FIT5190 Introduction to IT Research Methods

Assignment 3

Overview of a Research Project

SQL injection attack detection Using Fuzzy Logic and Naïve Bayes

HU Ying (2819****)

Southeast University - Monash University Joint Graduate School

Submission date: 29/5/2017

Abstract

As web applications become more widely used, web security also faces more threats. SQL injection attack is a type of code injection attacks, which is one of the most serious threats to web security. Since it has a lot of forms and variations, being the obstacles in detection techniques, new methods with flexibility and scalability are desired to be developed. This paper proposes a novel detection method for SQL injection attacks using Fuzzy logic and the Naïve Bayes Algorithm. In this method, all features representing a SQL query are clustered to a specific malicious level by using K-means, which the fuzzy degrees are calculated based on. Then a modified Naïve Bayes classifier is deployed to do the final classification of SQL queries.

Keywords: SQL injection, SQL injection detection, web security

1. Introduction

With extensively using of web applications, more and more new attacks keep emerging endlessly with different targets. Web security deserves more and more attentions. SQL injection attacks is a kind of malicious codes where SQL queries are likely to be changed by un-sanitized user inputs, leading to undesired effects on server databases, which is also one of the most frequent web attacks (Valeur et al., 2005). Since there are a number of SQL injections, such as tautologies, illegal incorrect queries, union queries and PiggyBacked Queries, effective methods for detecting and preventing various SQL injections are urgently needed (Halfond et al. 2006). So far, many methods have been put forward using different analysis methods. Pattern matching is the most common methods used to detect and prevent SQL injection attacks. Prabakar et al. (2013) propose a method based on Aho–Corasick pattern matching algorithm, consisting of static phase for matching patterns with static pattern list and dynamic phase for manually adding new patterns into the pattern list. Another method is to compare the user-generated queries with the attribute values being removed with pre-defined queries (Lee at al., 2012). Jang and Choi (2014) develop a detection technique based on estimating the query results of user-generated queries to distinguish the malicious queries from the normal queries. But most detection methods lack flexibility and scalability. Facing explosively increasing varieties of malicious codes, these methods may lose effectiveness. To fill this gap, machine learning is more suitable for a broader range of attacks. Joshi and Geetha (2014) use Naïve Bays machine learning algorithm to classify the SQL queries. But this method only tests on three types of SQL injections, which is far less than enough.

In this research, we proposed a novel method for SQL injection detection using a combination of Fuzzy logic and Naïve Bayes. Fuzzy Logic is usually employed to handle degrees of uncertainty in attack detection. A number of features are extracted from SQL queries for detection so that the feature space is quite large. Some features are dependent to some degree, while some are mostly independent. That's the reason of using a Naïve Bayes classifier which is one of simple probabilistic classifiers based on

Bayes' theorem with an assumption with a naïve independence between features. The fuzzy degrees are used in Naïve Bayes to calculate the final posterior probabilities.

In the rest of this paper, we first state the objectives of our research, then introduce our proposed methodology for SQL injection detection. In section 4, the novelty of this research is discussed. Finally, the conclusion is presented in section 5.

2. Objectives

In this research, we propose a novel method that has the ability to effectively detecting SQL injection using Fuzzy logic and Naïve Bayes. Give that different types of SQL injection may have different features, some features are not exactly independent. But it is merely impossible to classify all SQL queries in a large features space. So Fuzzy rules can be used to define relations between some features, contributing to the Naïve Bayes classifier for a higher detection accuracy rate.

3. Methodology

In this section, our proposed method for SQL injection detection is introduced. This method is divided into four stages: extracting SQL injection features, clustering features using K-means, Applying Fuzzy logic and Naïve Bayes classifier, as explained below.

A. Extracting SQL injection features

SQL injection attacks are generally conducted through adding a series of undesired string concatenation of static strings and variables. These keywords usually to perform some operations on the database. So, we extract specific tokens or characters which present high frequency in malicious SQL queries. Every query is represented by a seven-dimensional feature vector which represent one of seven features in Table 1. These chosen features are quite typical for most types of SQL injections including tautologies, illegal incorrect queries and union queries.

