SQL Injection Detection with Composite Kernel in Support Vector Machine

Yi Wang[1] and Zhoujun Li[1]

[1]*State Key Laboratory of Software Development Environment, Beihang University*
{wangyi160@hotmail.com, lizj@buaa.edu.cn}

Abstract

Modern web application systems are generally consisted of database systems in order to process and store business information. These systems are highly interesting to hackers as they contain sensitive information and the diversity and amount of attacks severely undermine the effectiveness of classical signature-based detection. In this work we propose a novel approach for learning SQL statements and apply machine learning techniques, such as one class classification, in order to detect malicious behavior between the database and application. The approach incorporates the tree structure of SQL queries as well as input parameter and query value similarity as characteristic to distinguish malicious from benign queries. We develop the learning system integrated in PHP and demonstrate the usefulness of our approach on real-world application.

Keywords: *sql injection, web security, machine learning, kernel tricks*

1. Introduction

The majority of today's web-based applications does employ the multi-layer infrastructure and rely heavily on database storage for information processing. A lot of attacks against web-applications are aimed at injecting commands into database systems in order to gain unprivileged and access to sensitive records stored in these systems. The approach of protecting web application is by introducing detection models on the network layer firewall systems. These systems employ misuse detection approach and try to detect attacks by matching network traffic against known attack patterns, eg. Snort IDS and ModSecurity module.

Besides pattern based approaches, there exists a variety of research on employing anomaly based methods for detecting web-based intrusions or program analysis on source code of target web application. These approaches are either rooted at the network or application protocol layer or in need of source code. Moreover, there exists plenty of machine learning related works in the wild [1-8]. In this work we focus on the detection at the spot between application and database, detecting of anomalous SQL statements, which are malicious in the sense that they include parts of injected code or differ from the set of queries usually issued within an application.

The main contribution of our work is the use of both syntax and semantic based analysis, i.e. tree-vector-kernel based learning, which became popular within the field of natural language processing (NLP). Our approach incorporates the parse tree structure of SQL queries as well as input parameter and query value similarity characteristic to distinguish malicious from benign queries. By applying this kernel trick into the SVM(support vector machine) classifier, we can determine abnormal query accurately and efficiently.

2. SVM Kernel Tricks and Tree-vector Kernel

Support Vector Machines are linear classifiers that, through the Kernel trick, operate in reproducing Kernel Hilbert spaces and are thus able to perform non-linear classification and regression in their input space. As in this work we are dealing with the detection of malicious database queries, we choose a tree-vector based approach to represent SQL queries for making it suitable for machine learning.

2.1. SVM kernel Tricks

The Kernel trick is powerful because it provides a bridge from linearity to non-linearity to any algorithm that solely depends on the dot product between two vectors. It comes from the fact that, if we first map our input data into a higher-dimensional space, a linear algorithm operating in this space will behave non-linearly in the original input space. It is really useful because that mapping does not need to be ever computed. If the algorithm can be expressed only in terms of a inner product between two vectors, all we need is replace this inner product with the inner product from some other suitable space. The kernel function denotes an inner product in feature space and is usually denoted as:

$$K(x,y) = <\varphi(x), \varphi(y)>$$

Using the Kernel function, the algorithm can then be carried into a higher-dimension space without explicitly mapping the input points into this space. This is highly desirable, as sometimes the higher-dimensional feature space could even be infinite-dimensional and thus infeasible to compute.

2.2. Tree-vector Kernel for Structured Data

Exploit the syntactic parse tree information is considered most useful tools for structured data processing. For example, the learning models for automatic Word Sense Disambiguation or Coreference Resolution would benefit from syntactic tree features but their design and selection is not an easy task.

Convolution kernels are an alternative to the explicit feature design. They measure similarity between two syntactic trees in terms of their sub-structures.

*Definition 1 (**Vector sets**):* Given two objects, O_1 and O_2, described by two sets of feature vectors, $\{\vec{v}_1, \vec{v}_2, ..., \vec{v}_{n_v}\}$ and $\{\vec{u}_1, \vec{u}_2, ..., \vec{u}_{n_u}\}$, several kernels can be defined:

$$K(o_1, o_2) = K(\{\vec{v}_1, \vec{v}_2, ..., \vec{v}_{n_v}\}, \{\vec{u}_1, \vec{u}_2, ..., \vec{u}_{n_u}\})$$

*Definition 2 (**Tree forests**):* O_1 and O_2, described by two sets of trees, $\{T_1, T_2, ..., T_n\}$ and $\{T_1', T_2', ..., T_n'\}$, two type of tree kernels, SubSet Tree kernel (SST) and Subtree kernel (ST) can be defined on:

$$K(o_1, o_2) = K(\{T_1, T_2, ..., T_n\}, \{T_1', T_2', ..., T_n'\})$$

*Definition 3 (**Combinations of Trees and vectors**)*:

(1) *sequential summation*, the kernels between corresponding pairs of trees and/or vectors in the input sequence are summed together. The τ parameter rules the contributions of tree kernels K_t with respect to the feature vector kernel k_b. Each of the two types of kernels can or cannot be normalized according to a command line parameter. More formally:

$$K_s(o_1,o_2)=\tau\times\sum_{i=1,\ldots,min(n,n')}k_t(T_i,T_i')+\sum_{i=1,\ldots,min(n_v,n_u)}k_b(\vec{v}_i,\vec{u}_i)\ (1)$$

K_t can be either the SST or the ST kernel whereas k_b can be one of the traditional kernels on feature vectors, e.g. gaussian or polynomial kernel.

