Cristian Pinzon
Universidad Tecnológica de Panamá
35 PUBLICATIONS **206** CITATIONS
SEE PROFILE

Juan F. De Paz
Universidad de Salamanca
285 PUBLICATIONS **1,449** CITATIONS
SEE PROFILE

Javier Bajo
Universidad Politécnica de Madrid
299 PUBLICATIONS **2,435** CITATIONS
SEE PROFILE

Juan Manuel Corchado Rodríguez
Universidad de Salamanca
832 PUBLICATIONS **12,471** CITATIONS
SEE PROFILE

An Adaptive Mechanism to Protect Databases against SQL Injection

Cristian I. Pinzón, Juan F. De Paz, Javier Bajo, Juan M. Corchado
Universidad de Salamanca, Plaza de la Merced s/n, 37008, Salamanca, Spain
{cristian_ivanp, fcofds, jbajope, corchado}@usal.es

Abstract. The purpose of this article is to present an adaptive and intelligent mechanism that can handle SQL injection attacks. This proposal focuses on integrating a case-based reasoning (CBR) mechanism with a neural network. The proposed solution thus adapts to changes in attack patterns and provides the ability to detect attacks independently of their evolution. A prototype of the architecture was developed and the results obtained are presented in this study.

Keywords: SQL Injection, database security, case-based reasoning, neural network

1 Introduction

SQL injections are one of the security problems for web solutions that involve unauthorized access to databases [1]. This attack takes place at the database layer when a user request that has been sent through an HTTP request is executed without prior validation.

Various approaches have attempted to deal with the problem of SQL injections [1] [2] [3] [4] [5]. However, the biggest inconvenience of these solutions is their inability to adapt to the rapid changes in attack patterns, which renders them a bit inefficient in the long term. More complex SQL attacks are characterized by the various techniques used for remaining undetected by existing security solutions.

This article presents the SQLCBR classifier. It is a new solution that incorporates a detection strategy that compares attack patterns (signature detection) and a detection pattern that studies the behavior in the technique of the attack (anomaly detection). The former strategy applies an initial filter to detect simple attacks, while the latter focuses on complex attacks that remain unsolved after the first filter. This strategy is based on a CBR reasoning mechanism combined with a Perceptron Multilayer neural network. The CBR system is the key component of the SQLCBR classifier mechanism. The CBR systems are based on the notion that similar problems have similar solutions [6][7]. By combining the CBR mechanism with the neural network, the system we propose is able to learn quickly and adapt to changes in the SQL attack patterns, thus facilitating the task of determining when a user request actually involves a type of SQL injection attack.

The rest of the paper is structured as follows: section 2 presents the problem that has prompted most of this research. Section 3 focuses on the design of the proposed architecture, and section 4 provides a detailed explanation of the classification model. Finally, section 5 presents the results and conclusions obtained by the research.

2 SQL Injection Attack

A SQL injection attack takes place when a hacker changes the semantic or syntactic logic of a SQL text string by inserting SQL keywords or special symbols within the original SQL command that will be executed at the database layer of an application [1]. A SQL injection attack can cause serious damage to an organization, including financial loss, breach of trust with clients, among others.

There have been many proposed solutions for SQL injection attacks, including some Artificial intelligence techniques. One of the approaches is WAVES (Web Application Vulnerability and Error Scaner) [5]. This solution is based on a black-box technique. WAVES is a web crawler that identifies vulnerable points, and then builds attacks that target those points based on a list of patterns and attack techniques. WAVES monitors the response from the application and uses a machine learning technique to improve the attack methodology. WAVES cannot check all the vulnerable points like the traditional penetration testing. The strategy used by the intrusion detection systems has also even been implemented to deal with some SQL injection attacks. Valeur [4] presents an IDS approach that uses a machine learning technique based on a dataset of legal transactions. These are used during the training phase prior to monitoring and classifying malicious accesses. Generally, IDS systems depend on the quality of the training set; a poor training set would result in a large number of false positives and negatives. Skaruz [2] proposes the use of a recurrent neural network (RNN). The detection problem becomes a time serial prediction problem. The main problem with this approach is the large number of false positives and false negatives.