Table 1 Extracted Features

Feature	**Description**
f1	Frequency of dangerous characters (e.g. --, #, /**/, ‘, “, ||, =, @@, \\\\).
f2	Frequency of dangerous tokens (e.g. delete, insert, create, union, exec, or).
f3	Frequency of punctuations (e.g. <, >, *, ;, -, (,), @, +, -, ?).
f4	Frequency of SQL tokens (e.g. select, where, table, and, delete, update, insert, join, check).
f5	Length of SQL statement.
f6	Frequency of spaces (‘ ’) within the query parameter with more possibilities of attack.
f7	The number of identical equations in SQL statements (e.g. 1=1, ‘ab’=’ab’).

B. Clustering features using K-means

To utilized the extracted features mentioned above in fuzzy logic, linguistic terms are needed to represent malicious level for all features. Therefore, K-means, which is a typical unsupervised learning method that partitions the dataset into K clusters, is used to cluster all features into three levels, low (L), medium (M) and high (H). A clustering procedure is implemented for each feature $f_i(i = 1,2,\dots,7)$. We use $X = \{x_i | i = 1,2,\dots,n\}$ to represent the dataset for each feature, and $C = \{c_i | i = 1,2,\dots,k\}$ to represent the initial centers. The general clustering process is comprised of following steps:

1) Randomly select three cluster centers $(k = 3)$ for L, M, H.
2) Calculate the distance between each point and each cluster center. The point that is nearest to the cluster center were clustered into the cluster.
3) Calculate the average value of all points as the new cluster center: $c_i = (\frac{1}{m_i}) \sum_{j=1}^{m_i} x_i$, where $m_i(i = 1,2,\dots,k)$ is the number of points in each cluster.
4) Repeat step 2-3 until the value of the object function $\mathrm{J(C)} = \sum_{i=1}^{k} \sum_{j=1}^{m_i} (\| x_j - c_i \|)^2$ is smaller than a threshold ε.

Through these steps, all features are clustered into L, M, or H.

C. Applying Fuzzy Logic

Fuzzy logic provides a formal system of numerical computation for dealing with linguistic variables whose values (terms) are characterized by fuzzy sets. A fuzzy set is a class of objects characterized by a membership, generally represented as $D = \{(x, \mu_D(x)) | x \subseteq X, \mu_D(x) \in [0,1]\}$. x is the input feature. D is usually represented as a specific linguistic term, a fuzzy subset in X describing x belonging to the universal dataset X. $\mu_D(x)$ is a membership function mapping the input space to degrees of membership. There are seven inputs corresponding to seven features mentioned above and one output for the malicious level of whole SQL queries. Three linguistic terms of malicious levels are used to describe these features, following the three clusters, (L, M, H) representing low, medium and high. We calculate the respective membership functions $\mu_D(D \in \{L, M, H\})$ for these three terms as: $f(x) = max(min\left(\frac{x-l}{m-l}, \frac{u-x}{u-m}\right), 0)$, where l, m and h are the lower, center and upper boundaries of the respective cluster of K-means clustering.

To utilized the old experience of detecting SQL injection using features, a series of fuzzy rules are necessary to be defined. A fuzzy rule is a form of a conditional statements of IF-THEN consisting of condition and action. In this fuzzy logic system, for concise illustration, we use $f_i(i = 1,2,\dots,7)$ to represent the malicious level of each features, and $mLevel$ to represent the malicious level of the SQL queries. So fuzzy rules are described as:

1) IF f_1 AND f_2 are H, THEN $mLevel$ is H.
2) IF f_1 AND f_2 are M, THEN $mLevel$ is M.
3) IF f_1 , f_2 AND f_7 are H, THEN $mLevel$ is H.

4) IF f_1 , f_2 OR f_7 are H, THEN $mLevel$ is M.
5) IF f_1 AND f_7 are H, THEN $mLevel$ is H.
6) IF f_2 AND f_7 are H, THEN $mLevel$ is H.
7) IF half in $f_i(i = 1,2,\ldots,7)$ are H, THEN $mLevel$ is H.

