Expert Systems with Applications 41 (2014) 5416–5430

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Data-mining based SQL injection attack detection using internal query trees

Mi-Yeon Kim, Dong Hoon Lee *

Center for Information Security Technologies (CIST), Korea University, 145 Anam, Seongbuk, Seoul, Republic of Korea

ARTICLE INFO

Keywords:
Intrusion detection
SQL injection attack
Database
Data mining
SVM

ABSTRACT

Detecting SQL injection attacks (SQLIAs) is becoming increasingly important in database-driven web sites. Until now, most of the studies on SQLIA detection have focused on the structured query language (SQL) structure at the application level. Unfortunately, this approach inevitably fails to detect those attacks that use already stored procedure and data within the database system. In this paper, we propose a framework to detect SQLIAs at database level by using SVM classification and various kernel functions. The key issue of SQLIA detection framework is how to represent the internal query tree collected from database log suitable for SVM classification algorithm in order to acquire good performance in detecting SQLIAs. To solve the issue, we first propose a novel method to convert the query tree into an n-dimensional feature vector by using a multi-dimensional sequence as an intermediate representation. The reason that it is difficult to directly convert the query tree into an n-dimensional feature vector is the complexity and variability of the query tree structure. Second, we propose a method to extract the syntactic features, as well as the semantic features when generating feature vector. Third, we propose a method to transform string feature values into numeric feature values, combining multiple statistical models. The combined model maps one string value to one numeric value by containing the multiple characteristic of each string value. In order to demonstrate the feasibility of our proposals in practical environments, we implement the SQLIA detection system based on PostgreSQL, a popular open source database system, and we perform experiments. The experimental results using the internal query trees of PostgreSQL validate that our proposal is effective in detecting SQLIAs, with at least 99.6% of the probability that the probability for malicious queries to be correctly predicted as SQLIA is greater than the probability for normal queries to be incorrectly predicted as SQLIA. Finally, we perform additional experiments to compare our proposal with syntax-focused feature extraction and single statistical model based on feature transformation. The experimental results show that our proposal significantly increases the probability of correctly detecting SQLIAs for various SQL statements, when compared to the previous methods.

1. Introduction

With the development of information technology, a massive amount of sensitive personal information and proprietary information has accumulated in databases, which are considered to be the most valuable asset of organizations. However, the more the economic value of data increases, the more the attempt to extort the data increases. As database security concerns increase, Gartner recognizes the emergence of database audit and protection (DAP) tool (Wheatman, 2012). In addition, Gartner considers the detection and prevention of database intrusion attack as one increasingly important use case of DAP.

Database intrusion attacks could be broadly categorized into two types, depending on the access point. In the first type, malicious users with privileged user accounts or compromised user accounts directly access the database, and abuse the structured query language (SQL), to harvest the data. In the second type, malicious users indirectly access the database using the vulnerability of database-driven web applications. That is, malicious users attack the database by altering the original SQL statements within the applications, through the user input values. This type of database intrusion is well known as an SQL injection attack (SQLIA).

For the first type of database intrusion, the detection of abnormal behaviors of users has been much studied through the analysis

* Corresponding author. Tel.: +82 2 3290 4892; fax: +82 2 928 9109.
E-mail address: donghlee@korea.ac.kr (D.H. Lee).

http://dx.doi.org/10.1016/j.eswa.2014.02.041

of database log (Bertino, Terzi, Kamra, & Vakali, 2005; Mathew, Petropoulos, Ngo, & Upadhyaya, 2010; Shebaro, Sallam, Kamra, & Bertino, 2013). On the other hand, for the second type of database intrusion, namely SQLIA, the detection of malicious user inputs has been studied mainly through the analysis of queries generated within the web application (Shar & Tan, 2013). The approach to detect SQLIA at the application level turns out to be unable to detect some types of attacks (which will be discussed later in more detail). The theme of the paper is to design an efficient and accurate method to detect such SQLIAs at the database level, using a query tree (which is an internal representation of an SQL statement) written in database logs. The basis of our proposal is to model SQLIA detection as a data-mining based binary classification problem, in order to separate malicious query trees from normal query trees. The data-mining based method to detect SQLIA is beneficial in detecting unknown attacks with high accuracy against the rapid emergence of various forms of attack (Choi, Choi, Ko, & Kim, 2012; Pinzón et al., 2011; Santos, Brezo, Ugarte-Pedrero, & Bringas, 2011; Wu & Yen, 2009).

We utilize the Support Vector Machine (SVM) as a binary classification. The SVM is known to provide high accuracy in the process of a binary classification, and to deal with high-dimensional data (Boser, Guyon, & Vapnik, 1992; Burges, 1998). During the binary classification processing, we use the non-linear vector kernel function as a vector similarity measurement. The non-linear vector kernel function helps the linear binary classifier be extended to a non-linear binary classifier (Hofmann, Schölkopf, & Smola, 2008). In most cases, the non-linear classifier has better accuracy than the linear classifier (Ben-Hur & Weston, 2010). Through the experiments, we select the non-linear vector kernel function, and good values for the kernel parameters that are suited for SQLIA detection.

