Detection of SQL Injection Attacks Based on Improved TFIDF Algorithm

Yingbo Li¹², Bin Zhang¹²
¹ PLA SSF Information Engineering University, Zhengzhou, China
² Henan Key Laboratory of Information Security, Zhengzhou, China
934911469@qq.com

Abstract. Detecting SQL injection attacks (SQLIAs) effectively is one of the critical issues to be solved in Web secure fields. Aiming at the problem that the distribution of feature words in the same kind of statements is not considered when using TFIDF algorithm to vectorize the text of SQL statements, a method of detecting SQLIAs based on improved Term Frequency-Inverse Document Frequency (TFIDF) algorithm is proposed. Firstly, TFIDF algorithm is improved based on the distribution of feature words in the same kind of statements. Then, the improved TFIDF (ITFIDF) algorithm is used to vectorize the dataset of SQL statements to increase the feature weight of the SQL statement. Finally, the detection of SQLIAs is carried out based on Support Vector Machine (SVM). The experimental results show that the combination of SVM and ITFIDF has higher accuracy, recall rate and F-score compared with other similar methods. At the same time, the experimental results also show that SVM has better classification performance when dealing with this problem than other machine learning models such as DNN and Decision Tree.

1. Introduction
As a code injection attack that usually targets the website database [1-2], SQLIAs is one of the most serious threats to current Web security [3]. It utilizes the defects of filtering special strings incompletely by Web applications and carefully constructs strings to achieve the purpose of accessing website database contents illegally or executing commands in database. Just like normal requests, SQLIAs utilizes HTTP protocol to access the websites. Ordinary firewalls can’t intercept them because of there is no difference between attack requests and legitimate requests [4]. When an attack occurs, there are no obvious changes in network traffic and user behaviors, which makes the attack difficult to detect and has strong concealment [5]. Therefore, how to detect SQLIAs effectively is one of the critical issues to be solved urgently in the current Web security field.

At present, the researches on SQLIAs detection are mainly divided into the following three types. The first method is the detection of SQLIAs based on rule matching. It builds a knowledge base by learning all legitimate SQL statements in a secure environment and matches the SQL statements with the knowledge base in a real-time working environment, such as the method based on syntax tree matching [6-7], the method based on clearing the attribute values of SQL statements [8]. This method performs well in real-time environment, but it needs to build knowledge base for legitimate queries. The accuracy of detection depends on the coverage of knowledge base. The second method is the detection of SQLIAs based on static analysis, which allows applications to find SQL injection vulnerabilities by analyzing source programs when they are not running, such as the method of
determining vulnerabilities by marking tags [9] and the method of detecting vulnerabilities based on sensitive flow [10]. This kind of method can find vulnerabilities from the root, but it needs code auditing with the help of human resources. Moreover, the resource investment is too large. The third is the detection of SGLIAs that combines text characterization and machine learning, such as the method based on document similarity matching [11], the method based on feature hash training to implement SVM algorithm [12], the method of SVM based and feature vector extraction of text [13], the method based on improved query normalization and double HMM integration [14]. However, these methods have the disadvantages of low accuracy and unsatisfactory recall rate, mainly because the input vector of the machine learning method is weakly characterized, and it is difficult to obtain the most accurate model. At the same time, using the text vectorization algorithm such as TFIDF algorithm [15] can obtain the dataset with better representation when processing the SQL statement dataset, but when using this algorithm, the slight weak representation of attributes will lead to inaccurate description of keyword weight.

Aiming at the above problems, SQL statement text vectorization algorithm is designed and implemented based on the idea of text vectorization and ITFIDF algorithm. SGLIAs detection is performed on the processed data by using SVM, and the performance of the proposed method is verified by experiments.