Other strategies based on string analysis techniques and the generation of dynamic models have been proposed as solutions to SQL injection attacks. Halfond and Orso [1] propose AMNESIA (Analysis and Monitoring for Neutralizing SQL Injection Attacks). Kosuga et al. proposes SANIA (Syntactic and Semantic Analysis for Automated Testing against SQL Injection) [3]. With only slight variations of accuracy in the models, these strategies have as drawback their meaningful rate of false positives and negatives

3 Classifier Mechanism Design

The mechanism was strategically designed for classifying user requests into two categories: legal or malicious. The legal requests are forwarded to the database for their execution, while the malicious requests are rejected with an alert activated and sent to security personnel.

The SQLCBR Classifier Mechanism is made up of 3 principle components, as shown in Figure 1.

- The Filter-Traffic component: In charge of capturing traffic directed towards the database Server. JPCAP is the API used to identify and capture any traffic that contains HTTP requests and later facilitating the extraction of the SQL string from the rest of the traffic generated by the request.
- The SQL-Analyzer component: Syntactically analyzes the SQL string passed through the first component. The Cup and JFlex tools are used to extract and format the information required for continuing on to the final component.
- The SQLCBR-Classifier component is the core of the mechanism. The classification mechanism is divided into three subcomponents. The first is a subcomponent that is designed to identify simple attack patterns based on a pattern matching. The application of this initial filter will reduce the number of cases passed on to the CBR engine, thus improving the operation of the solution. The second subcomponent is based on a case based reasoning engine using a neural network, specifically the Multicap Perceptron, for carrying out the classification. Finally, the third subcomponent corresponds to a user interface for interacting with security personnel.

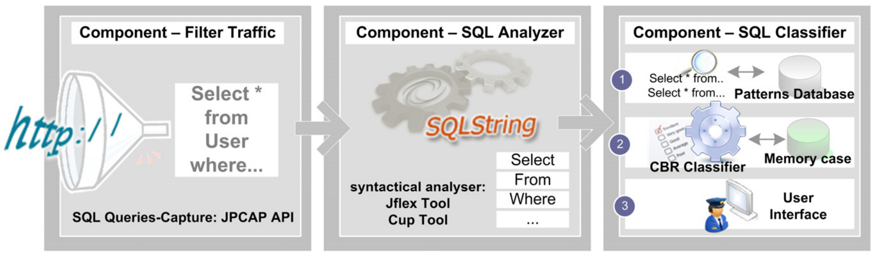

Component – Filter Traffic
http://
Select * from User where...
SQL Queries-Capture: JPCAP API
Component – SQL Analyzer
SQLString
Select
From
Where
...
syntactical analyser: Jflex Tool Cup Tool
Component – SQL Classifier
1
Select * from..
Select * from...
Patterns Database
2
CBR Classifier
Memory case
3
User Interface

Fig. 1. SQLCBR Classifier Components

4 Mechanism for the Classification of SQL Queries

CBR can be defined as a reasoning model that incorporates problem solving, understanding, and learning, and integrates all of them with memory processes. These tasks are performed using typical situations, called cases, already experienced by a system [6]. Using a CBR focus to solve problems implies the execution of a CBR cycle which is based on 4 known phases: retrieval phase, reuse phase, revise phase, retain phase. A case is defined as a previous experience and is composed of three elements: a description of the problem; a solution; and the final state.

The elements of the SQL query classification are described as follows:

(a) Problem Description that describes the initial information available for generating a plan. Table 1 lists a problem description that consists of a case identification, user session and SQL query elements. (b) Solution that describes the action carried out in order to solve the problem description. Table 1, lists the case identification and the applied solution. (c) Final State that describes the state achieved after that the solution has been applied.