(2) *all vs all summation*, each tree and vector of the first object are evaluated against each tree and vector of the second object:

$$K_a(o_1,o_2)=\tau\times\sum_{\substack{i=1,\ldots,n\\ j=i,\ldots,n'}}k_t(T_i,T_j')+\sum_{\substack{i=1,\ldots,n\\ j=i,\ldots,n'}}k_b(\vec{v}_i,\vec{u}_j)\ (2)$$

3. Kernel Function for SQL Query

In this section we will introduce the related kernel functions specific to SQL query respectively.

3.1. Tree Kernel Function

Malicious SQL query often contain carefully prepared user string input in order to change the normal database behavior by different command or condition checking. This may lead to syntax change from benign SQL query in most cases. For example, by appending "' OR 1=1 --" into "age" value will cause the original query "Select * from user where name='john' and id='1001' into different tree structure "Select * from user where name='john' and id='1001' OR 1=1 =='.

The main idea of tree kernels is to compute the number of the common sub-structures between two trees $T1$ and $T2$ without explicitly considering the whole fragment space. For this purpose, we need to define the tree kernel function in order to compute the similarity of two trees.

$$K_t(T_1,T_2)=\sum_{n_1\in N_{T_1}}\sum_{n_2\in N_{T_2}}\Delta_k(n_1,n_2)\ (3)$$

where N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes, respectively. By adopting the concept of tree kernel from (ECAL 2006), we can define

$$\Delta_k(n_1,n_2)=\begin{cases}0 & if\ \ prod(n_1)\neq prod(n_1)\\ \lambda & if\ height(n_1)=height(n_2)=1\ (4)\\ \lambda\prod_{j=1}^{|n_1|}(\sigma+\Delta(c_{n_1}^j,c_{n_2}^j)\ otherwise & \end{cases}$$

where λ is the decay factor and $\sigma\in[0,1]$ is the counting factor, $|n|$ is the number of the children of node, for the last condition, $|n_1|=|n_2|$.

3.2. Vector Kernel Function

In web application programs, user input data will be passed as HTTP parameters via GET or POST methods. They will influence the data flow of program and may ultimately determine what query value in SQL query will be sent. Thus, it is naturally to assume that there exists strong relation between HTTP parameter and query value and this will also be an important factor in deciding the difference between malicious from benign queries. For the purpose of that, we extract user input values $\{u_1, u_2, ..., u_n\}$ and query values $\{v_1, v_2, ..., v_m\}$.

The best-known character-based string similarity metric is Levenshtein distance(LD), defined as the minimum number of insertions, deletions or substitutions necessary to transform one string into another. For our case, we adopt this simple algorithm due to the unavailability of term sets for user inputs. In order to make measurement for similar strings bigger than different strings, we define:

$$\Delta_b(u,v) = \frac{1}{LD(u,v)} \quad (5)$$

By making input-query value pair $P = \{\Delta_b(u_1,v_1), \Delta_b(u_2,v_1), ..., \Delta_b(u_m,v_n)\}$, we can define vector kernel to calculate the similarity:

$$K_b(P,P') = \sum_{i=1}^{|P|} Gaussian(P_i, P_i') \quad (6)$$

Where |P|=|P'| Gaussian(…) is the Gaussian radial basis function:

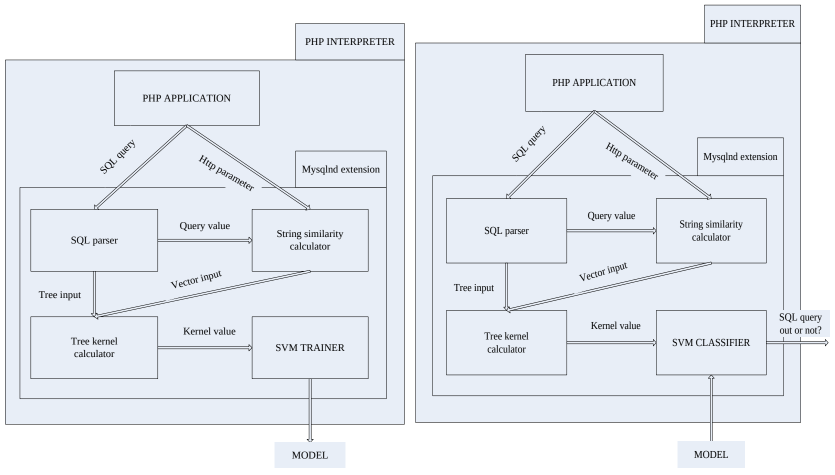
$$Gaussian(x_i, x_j) = exp(-\gamma \| x_i - x_j \|^2), \gamma > 0 \quad (7)$$

4. System Design and Evaluation

We present our prototype system SQLLEARN as a mysqlnd extension integrated in PHP interpreter. It functions as a SQL proxy with the ability of query learning and anomaly detection between PHP application and Mysql database. Several vulnerable PHP content management system applications are tested within this framework, the results show that our system can provide accurate and complete protection against SQL injection attacks.