We report the evaluation results of the SVM model with the chosen kernel function and kernel parameters. The experimental result of our proposal shows that the area under receiver operating characteristics curve (AUC) is 0.999 for SELECT and INSERT statements, and the AUC is 0.996 for stored procedures. This means that our SQLIA detection method yields at least 99.6% of the probability that the probability for malicious queries to be correctly predicted as SQLIA is greater than the probability for normal queries to be incorrectly predicted as SQLIA.

The rest of this paper is structured as follows. In Section 2, we describe related works. In Section 3, we propose the framework to detect SQLIA at the database level and describe our contributions. In Section 4, we describe a method to convert the query tree into a feature vector representation, which is our major contribution. We report experimental results on our proposal in Section 5. We conclude in Section 6 with conclusions and future work.

2. Related works

SQLIA is a notorious attack type on the web application and database. The Open Web Application Security Project (OWASP) ranks the SQLIA as the most critical web application security risk in 2013 (Williams & Wichers, 2013). Under these circumstances, researchers have proposed various ways to defeat SQLIAs: manual and automatic defective coding techniques, static and dynamic vulnerability analysis, and runtime SQLIA detection and prevention.

SQLIA can be solved by defensive coding techniques of application developers, and code modification using the vulnerability analysis tool. However, defensive coding practices are time-consuming and labor-intensive. The vulnerability analysis tool has difficulty in analyzing precisely some of the complex source codes. Although incurring a performance penalty, the runtime SQLIA checker performs fairly well against various attack types, due to judgments based on the execution required SQL statements. Hence, our research focuses on the runtime SQLIA detection and prevention, and we describe the previous works related to this field in this section.

SQLIA can be classified into two forms, depending on the time when the malicious user inputs are entered, and the attacks occur. The primary form of SQLIA is that attackers insert malicious user inputs, and the attacks immediately occur, as the malicious inputs are concatenated with the SQL statement. The other form of SQLIA is that attackers seed malicious inputs into the database, which are used at a later time, to indirectly trigger SQLIA. The former form is called a first order SQLIA, and the latter form is called a second order SQLIA.

Of these two types of attacks, a mechanism to detect SQLIA at the application level cannot afford to defend against the second order SQLIA attack, because the malicious inputs supplied by the attacker are concatenated with the SQL statement at the database level. Similarly, a mechanism to detect the SQLIA at the application level is difficult to defend against the SQLIA based on stored procedures (Wei, Muthuprasanna, & Kothari, 2006). The stored procedure is an operation set, typically written in SQLs. Because the stored procedure is stored and executed within the database system, application layer does not notice that the syntactic structure of the SQL statements in the stored procedure by user inputs is changed.

Table 1 summarizes the availability of the countermeasures that are provided by the previous runtime SQLIA detection techniques against various SQLIA types at the application level. It lists with reference to the previous works (Halfond, Viegas, & Orso, 2006; Kindy & Pathan, 2011).

As shown in Table 1, the previous runtime SQLIA detection techniques at the application level fail to prevent the SQLIA using a stored procedure. Although omitted from Table 1, none of the previous detection techniques is capable of thwarting second order SQLIA, since they only focus on the SQL generated at the application level.

To overcome the inability of the approach to detect SQLIA at the application level, a few research was conducted to detect SQLIA at the database level. Since the Gartner research has published database activity monitoring (DAM), an earlier version of DAP, as one of the top five strategies to prevent data loss and information leaks, the importance of research to defend the SQLIA attack at the database level has been emphasized. Nevertheless, research in this area is still inactive (Kamra & Bertino, 2009).

The approaches to detect SQLIA at the database level are divided into two, depending on the viewpoint of the database logs. One approach views the database logs as the sequence of data access, and analyzes the data dependency among data items. Hu and Panda (2004) proposed the methods to generate the rules of data dependency, by mining sequential data access patterns. The data dependency rules consist of a series of data items to be read, and a series of data items to be written, before writing a specific data item. If a database transaction violates the data dependency rules, the transaction is identified as malicious. They categorized the SQL statements into the read operation and the write operation, and regarded the transaction as a logical unit, which includes one or more read/write operations for data items. Hu, Campan, Walden, Vorobyeva, and Shelton (2010) proposed a multi-level and multi-dimensional data dependency model, by extending their previous data dependency model. This model considers the multi-granularity level, such as attribute, relation, database, or even distributed site, while the previous model focuses only on the attribute dependencies. Also, this model considers correlations between the read operation and the write operation.