Table 1. Structure of the problem definition and solution for a case of SQL query classification

Problem description fields			
IdCase	Integer	Numer_And	Integer
Sesion	Session	Numer_Or	Integer
User	String	Number_literals	Integer
IP_Adress	String	Number_LOL	Integer
Query_SQL	Query_SQL	Length_SQL_String	Integer
Affected_table	Integer	Cost_Time_CPU	Float
Affected_field	Integer	Start_Time_Execution	Time
Command_type	Integer	End_Time_Execution	Time
Word_GroupBy	Boolean	Query_Category	Integer
Word_Having	Boolean	Idcase	Integer
Word_OrderBy	Boolean	Classification_Query	Integer

The first step within the classification process consists of submitting the SQL string to a pattern matching in order to identify simple attack patterns. If an association is found between an attack pattern and the SQL request string, the user's request is immediately rejected and an alert is initiated over the security personnel user interface. After this initial filter has been applied to detect simple attack patterns, the number of SQL strings that continue to the SQLCBR classifier is reduced. However, if no anomaly is found among the patterns within the SQL string, the results of the syntactic analysis that was applied on the SQL string are passed on to the SQLCBR Classifier. These results will become the case description of the new problem as shown in Table 1. An example will be presented below.

Let us assume a query with the following syntax for bypassing the authentication mechanism: `Select field1, field2, field3 from table1 where field1='' or 11 = 11 -- 'and field2=''`. The analysis of the SQL string would generate the result presented in Table 2, with the following fields: Affected_table$^{(c1)}$, Affected_field$^{(c2)}$, Command_type$^{(c3)}$, Word_GroupBy$^{(c4)}$, Word_Having$^{(c5)}$, Word_OrderBy$^{(c6)}$, Numer_And$^{(c7)}$, Numer_Or$^{(c8)}$, Number_literals$^{(c9)}$, Length_SQL_String$^{(c10)}$, Number_LOL$^{(c11)}$, Cost_Time_CPU$^{(c12)}$, Query_Category$^{(13)}$. The fields Command_type and Query_Category have been encoding with the following nomenclature Command_Type: 0=select, 1=insert, 2=update, 3=delete; Query_Category: -1=suspicious, 0=illegal, 1=legal, 2= undefined.

Table 2. SQL String transformed through the string analysis

c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13
1	3	0	0	0	0	1	1	2	88	1	2,91	2

The first phase of the CBR cycle consists of recovering past experience from the memory of cases, specifically those with a problem description similar to the current SQL query. In order to do this, a cosine similarity-based algorithm is applied, allowing the recovery of those cases which are at least 90% similar to the current SQL query. The recovered cases are used to train the neural networks implemented in the recovery phase; the neural network with a sigmoidal function is trained with the recovered cases that were an attack or not. A preliminary analysis of correlations is required to determine the number of neurons of the input layer of the neuronal networks. Additionally, it is necessary to normalize the data (i.e., all data must be values in the interval [0.1]).

With the trained neural network we mean to detect complex SQL injection attacks. In the reuse phase the network is trained by a back-propagation algorithm for the set of available training patterns (this particular neural network is named Multilayer Perceptron), using a sigmoidal activation function (which will take values in [0.1], where 0 = malicious and 1 = legal). The number of neurons in the output layer for the Multilayer Perceptrons is 1. It is responsible for deciding whether or not there is an attack. The sigmoidal activation function is given by.

$$f(x) = \frac{1}{1 + e^{-ax}} \tag{1}$$

The review stage is performed by an expert whose final opinion will determine whether the case is stored in the memory of cases and whether the list of well-known patterns has to be updated in the retain phase.

