SQLID Framework In Order To Perceive SQL Injection Attack on Web Application

Bandi Aruna and Bhimavarapu Usharani

K.L.E.F, Green Fields, Vaddeswaram, Guntur Dist, AP, India

E-mail: bandiaruna@kluniversity.in

Abstract. The opportunity to admittance the web anytime is great help to the technical field. Web is more common, attacks on web also growing. SQL injection is a risk to servers like military and banking systems. SQL injection deals with to retrieve or insert information from database.

Malicious users bind the malicious SQL command to the original web application query and make it run on the server. By successful execution of these malicious queries, the illegitimate users can insert, delete, update, execute and read the schema details and other confidential information from the database. It presents a novel and fast technique to detect SQL injection attack.

To reduce the resources available to counter the injection attacks, we suggested a new system.

Keywords: SQL Injection, signature checking, SQL query checking, malicious SQL command, Evasion techniques etc.

1. Introduction

Most people are making online transactions with speedy development of web technology. There are numerous security issues in the web applications. The primary objective of the SQL injection attack is to circumvent authentication and retrieve confidential and sensitive information to change the database data, to execute the remote malicious code and to take control over the server. SQL injection attack changes the normal SQL query into the malicious query. The SQL injection can be done by two ways:

1. Through users input 2. Through URL

SQL injection attacks make it possible to monitor the database secretly or illegally, resulting in sensitive information leaks, database deletion, network hacking, financial fraud. Many individuals and many of them are unaware of data protection and privacy. The unauthorised persons add the malicious code to the web applications due to the high versatility of the SQL language and collect the confidential information such as credit card numbers from the online shopping websites and collect the critical banking system information. Because of the high flexibility of the SQL system, the malicious users change the appearance of the attack and easily bypass the detection system. The malicious users append the malicious SQL command to the web application original query and make them execute in the server.

Then the illegitimate users can run many malicious codes on the operating system, even execute many malicious codes and launch the new attacks on both the server and the database.

If the malicious SQL query evaluates to true, then the web application continues its functioning normally. A successful SQL injection attack allows the illegitimate user can access the original SQL
query and gain the privilege access and retrieve the authorized data from the database.

If the malicious SQL query is evaluated as incorrect, the web application may behave abnormally and may stop running. These error messages also assist the illegitimate users, they come to know the column names and data types and table names also.

The extreme level of content is organized: related work provides a brief general idea of the state-of-the-art techniques and discussed in segment 2. The future method is discussed in the segment 3. Segment 4 provides the tentative setup and result analysis. Finally, the Conclusion section bring up the future span.

2. Literature Review
The most well-known web attacks contain SQL Injection, denial of service, phishing, spamming, URL misinterpretation etc. It was a light and fully automated device for SQL Injection Prevention by Contribution labeling, through arranged statements to avoid SQL Injection attacks at run time. Some state to the art techniques to detect the SQL Injection attacks are described in Table 1.

Table 1: State of the art techniques for the SQL Injection Detection

| Project          | Description                                                                                   | Dataset                                                                                      | Classifiers / Tools                                                                 |
|------------------|-----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| Das et al[1]     | Detection of dynamic SQL vulnerability. Efficient in managing several runtime queries based on input data from users and their SQLIA identification vulnerabilities. | Authors prepared synthetic dataset from web applications with multiple structures             | Naïve Bayesian, SVM, MYSQL Server, LINUX server                                     |
| Li et al[2]      | SQL injection attack detection supported on LSTM (long short-term memory). SQL injection attack detections based on machine learning. Static AND dynamic techniques to detect and block SQL injection attacks. | Exploit, tamper scripts on certain websites.                                                  | SVM, K-Nearest Neighbour, Tensor flow                                               |
| Medeiros et al[3]| Protection mechanisms are introduced inside the programming libraries and the operating system | 66 code samples of web application                                                             | Spring framework                                                                    |
| Aliero et al[4]  | Automatic black box testing used to ease the cost of manual inspection susceptibilities and the risk | 3 custom web applications                                                                     | Vega, zap, nikito, wapiti, acunetix, wvs, w3, AF, appscan                           |
| Lee et al[5]     | Using a mixture of SQL parameters and a combination of static and dynamic analysis Techniques..SQL injection attacks on the web and any database-related applications | 5 types of web applications                                                                    | Paros attack tool                                                                   |
| Gillman et al[6] | SQL injection is prevented by type checking the inputs.                                       | Web host(Akamai)                                                                              | DNS servers, edge servers                                                          |
| Le et al[7]      | SQL injection takes the structure of web request and database queries without looking into the values of the input parameters. Double guard forms a container-based intrusion system. | Static testing website, Dynamic Blog, 1000 artificial traffic sessions                           | attack tools, SQMAP, NIKTO web server, Metasploit                                   |
Attacker