4.1. System Design

We first utilize the internal SQL parser in open source Apache Derby database to generate the parse tree of SQL query. Next, we build extension on mysqlnd(a php module which communicate with mysql database) to capture HTTP request parameter. By combining the two tools, we can gather the information needed for calculating kernel function both in training phase and testing phase. The tree structure data together with vector pairs will be write in input format for SVM-LIGHT-TK (tree kernel extension of a popular SVM solver) so as to reuse its optimized implementation to compute the kernel value. Finally, the precomputed kernel value will be used as input for LIBSVM (another SVM solver) and reuse its one-class classifier to learn and the model for later testing. The system architecture is shown in Figure 1.



(a) Training phase (b) Testing phase

Figure 1. SQLLEARN System Architecture

4.2. Evaluation

We choose ST kernel K_s for the test case, as SQL queryand HTTP request parameter's sequence in real life nearly never change and SST kernel K_a will only bring more computation burden. By using different values of λ, σ, γ and τ, we come to the optimized combination as 1, 0.6, 1 and 0.5 respectively in order to achieve the best detection rate (TPR) and fraction of false positive rate (FPR). Moreover, we compare the tree-vector kernel with tree and vector kernel alone to show that the combination kernel surpass any singleton kernel and obtain the best result, as show in Table 1. This reflect the fact that both syntax and application context play important role in the detection of malicious SQL injection.

Table 1. Detection and False Positive Rate of the Different Models based on SVM

Kernel Type	TPR	FPR	Time(ms)
Tree	0.890	0.002	2.023
Vector	0.560	0.234	0.450
Tree-vector	0.982	0.000	3.862

5. Conclusion

We presented an approach using tree-vector-kernels in SVM for SQL statements to prevent SQL injection in web applications. The results confirm the benefit of incorporation of syntax information of query and semantic context from application in analyzing SQL queries. Compared to previous approaches, the combination gains more accuracy than using syntax or context analysis alone as it brings more information into classification. Although tree kernel calculation is a time consuming task, we can separate the classification into sub-task according to request URL, which will make efficient online testing possible. With the system integrated into PHP interpreter, no modification is needed for web application to gain extra protection against SQL injection attacks. As far as we know, this is the first practical, light-weight learning solution to fight against such attacks. In future works we plan to improve the string similarity calculator and tree kernel functions in both accuracy and efficiency in order to make it suitable in commercial product.

References

[1] A. Moschitti, “Making tree kernels practical for natural language learning”, In Proceedings of the Eleventh International Conference on European Association for Computational Linguistics, Trento, Italy, **(2006)**.

[2] S.-Y. Lee, W. L. Low, P. Y. Wong, “Learning fingerprints for a database intrusion detection system”, In: Gollmann, D., Karjoth, G., Waidner, M. (eds.) ESORICS 2002. LNCS, vol. 2502, pp. 264–280. Springer, Heidelberg **(2002)**.

[3] G. Buehrer, B. W. Weide, and P. A. G. Sivilotti, “Using parse tree validation to prevent sql injection attacks”, In: Proc. of SEM, pp. 106–113. ACM, New York **(2005)**.

[4] R. Gerstenberger, “Anomaliebasierte Angriffserkennung im FTP-Protokoll”, Master’s thesis, University of Potsdam, Germany **(2008)**.

[5] P. D¨ussel, C. Gehl, P. Laskov and K. Rieck, “Incorporation of application layer protocol syntax into anomaly detection”, In: Sekar, R., Pujari, A.K. (eds.) ICISS 2008. LNCS, vol. 5352, pp. 188–202. Springer, Heidelberg **(2008)**.

[6] C. Bockermann, M. Apel and M. Meier, "Learning sql. for database intrusion detection using context-sensitive modelling", in Detection of Intrusions and Malware, and Vulnerability Assessment, Volume 5587/2009. Springer Berlin / Heidelberg, pp. 196-205 **(2009)**.

[7] R. Dewhurst, “Damn Vulnerable Web Application (DVWA)”, http://www.dvwa.co.uk/, **(2012)**.

[8] Bernardo Damele A. G., Miroslav Stampar, “Sqlmap: automatic SQL injection and database takeover tool”, http://sqlmap.sourceforge.net/ **(2012)**.

Authors

Yi Wang

Born in 1980. PhD candidate. His research interests include web security and program analysis

Zhoujun Li

Born in 1963. PhD. Professor. PhD supervisor and senior member of China Computer Federation. His main research interests include formal verification of security protocol, program analysis and data mining.