false negative because of imprecision. Dharam and Shiva [14] also proposed a Runtime Monitoring framework that is monitoring the behavior of the application in the post-deployment to detect and prevent SQL injection attack based on Tautology. However, this method only detecting SQL injection attack based on Tautology. Qing and He [15] proposed a method that detect and prevent SQL injection attacks using a model that is generated by using an analysis technique to extract legitimate SQL queries and match it with AOP. However, the main limitation of this method that the source code of the program has to be visible.

Most of the existing SQL injection countermeasures either used syntax-based detection methods or a list of predefined rules to detect the SQL injection. Although such solutions can be suitable for basic SQL injection attacks, they are vulnerable to advance and sophisticated type attacks. This is because attackers create new ways to evade the detection utilizing their pre-knowledge about the conventional detection and mitigation approaches are based on parsing the SQL syntax.

Although semantic-based features can improve the detection, up to our knowledge, no studies focused on extracting the semantic-based features from SQL statements, while useful patterns can be extracted using machine learning techniques from the SQL queries traffic between the web server and the application server. This research proposed a model to enhance the performance of a hybrid analysis method for detecting SQL injection attack by extracting the semantic-based features using machine learning techniques.

IV. Proposed Model

The proposed model construction for improving the efficacy of detection SQL injection attack consists of three phases, Dataset Phase, Static and Dynamic analysis Phase, and Model Construction Phase. Dataset phase is the previous proposed method which this study is focusing on improving its performance by the proposed model construction. The second phase is the static and dynamic analysis process. Then, the third phase is the construct model. This section covers the details of each phase of the proposed model.

Phase I: Dataset

This phase is the previous method for detection SQLIA proposed by Ghafarian [16] which is combined between static and dynamic analysis method. Firstly, it suggested to insert a record at every table of the database that has only Symbols. Table 1 shows an example for this proposed database design.

TABLE 1 Example of proposed design of Database table from [16]

ProductID	ProductName	Constructor
101	%$%	%$%
102	Pen	Stabilo
103	Pencil	Steadtler
104	Eraser	Artline

Also, it proposed an algorithm that implementing at the business logic layer of the web application which is an external program that will be applied in CGI interface for monitoring all the input queries before passing the request to the database.

The proposed algorithm:

"

```
{
Input_string = SQL Query ;
Output_string =Null;
Table_Name=get_table_name(Input_String);
Where_String=get_where_Query(Input_String);
Output string="select *
from"+table_Name+"where"+where_string;
DataTable=Exec_Query(Output_string);
X=find(DataTable,"$%$");
If(x==1) Return "Not Valid";
Else Return "Valid";
}
```

"

Phase II: Static & Dynamic Analysis

This phase is where the static and dynamic analysis take a place at the database access layer. By using the result that

getting from the algorithm and match it with the database that has the proposed design in phase I, so that will detect any types of SQL injection attack.

Phase III: Model Construction

In this phase, the semantic features that were extracted from the dynamic and static analysis will be combined and used to construct a model using machine learning techniques. Many algorithms will be tested such as the Random Forest algorithm (RF), Artificial Neural Network (ANN), support vector machine (SVM), and logistic regression (LR). The dataset which contains two types of SQL queries namely the benign and malicious SQL query will be used. A tenfold approach will be used to evaluate and validate the proposed detection model. Figure 1 shows the block diagram of the proposed SQL detection model.

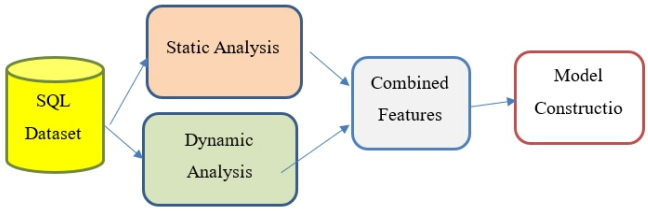

Fig.1. Model Construction of improve the efficacy of SQL injection attack detection

V. Performance Measurement Plan

Measuring the method performance is based on two factors that will take into consideration in evaluating the method and defining its limitation which are Detection Overhead and Error Rate. Following is discerption each factor:

1) Detection Overhead

Detection Overhead is the time that the method takes until detecting SQL attack which is called Detection Overhead.

2) Error Rate

Measuring the error rate of SQL injection attack methods is depending on two common issues which are, False-positive and False-negative. False-positive is to define correct request as SQL attack. False-negative is to define the SQL attack as a normal request and could not detect it.

VI. Evaluation and Implementation Plan

The evaluation and implementation plan are to create a testbed that included insecure web applications which is vulnerable to SQL injection attacks and databases, then perform a list of SQLIA without using the model. After that, re-perform the SQL injection attacks with the previous proposed method and with the model construction. Finally, the proposed model will be evaluated by reporting the results based on the performance measurement that have mentioned in the previous section which are, error rate and detection overhead. Figure 2 shows the implementation plan flow.