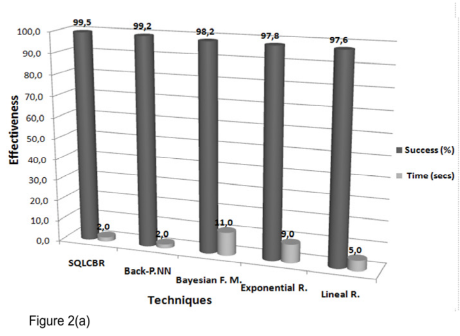
Finally, in the event that the SQLCBR Classifier is unable provide an exact decision, the user's request is rejected. This decision was made after taking into consideration the risk posed by a suspicious query that, if it turned out to be an attack, could be executed within the database.

5 Results and Conclusions

This article has presented a novel solution based on a classification mechanism capable of learning and adapting to changes in the attack patterns of the SQL injections. The proposed solution implements a case-based reasoning engine in conjunction with a neural network. Moreover, an initial filter was implemented based on a pattern matching. This will quickly resolve the simple SQL string requests and allow an improved response time for the solution.

A case study was proposed in order to validate the effectiveness of the SQLCBR classifier prototype. The tests were conducted with a simple web application with database access, MySQL 5.0. The entries were automated by using the SQLMap 0.6.3 tool, with which an initial case base was established for training the SQLCBR-Classifier. In addition the malicious queries to be analyzed by our solution were executed by this tool. Figure 2(a) shows a comparison between the SQLCBR Classifier and other techniques. The empirical results show that the best methods are those that involve the use of a neural network. This method is more accurate than statistical methods for detecting attacks to databases because the behaviour of the hacker is not linear, but dynamic and chaotic. The effectiveness of our solution was demonstrated by the results obtained.

Other data important to note here is the number of training patterns. Figure 2(b) shows the percentage of prediction with regards to the number of patterns in the training phase. It is clear that a large number the pattern of training improves the percentage of prediction. As we are working with CBR systems, which depend on a larger amount of data stored in the memory of cases for each user, the percentage of success in the prediction increases, as shown in Figure 2(b). CBR systems need to draw from initial information (past experiences) in order to generalize efficient results.



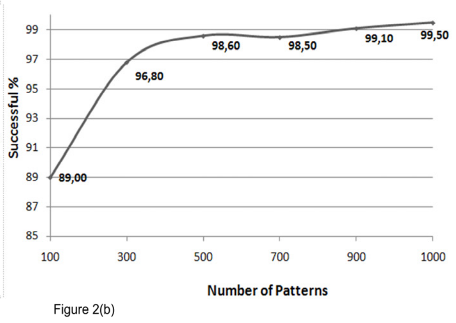


Fig. 2(a). Comparison of the SQLCBR-Classifier with other classification techniques
Fig. 2(b). Sucessful (%) vs Number of patterns

References

1. Halfond, W. and Orso, A.: AMNESIA: Analysis and Monitoring for Neutralizing SQL-injection Attacks. In: 20th IEEE/ACM international Conference on Automated software engineering, pp. 174--183. USA, New York, (2005)
2. Skaruz, J. and Seredynski, F.: Recurrent neural networks towards detection of SQL attacks. In: 21th International Parallel and Distributed Processing Symposium, IEEE International, pp.1--8 (2007)
3. Kosuga Y., Kono K., Hanaoka M., Hishiyama M. and Takahama Y.: Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection. In: 23rd Annual Computer Security Applications Conference, IEEE Computer Society, pp.107--117 (2007)
4. Valeur, F., Mutz, D. and Vigna, G.: A Learning-Based Approach to the Detection of SQL Attacks. In: Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment, pp. 123-140. Vienna, Austria (2005)
5. Huang Y., Huang S., Lin T. and Tsai C.: Web application security assessment by fault injection and behavior monitoring. In: 12th international conference on World Wide Web, ACM, pp. 148--159. New York, USA (2003)
6. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches, AI Commun, vol. 7, pp. 39--59 (1994)
7. Bajo, J., De Paz, Juan F., Corchado, J. M.: Distributed Prediction of Carbon Dioxide Exchange Using CBR-BDI Agents, INFOCOMP, Special Edition, pp.16-25 (2007)