2. ITFIDF text vectorization algorithm

TFIDF algorithm is a weighted technology for information retrieval and data mining. It is used to evaluate the importance of a word to a document in a file set. The importance of a word increases with the number of times it appears in the file set. But at the same time it will decrease inversely with the frequency it appears in the file set. Compared with utilizing single attribute term frequency to depict text weights, TFIDF algorithm takes term frequency and inverse document frequency into account comprehensively, which effectively alleviates the problem that the single attribute depicts the text weight inaccurately. When using this algorithm to vectorize the SQL statements, the main basis is as follows. If a word appears frequently in attack statements while rarely in normal statements, it is considered that the word has good ability to recognize abnormal classes. Therefore, the algorithm is used to perform text vectorization on SQL statements based on the sensitive characters and keywords. It can generate feature values with obvious difference between text vectors of attack-type statements and normal-type statements.

However, only the relationship between the feature word and the number of SQL statements containing it is considered when using TFIDF algorithm to detect SGLIAs. While the distribution of the feature word in similar statements is ignored. Therefore, it is difficult to accurately depict the different types of SQL statements by the feature vectors generated by this algorithm, resulting in poor classification performance. To this end, this paper proposes an ITFIDF method that can more accurately depict the feature vectors of SQL statement texts. The TFIDF algorithm and the ITFIDF algorithm are introduced separately as follows.

2.1. Review of TFIDF algorithm [15]

TFIDF algorithm is denoted by $TF^*IDF$, where $TF$ denotes term frequency and $IDF$ denotes inverse document frequency. When using this algorithm to vectorize the SQL statements, $TF$ denotes the frequency of the sensitive character $t_i$ appearing in the SQL statement $d_j$, and $IDF$ denotes the inverse document frequency of the sensitive character $t_j$ in the SQL dataset. A sensitive character with a high term frequency in a statement and a low document frequency in the dataset has a high TFIDF value.

The term frequency of each sensitive character in the SQL statement is as formula (1).

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$  \hspace{1cm} (1)
Where $n_{ij}$ denotes the number of times the word $t_i$ appears in the statement $d_j$ and $\sum_i n_{i,j}$ denotes the sum of all the words in the statement $d_j$. The inverse document frequency of each sensitive character in the SQL statement is as formula (2).

$$\text{idf}_i = \log \frac{|D|}{|\{j : t_i \in d_j\}|}$$  

Where $|D|$ denotes the total number of statements in the dataset and $|\{j : t_i \in d_j\}|$ denotes the number of statements in which the word $t_i$ appears in the dataset.

TFIDF algorithm is used to calculate the weight of each sensitive character. As shown in (3).

$$\text{tfidf}_{i,j} = \frac{n_{i,j}}{\sum_i n_{i,j}} \log \frac{|D|}{|\{j : t_i \in d_j\}|}$$

2.2. Improved TFIDF algorithm

It is difficult to depict the feature of SQL statements accurately by vectorizing SQL statements using TFIDF algorithm. Because the IDF part of the algorithm only considers the relationship between sensitive characters and the number of SQL statements containing them while ignores the distribution of sensitive characters in different statements in one category. Take Example 1 to illustrate the difference before and after the improvement of the algorithm.

Example 1: In a dataset with 1000 SQL statements, there are 500 attack statements (A) and 500 normal statements (N) containing “select”, 800 attack statements and 200 normal statements containing “and”, respectively.

According to TFIDF algorithm, the $\text{IDF}$ values of “Select” and “And” in A and N classes are $\text{IDF}_1$ and $\text{IDF}_2$ respectively, and $\text{IDF}_1 = \text{IDF}_2 = \log \frac{2000}{1000} = 0.3$. It is impossible to judge which word is easier to distinguish categories. While according to the actual observation, “And” is unevenly distributed in both classes while “Select” is evenly distributed, which indicates that “And” has better ability to distinguish categories than “Select”. The IDF part is improved as shown in formula (4) with considering the distribution of feature words in the same kind of statements. When calculating the IDF value of a sensitive character $t_i$ in the attack class statements, if the number of SQL attack statements containing $t_i$ is $m$ and the number of SQL normal statements containing $t_i$ is $k$. The total number of SQL statements containing $t_i$ is $|n| = |m| + |k|$. When $|k|$ equals zero, $\text{IDF}$ is the largest. When $|k|$ approaches to zero, the ratio of $|m|$ to $|n|$ approaches to one and then $\text{IDF}$ approaches to maximum. Therefore, the weight of the feature words can be more accurately expressed by using formula (4).

$$i_{\text{idf}} = \log \frac{|m| \cdot |D|}{|n|}$$

Where $|m|$ denotes the number of SQL statements containing $t_i$ in this class, $|D|$ denotes the total number of statements in the dataset, and $|n|$ denotes the number of statements in the dataset where the word $t_i$ appears.