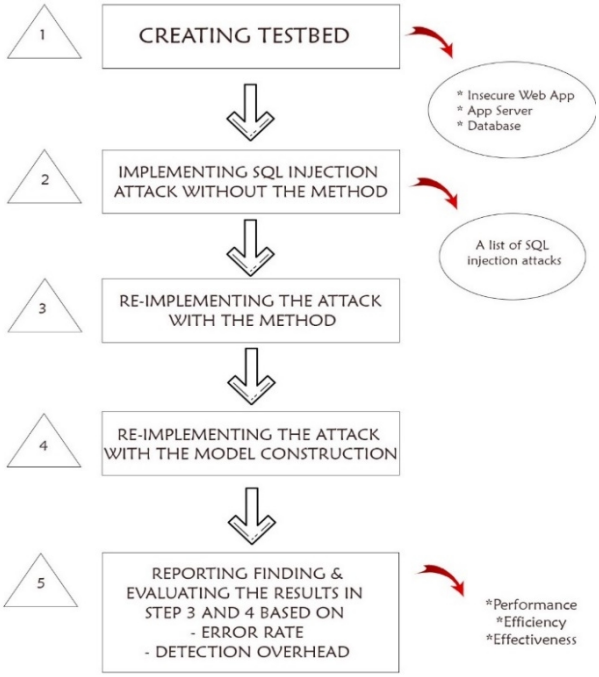

Fig. 2. Flow of implementation and evaluation plan

VII. Conclusion

The aim of this research is to create a model that can increase the accuracy of SQL injection attack detection using machine learning techniques by extracting semantic features that can effectively indicate the attack. The proposed model consists of three phases which are Dataset, Static and Dynamic Analysis, and Model Construction. The evaluation and validation of the proposed model are in progress and the results will be presented in the next publications.

References

[1] Justin Clarke. (2012). *What Is SQL Injection?*

[2] S. Steiner, D. Conte de Leon, and J. Alves-Foss. (2017). A Structured Analysis of SQL Injection Runtime Mitigation Techniques. *Proc. 50th Hawaii Int. Conf. Syst. Sci.*, 2887-2895. Doi: 10.24251/hicss.2017.349.

[3] W. G. J. Halfond, J. Viegas, and A. Orso. (2008). A Classification of SQL Injection Attacks and Countermeasures. *Prev. Sql Code Inject. By Comb. Static Runtime Anal.*, 53.

[4] P. Kumar and R. K. Pateriya. (2012). A Survey on SQL Injection Attacks, Detection and Prevention Techniques. *2012 3rd Int. Conf. Comput. Commun. Netw. Technol. ICCCNT 2012*. Doi: 10.1109/ICCCNT.2012.6396096.

[5] G. Wassermann and Z. Su. (2004). An Analysis Framework for Security in Web Applications. *SAVCBS 2004 Specif. Verif. Component-Based Syst.*, 70. [Online]. Available: http://web.cs.ucdavis.edu/~su/publications/savcbs.pdf%0Ahttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.72.2255&rep=rep1&type=pdf#page=82.

[6] C. Gould, Z. Su, and P. Devanbu. (2004). JDBC Checker: A

Static Analysis Tool for SQL/JDBC Applications. *Proc. - Int. Conf. Softw. Eng.*, 26, 697-698. Doi: 10.1109/icse.2004.1317494.

[7] Y. Kosuga, K. Kono, M. Hanaoka, M. Hishiyama, and Y. Takahama. (2007). Sania: Syntactic and Semantic Analysis for Automated Testing Against SQL Injection. *Proc. - Annu. Comput. Secur. Appl. Conf. ACSAC*, 107-116. Doi: 10.1109/ACSAC.2007.20.

[8] X. Fu, X. Lu, B. Peltsverger, S. Chen, K. Qian, and L. Tao. (2007). A Static Analysis Framework for Detecting SQL Injection Vulnerabilities. *Proc. - Int. Comput. Softw. Appl. Conf.*, 1(August), 87-94. Doi: 10.1109/COMPSAC.2007.43.

[9] D. Appelt, C. D. Nguyen, L. C. Briand, and N. Alshahwan. (2014). Automated Testing for SQL Injection Vulnerabilities: An Input Mutation Approach. *2014 Int. Symp. Softw. Test. Anal. ISSTA 2014 - Proc.*, May, 259-269. Doi: 10.1145/2610384.2610403.

[10] A. Ciampa, C. A. Visaggio, and M. Di Penta. (2010). A Heuristic-based Approach for Detecting SQL-injection Vulnerabilities in Web Applications. *Proc. - Int. Conf. Softw. Eng.*, January, 43-49. Doi: 10.1145/1809100.1809107.

[11] Y. Shin. (2004). Improving the Identification of Actual Input Manipulation Vulnerabilities, 1-4.

[12] W. G. J. Halfond and A. Orso. (2005). AMNESIA: Analysis and Monitoring for Neutralizing SQL-injection Attacks. *20th IEEE/ACM Int. Conf. Autom. Softw. Eng. ASE 2005*, 174-183. Doi: 10.1145/1101908.1101935.

[13] R. Mui and P. Frankl. (2010). Preventing SQL Injection through Automatic Query Sanitization with ASSIST. *Electron. Proc. Theor. Comput. Sci.*, 35, 27-38. Doi: 10.4204/eptcs.35.3.

[14] R. Dharam and S. G. Shiva. (2012). Runtime Monitoring Technique to handle Tautology based SQL Injection Attacks. *Int. J. Cyber-Security Digit. Forensics (IJCSDF)*, 1(3), 189-203,

[15] W. Qing and C. He. (2016). The Research of an AOP-based Approach to the Detection and Defense of SQL Injection Attack, 731-737. Doi: 10.2991/aest-16.2016.98.

[16] A. Ghafarian. (2018). A Hybrid Method for Detection and Prevention of SQL Injection Attacks. *Proc. Comput. Conf. 2017*, 833-838. Doi: 10.1109/SAI.2017.8252192.