D. Naïve Bayes Classifier

In this step, we assume an independence between most features in most conditions except those mentioned in the fuzzy rules. Naïve Bayes classifier is a kind of probabilistic model assuming one feature is unrelated to any other features. It is based on Bayes' theorem: $\mathrm{P}(h|D) = \frac{\mathrm{P}(D|h)\mathrm{P}(h)}{\mathrm{P}(D)}$, where h is the class of the malicious level, D is the input feature vector of one SQL query: $(f_1, f_2, \ldots, f_7)$, $\mathrm{P}(D|h)$ is the prior probability, $\mathrm{P}(h)$ is the probability distribution, $\mathrm{P}(D)$ is the likelihood, and $\mathrm{P}(h|D)$ is the posterior probability. To utilized the results from the fuzzy logic system, we modify the formula as:

$$\mathrm{P}(h|D) \propto \max\{\left(\prod_{i=1}^{k} \mathrm{P}(h|f_{a_i})\right) \times P(h|R) | C_F R = \{a_i | i = 1,2,..,k\}\}, \tag{1}$$

where k features ($\{a_i | i = 1,2,..,k\}$) are considered as mutual independent, F represents the universal feature set, R is a set of interdependent features, and $C_F R$ is the complementary set of R in F. Since different types of malicious queries present different subset of features, this formula also aims to find the most possible type of SQL injection.

Maximum Posterior (MAP) is defined as the most possible class for an input data point: $h_{MAP} = \mathbf{arg}\,\boldsymbol{max}_{h\in H} P(h|D)$, where $H = \{L, M, H\}$. So h_{MAP} is the classification result of one input SQL query.

4. Novelty

SQL injection detection methods have already been studied for many years. But the increasing varieties of malicious codes may need advanced modification of these methods. This research proposes a detection method having several novelties. Above all, it has the adaption capability to detect new kinds of attacks, which is the most significant superiority distinguishing from traditional detection methods. In the procedure of this method, the extracted features and fuzzy rules are tunable for some new characteristics in SQL injection Even the malicious levels can also be adjusted for different security requirements. Furthermore, the utility of fuzziness lessens the influence brought by the quality of training dataset, contributing to a higher detection accuracy at runtime. The efficiency is also improved by assuming the independence relations between features reasonably in Naïve Bayes Classifier.

5. Conclusion and Significance

In this paper, we proposed a novel detection method for SQL injection with effectiveness and efficiency, through a combination of Fuzzy logic and Naïve Bayes. In this detection method, the SQL query are actually represented by vectors of typical features characterizing the SQL keywords, especially in malicious queries. These extracted features exert great influence on the detection accuracy. This method also can quickly adapt to new patterns that may be unable to be detected by adding new features and new rules. K-means clustering provides the basis for the Fuzzy system, especially for the membership function. By using a modified Naïve Bayes classifier with all contribution from previous steps, the SQL query are classified to the most likely malicious level.

References

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, 1, 13-15.

Jang, Y. S., & Choi, J. Y. (2014). Detecting SQL injection attacks using query result size. Computers & Security, 44, 104-118.

Joshi, A., & Geetha, V. (2014). SQL Injection detection using machine learning. International Conference on Control, Instrumentation, Communication and Computational Technologies, 1111-1115.

Lee, I., Jeong, S., Yeo, S., & Moon, J. (2012). A novel method for SQL injection attack detection based on removing SQL query attribute values. Mathematical & Computer Modelling, 55(1–2), 58-68.

Prabakar, M. A., Karthikeyan, M., & Marimuthu, K. (2013). An efficient technique for preventing SQL injection attack using pattern matching algorithm. International Conference on Emerging Trends in Computing, Communication and Nanotechnology, 503-506.

Valeur, F., Mutz, D., & Vigna, G. (2005). A learning-based approach to the detection of SQL attacks. International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, 3548, 123-140.