Formula (4) is used to verify the example in Table 1. $\text{IDF}_1 = \log(500 \cdot 2000 / 1000) = 3$, $\text{IDF}_2 = \log(800 \cdot 2000 / 1000) = 3.2$. It can be seen that the word “And” has better classification ability and is in line with the actual situation. Therefore, ITFIDF algorithm is represented as formula (5).

$$i_{\text{tfidf}} = \frac{n_{i,j}}{\sum_i n_{i,j}} \log \frac{|m| \cdot |D|}{|n|}$$
3. Proposed detection method

The detection method of SQLIAs based on ITFIDF is shown in figure 1. It mainly includes data preprocessing and SVM model training and detecting.

![Figure 1. Detection method of SQLIAs based ITFIDF](image)

The specific process is as follows: firstly, we use word segmentation tool to segment SQL statements which are in the dataset and vectorize them by utilizing the improved TFIDF algorithm. By this way, we can gain the feature dataset, which we complete the data preprocessing. Next, the dataset is divided into training set and testing set. At the same time, the training set is used to train the SVM classifier model and the testing set is used to verify the generated model. Finally, the classification results are evaluated. The text vectorization module based on ITFIDF algorithm and SVM classification modules are introduced respectively.

3.1. Text vectorization of SQL statements

Text vectorization is performed based on ITFIDF algorithm using SQL dataset where each statement is segmented to words. At first, a large number of SQLIAs statements and normal statements are analyzed and the ITFIDF values of 32 sensitive characters summarized by the predecessors [15] are used as the partial feature points of SQL statements. The 32 sensitive characters are as follows: "!", ", #, $, %, &, ', (, ), *, +, -, ., /, :, ;, <, =, >, ?, @, [, ], ^, —, `, {, |, }, ~. At the same time, the length of the SQL statement and the term frequency of sensitive keywords are taken as the other two feature points of the SQL statement. Therefore, each SQL statement after processing contains 34 features which are the length of the SQL statement and the term frequency of sensitive keywords and the corresponding ITFIDF values of 32 sensitive characters. The sensitive keywords include: "and, restore, exists, in, substring, substr, infinite, select, information, char, xor, union, create, local group, mid, load, by, chr, version, master, inner, backup, order, alter, objects, truncate, exec, dumpfile, objects, update, Xp_cmdshell, ascii, outfile, user, group, data, have, count, insert, execute, join, load_file, drop, declare, len, length, administrators, schema, backup, or, delete".

Text vectorization algorithm for SQL statement is designed based on ITFIDF algorithm, as shown in Table 1.

**Table 1. Text vectorization algorithm for SQL statement based on ITFIDF**

| Input: segmented SQL dataset | Output: vectorized dataset(each statement expressed by $x_i$) |
|-------------------------------|----------------------------------------------------------|
| 1. Foreach statement in dataset; /*Calculate feature vectors for each data in dataset*/ |
| 2. $f_1 = $statement.Count(); /*Statistics of the length of the SQL statement and use it as the first feature value*/ |
| 3. Foreach term in statement/*Statistics of the term frequency of sensitive keywords in this SQL statement*/ |
| 4. if keywords[].contains (term); |
| 5. i++; |
| 6. End for |
| 7. $f_2 = i/f_1; /*Formula (1) is used to calculate the term frequency of sensitive keywords which is treated as the second feature value*/ |
| 8. Foreach(string arg in character[]) /*Calculate itfidf value for each sensitive character*/ |
tf = n/f1; /*Formula (1) is used to calculate the tf value of the sensitive character*/

i_idf = Math.Log(D*m/n,10); /*Formula (4) is used to calculate the idf value*/

i_tfidf = tf*idf; /*Formula (5) is used to calculate the i_tfidf value*/

End for

The i_tfidf values of 32 sensitive characters are taken as the other 32 feature values respectively;