Insert malicious SQL statements

SQLID Framework in Web Application

Retrieve Secure Information

3. Proposed Methodology
The following methodology provides an overview of the planned approach and explains how to detect the SQL Injection attacks.

The possible SQL injection evasion techniques contain White space, Null bytes, SQL Comments, URL encoding, character encoding, Hex encoding, declaring variables, Tautologies, Arbitrary String pattern, Stored procedure, and alternate encoding. In repetitions, the attacker uses the hexadecimal values, AND operator, OR operator, LIKE operator, NULL, union, union all, space to evade the signature checking. In arbitrary string pattern technique, the attacker uses c comments, union to escape from the detection of malicious code. In group concatenate string the intruders use concept, group CONCAT, OR operator, Like operator to protect from the detection. Stored procedure is the remote and dos attack and it employs the stored procedure. Alternate encoding technique uses the different signatures like ASCII(), DEC(), HEX() etc. In the SQLID framework we are checking all these different signatures to detect SQL Injection attacks. The detailed SQL Injection Detection algorithm is given below:

**SQL Injection Detection Algorithm**
**Input**: Dataset (CSV file)  
**Output**: Classification of Malicious and Benign SQL Queries  
**N**: Count of the word in each line

| A1 = list of malicious signatures |
|-----------------------------------|
| num_lines = 0                     |
| open CSV file:                    |
| for line in f:                    |
| num_lines += 1                    |
| f.close()                         |
| n = num_lines                     |
| a = initialize with 0’s           |
| for i in (n):                     |
| for j in (0, n):                  |
| a[i][j] = 0                       |
| open CSV file:                    |
| data = f.readlines()              |
| l, w = 0, 0                       |
| for line in data:                 |
| l = l + 1                         |
| words = line.split()              |
| for x in words:                   |
| w = w + 1                         |
| for x1 in a1:                     |
| check the signatures in different cases |
| for i in (l, l + 1, 1):           |
| for j in (w, w + 1, 1):           |
| a[i][j] = 1                       |

for row in a:  
count and join the elements in the array
The detection techniques that we used in the SQL Injection algorithm is the checking the line comments inside the query, checking concatenation, checking for the obfuscate string, checking for the encrypted string, ASCII encoding, and hexadecimal encoding. From the SQL Injection detection algorithm, we can calculate the execution time that gives approximate \( O(\log N) \), where \( N \) is the count of the characters at each line in the dataset. The algorithm maximum time is proportional to the logarithm of the input size. It is the much faster algorithm.

4. Experimental Results

4.1 Dataset
We have collected the dataset from HTTP CSIC dataset. We stored the data in the CSV file and take this CSV file as the input. We implemented the proposed EGMSA algorithm with Python and we developed the code in Jupyter Notebook in Windows 10 platform.

4.2 Performance Measures
We evaluate the SQL Detection Algorithm for the dataset HTTP Dataset CSIC 2010. This data set consists of separate sets for training and testing. The accuracy of the results can be measured based on the true positive(TP), False Positive (FP), False negative(FN), True Negative(TN). The confusion matrix can be represented as below:

| True Positive classified as injected statement | False Positive classified as non-injected SQL statement |
|-----------------------------------------------|--------------------------------------------------------|
| False negative - Incorrect classification of injected SQL statement | True Negative - correct classification of an non-injected SQL statement |

Table summarizes the results of the detection algorithm.