Return \( x_i \);

Return Text Vectorized Dataset

3.2. SQLIAs detection based on SVM

As a classical statistical learning algorithm, SVM stands out with the theory of structural minimization and kernel space [16]. The two sample sets in figure 2 are linearly separable and a straight line can be found to separate the two classes. Straight line \( H \) in figure 2 separates the two types of samples correctly, and on this basis, the classification interval is guaranteed to be max. Therefore, the line \( H \) is the optimal classification line, which is expressed by equation \( w \cdot x + b = 0 \). In the figure 2, \( d \) represents the distance between the support vector and the segmented hyperplane, \( w \) is the normal of the classified hyperplane and \( b \) is the offset. \( x \) represents the sample vector.

![Figure 2. Optimal classification line \( H \) of support vector machine](image)

The optimal classification line is extended to n-dimensional space, and SVM completes the classification process by searching the optimal hyperplane. How to find the optimal hyperplane in SVM is equivalent to solving the constrained optimization problem, as shown in formula (6).

\[
\begin{aligned}
\max_{\alpha} & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\
\text{s.t.} & \sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, n
\end{aligned}
\]  

(6)

Where \( C > 0 \) is a penalty parameter, indicating the degree of emphasis on outlier points, and \( \alpha_i \) is Lagrange multiplier. Here, \( x_i \) is the vectorized SQL statement by 3.1, and \( y_i \) is sample marker. The detection of SQLIAs only needs to determine whether the SQL statements are safe or not, which is equivalent to the binary classification process of SVM. The vectorized dataset is used as data source, and the dataset is trained and predicted by SVM classifier.

4. Experiment

4.1. Experimental platform

Experimental hardware environments are as follows: Intel (R) Core (TM) i5-7300HQ CPU@2.5gHz, 8G memory. Software environments which include Windows 7 operating system and Visual Studio 2013 and sklearn machine learning platform are installed. Moreover, C# and Python are used to perform programming. The SQLIAs data used in the experiment come from the open source
libinjection project on GitHub. We remove duplicate data from the project and make them contain as many kinds of SQLIAs as possible in order to make the dataset have more injection attack characteristics. In addition, by checking the database logs of the website, the access records of normal users to the database in a certain period of time are selected as legitimate SQL statements. Finally, there are 1000 attack data and 1000 legitimate data, totaling 2000.

4.2. Experiment and analysis

4.2.1. Data preprocessing. Data preprocessing mainly includes SQL statement segmentation, text vectorization and normalization. The “happierfuntokenizing” tool is used to divided the SQL statements into words in the phase of SQL statement segmentation. Take example 2 to illustrate the process.

Example 2: Select * From data Where uname = ‘admin’ and 1=1 and ‘a’ = ‘a’.

After being segmented, it corresponds to 22 words: ‘Select’; ‘*’; ‘From’; ‘data’; ‘Where’; ‘uname’; ‘=’; ‘’ ; ‘admin’; ‘’ ; ‘and’; ‘1’; ‘=’; ‘1’; ‘and’; ‘’ ; ‘a’; ‘’ ; ‘=’; ‘’ ; ‘a’; ‘’ .

In the phase of text vectorization, we can get the dataset where each SQL statement is represented by a 34-dimensional feature vector by using the ITFIDF-based SQL statement text vectorization algorithm which shown in Table 1 to process the dataset. Then each data in the dataset is marked. The statement is marked as -1 when it is an attack sample and is marked as +1 when it is not. In addition, in order to reduce the difference of different features during the experiment, the Min-Max method is used to normalize the data. As shown in formula (7).

\[
X_{\text{norm}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{7}
\]

Where \(X\) denotes the current sample data value, \(X_{\text{min}}\) denotes the minimum of the sample data, \(X_{\text{max}}\) denotes the maximum of the sample data, and \(X_{\text{norm}}\) denotes the normalized value.

4.2.2. Evaluating indicator. Generally, we use precision, recall and F-score to measure the classification performance of machine learning model. Among them, the precision represents the proportion of predicted accuracy in the total number of predictions, and indicates the accuracy of the classifier in predicting a certain category. As shown in formula (8).

\[
Pr = \frac{TP}{TP + FP} \tag{8}
\]

Recall represents the correct proportion which is predicted in all correct totals, reflecting the completeness of classification. As shown in formula (9).

\[
R = \frac{TP}{TP + FN} \tag{9}
\]

Precision and recall are mutually restrictive. Therefore, it can be measured comprehensively by F-Score. As shown in formula (10).

\[
F = \frac{Pr \times R \times 2}{Pr + R} \tag{10}
\]

Among them, \(TP\) denotes the number of statements that are actually SQL attack statements and are predicted to be SQL attack statements. \(FP\) denotes the number of statements that are actually SQL legitimate statements but are predicted to be SQL attack statements. \(FN\) denotes the number of statements that are actually SQL attack statements but are predicted to be SQL legitimate statements.

4.2.3. Result analysis. In order to verify the performance of ITFIDF method in vectorization of SQL statements, this paper compares it with similar methods such as literature [13] and literature [15].
Literature [13] summarizes six features by manual selection and mathematical statistics which transforms SQL statements into feature vectors of fixed dimensions. Literature [15] transforms the dataset of SQL statements into 34-dimensional feature vectors by TFIDF algorithm. In addition, in order to further verify the performance of SVM in detecting SQLIAs, SVM is compared with other machine learning methods such as KNN and Decision Tree. The main idea when using KNN to classify samples is that if a sample has most of the k most similar samples in feature space (ie, the nearest neighbor in the feature space) belong to a certain category. Then the sample also belongs to this category and k uses an integer not greater than 20. In KNN algorithm, the selected neighbors are all the objects that have been correctly classified. This method only decides the category of the sample to be classified according to the category of one or more nearest samples. Decision tree is a tree structure in which each non-leaf node represents a test on a feature attribute and each branch represents the output of the feature attribute in a certain range, and each leaf node stores a category. The decision process using the decision tree is to start from the root node, test the corresponding feature attributes in the item to be classified, and select the output branch according to its value until the leaf node is reached. Finally, the category stored in the leaf node is regarded as the decision result.

In experiment, the dataset is divided into train set and test set, which are 80% and 20% respectively. Then the datasets processed by literature [13], literature [15] and ITFIDF-based SQL statement text vectorization algorithm are imported into sklearn. At first, SVM is used for model training and prediction. The experimental results are shown in Table 2. In order to further verify the excellent performance of SVM model in detecting SQLIAs, other machine learning methods such as KNN and Decision Tree are compared with SVM. The experimental results are shown in Table 3.

| Table 2. Experimental results of different data processing methods by using SVM (%) |
|-------------------------------------|-----------|-----------|-----------|
| 6-Dimensional Feature Method       | Precision | Recall    | F-score   |
| TFIDF                              | 95.11     | 94.56     | 94.83     |
| ITFIDF                             | 99.08     | 99.34     | 99.21     |

Table 2 shows that the accuracy, recall and F-score of the proposed method are 99.08%, 99.34% and 99.21% respectively when detecting SQLIAs based on SVM, which are much higher than those of literature [13]. This is because compared with the method in literature [13], the ITFIDF values of 32 sensitive characters, the length of the SQL statement and keyword frequency are taken as feature points, which make up for the shortcomings of the method of taking term frequency as text vector in literature [13]. At the same time, the three indicators of this method are more than 3% higher than those of literature [15]. This is because compared with the method in literature [15], this method takes the distribution of feature words in similar statements into consideration, so the final feature vectors are more representative. Therefore, this method has higher precision, recall and F-score than other similar methods when detecting SQLIAs based on SVM.

| Table 3. Experimental results of different machine learning methods (%) |
|-------------------------------------|-----------|-----------|-----------|
| ITFIDF+SVM                          | 99.08     | 99.34     | 99.21     |
| ITFIDF+KNN                          | 98.22     | 98.12     | 98.17     |
| ITFIDF+DT                           | 98.18     | 97.98     | 98.08     |

Table 3 shows that dataset which is processed by ITFIDF algorithm is trained and classified by using SVM, KNN and Decision Tree classifier respectively. Compared with KNN and Decision Tree, the SVM classifier has higher precision, recall and F-score.