| Dataset   | TP  | TN  | FP  | TNR | FN  | Precision | specificity | accuracy |
|-----------|-----|-----|-----|-----|-----|-----------|-------------|----------|
| Dataset1  | 650 | 74  | 2   | 97  | 4   | 99        | 96          | 99       |
| Dataset2  | 638 | 97  | 3   | 97  | 2   | 99        | 99          | 99       |
| Dataset3  | 2554| 883 | 30  | 97  | 7   | 98        | 98          | 98       |
| Dataset4  | 1320| 395 | 16  | 96  | 5   | 98        | 98          | 98       |
| Dataset5  | 2070| 491 | 18  | 96  | 5   | 99        | 99          | 99       |
| Dataset6  | 2771| 880 | 35  | 96  | 7   | 98        | 98          | 98       |

\( TNR=TN/(TN+FP) \)

\( \text{Precision}=TP/(TP+FP) \)

\( \text{Recall}=TP/(TP+FN) \)

\( \text{Accuracy}=(TP+TN)/(TP+TN+FP+FN) \)

The table below depicts the ROC curve for the recital of the SQLID Framework outcomes. The ROC curve shows the relation between the values TPR and the FPR.
Figure 2: ROC curve

| classifier    | Accuracy |
|---------------|----------|
| Random Forest | 99.1%    |
| K-NN          | 98.3%    |
| SVM           | 98.5     |
| Bayes-Net     | 98.7%    |
| Naive Bayes   | 95.6%    |
| RBF           | 97.3%    |
| SQLID         | 99.0%    |

5. Conclusions.

In digital era, securely maintaining the information is essential despite security is a challenge. Phishing attacks are inevitable. Fraudsters play tricks to gather individual or financial information of legitimate users. Here the present framework for detecting SQL Injection attacks. SQL Injection detection framework mitigate the SQL Injection attacks are analyzed.
References.

[1] Das, Debasish and Sharma, Utpal and Bhattacharyya, DK “Defeating SQL injection attack in authentication security: an experimental study”, International Journal of Information Security, Vol.18, No. 1, pp.1—22, 2019

[2] Li, Qi and Wang, Fang and Wang, Junfeng and Li, Weishi “LSTM-Based SQL Injection Detection Method for Intelligent Transportation System”, IEEE Transactions on Vehicular Technology, Vol.68, No.5, pp.4182—191

[3] Medeiros, Ibraia and Beatriz, Miguel and Neves, Nuno and Correia, Miguel,” SEPTIC: Detecting Injection Attacks and Vulnerabilities Inside the DBMS”, IEEE Transactions on Reliability, 201

[4] Aliero, Muhammad Saidu and Ghani, Imran and Qureshi, KashifNaseer and Rohani, MohdFo’ad, “An algorithm for detecting SQL injection vulnerability using black-box testing”, Journal of Ambient Intelligence and Humanized Computing, PP.1--18, 2019

[5] Lee, Inyong and Jeong, Soonki and Yeo, Sangsoo and Moon, Jongsub, “A novel method for SQL injection attack detection based on removing SQL query attribute values”, Mathematical and Computer Modelling, Vol.55, No.1-2, pp. 58-68, 2012

[6] Gillman, David and Lin, Yin and Maggs, Bruce and Sitaraman, Ramesh K, “Protecting websites from attack with secure delivery networks”, Vol.48, No.4, pp.26-34, 2015.

[7] Gillman, David and Lin, Yin and Maggs, Bruce and Sitaraman, Ramesh K, “Protecting websites from attack with secure delivery networks”, Vol.48, No.4, pp.26-34, 2015.

[8] Le, Meixing and Stavrou, Angelos and Kang, Brent ByungHoon, “Doubleguard: Detecting intrusions in multitier web applications”, IEEE Transactions on dependable and secure computing, Vol.9, No.4, pp.512-525, 2011.

[9] Sharma, Pankaj and Johari, Rahul and Sarma, SS, “Integrated approach to prevent SQL injection attack and reflected cross site scripting attack”, International Journal of System Assurance Engineering and Management, Vol.3, No.4, pp.343-351, 2012.