5. Conclusion
This paper proposes a detection method of SQLIAs based on ITFIDF algorithm. Firstly, the “happierfuntokenizing” tool is used to divide the SQL statement into words. Then ITFIDF algorithm is used to vectorize the dataset. Finally, SVM machine learning model is used for training and testing.
The experimental results show that the proposed ITFIDF algorithm can further improve the detection rate of SQLIAs. The method has higher accuracy, recall rate and F-score compared with other similar methods. We will be focus on how to improve machine learning methods to get a better training model in the future.

References

[1] Ray D and Ligatti J. 2014. Defining code-injection attacks. In Proc of the 17th Int Conf on Information Security. Berlin: Springer, 425-441.

[2] Ray D, Ligatti J. 2012. Defining code-injection attacks. In Proc of the 39th Annual ACM SIGPLAN-SIGACT Symp on Principles of Programming Languages. New York: ACM, 179-190

[3] OWASP. 2017. The Ten most critical Web application security risks. OWASP Top 10

[4] Mi-Yeon K and Dong H L. 2014. Data-mining based SQL injection attack detection using internal query trees. Expert Systems with Applications. 17, 2. 5416-5430

[5] Das D, Sharma U and Bhattacharyya D K. 2017 Defeating SQL injection attack in authentication security: an experimental study. International Journal of Information Security. 18, 1 (Nov. 2017), 1-22

[6] Han Chenwang, Lin Hui and Huang Chuan. 2016. Research on the SQL injection filtering based on SQL syntax tree. Chinese Journal of Network and Information Security, 2, 11 (Nov. 2016) 1-8

[7] Shi Congcong, Zhang Tao and Yu Yong. 2012. New Approach for SQL-Injection Detection. Computer Science. 39, 6 (June. 2012) 60-64

[8] Inyong Lee, Soonki Jeong, Sangsoo Yeo and Jongsub Moon. 2012. A novel method for SQL injection attack detection based on removing SQL query attribute values. Mathematical and Computer Modeling. 58-68

[9] V. Benjam Livshits and Monica S. Lam. 2010. Finding Security Vulnerabilities in Java Applications with Static Analysis. Computer Science Department of Stanford University

[10] Nenad Jovanovic, Christopher Kruegel and Engin Kirda. 2012. Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities (Technical Report). Secure Systems Lab Vienna University of Technology

[11] E. Bertino, A. Kamra and J. P. Early. 2011. SQL injection detection using query transformation and document similarity. International Conference on DCIT

[12] Uwagbole S-O, Buchanan W-J, Lu F. 2017. Applied Machine Learning predictive analytics to SQL Injection Attack detection and prevention. Integrated Network and Service Management. Lisbon: Portugal, 1-7

[13] Wu Shaohua, Cheng Shubao and Hu Yong. 2015. Web attack detection method based on support vector machines. Computer Science, 42, 1. 362-364

[14] Kar D, Agarwal K, Sahoo A-K, et al. 2016. Detection of SQL injection attacks using hidden markov model. International Conference on Engineering and Technology. Coimbatore, India: 1-5

[15] Xu Dongdong, Xie Tongyi and Wan Zhuohao. 2018. Detection of SQL injection based on TF-IDF text vectorization. Journal of Guangxi University (Nat Sci Ed). 43, 5 (Oct. 2018) 1818-1826

[16] Paul R M, Kashif K and Qi S. 2018. SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel. Journal of Information Security and Applications. 4, 1. 199-216

Acknowledgments

This paper is supported by Funded Project of Basic and Frontier Technology Research Program of Henan Province (ID: 142300413201).