Classification Agent-Based Techniques for Detecting Intrusions in Databases

Cristian Pinzón[1], Yanira De Paz[2], and Rosa Cano[3]

[1] Universidad Tecnológica de Panamá, Av. Manuel Espinosa Batista, Panama
[2] Universidad Europea de Madrid, Tajo s/n 28670, Villaviciosa de Odón, Spain
[3] Instituto Tecnológico de Colima, Av. Tecnológico s/n, 28976, Mexico
cristian.pinzon@utp.ac.pa, yanirarosario.depaz@uem.es, rdegca@gmail.com

Abstract. This paper presents an agent specially designed for the prevention and detection of SQL injection at the database layer of an application. The agent incorporates a Case-based reasoning mechanism whose main characteristic involves a mixture of neural networks that carry out the task of filtering attacks. The agent had been tested and the results obtained are presented in this study.

Keywords: SQL injection, multiagent systems, case-based reasoning, neural networks.

1 Introduction

Database security is a fundamental aspect of all current information systems. There are many ways of exploiting the security vulnerability in a relational database. SQL injection is one of more common types of attacks at the database layer of desktop and Web applications. SQL injection occurs when the intended effect of the SQL sentence is changed by inserting SQL keywords or special symbols [1]. The problem of SQL injection attacks has been traditionally addressed by using centralized architectures [2], [3]. Because this type of solution is incomplete, several types of intrusion detection system (IDS) solutions have been proposed [4]. Although IDSs are effective, there are a number of drawbacks such as a large number of false positives and negatives, limited learning capacity, and limited ability in adapting to changes in attack patterns.

This article presents a CBR-BDI [5] deliberative agent based on the BDI (*Belief*, *Desire*, *Intention*) [6] model specifically designed for the detection and prevention of SQL injection attacks in database layers. Our study applies a novel case-based reasoning (CBR) [7] [8] classification mechanism that incorporates a mixture of neural networks capable of making short term predictions [9].

This proposal is an innovative approach that addresses the problem of SQL injection attacks by means of a distributed artificial intelligence technique. Specifically, it combines the characteristics of multiagent systems such as autonomy, pro-activity, social relations, etc., [5] with CBR [7]. CBR Systems are adequate in dealing with

E. Corchado, A. Abraham, and W. Pedrycz (Eds.): HAIS 2008, LNAI 5271, pp. 46–53, 2008.

SQL injection attacks, insomuch as these systems find solutions to new problems by using previous experiences. This fact allows us to equip our classifier agents with a great capacity for adapting and learning, thus making them very adept in resolving problems in dynamic environments. The system developed within the scope of this work proposes a solution which combines a distributed approach and an advanced classification system, incorporating the best of both approaches.

The rest of the paper is structured as follows: section 2 presents the problem that has prompted most of this research work. Section 3 focuses on the structure of the classifier agent which facilitates the detection and prevention of malicious injection attacks, and section 4 explains in detail the classification model integrated within the classifier agent. Finally, section 5 describes how the classifier agent has been tested inside a multi-agent system and presents the results obtained.

2 SQL Injection Problem Description

A SQL injection attack affects the security of personal, social, financial and legal information for both individuals and organizations. 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]. SQL injection attacks occur when user input variables are not strongly typed, thus making them vulnerable to attack. As a result, these attacks can produce unauthorized handling of data, retrieval of confidential information, and in the worst possible case, taking over control of the application server [2]. One of the biggest problems with SQL injection is the various forms of vulnerabilities that exist. Some of the better known strategies, such as tautologies, syntax errors or illegal queries, and *union* operators, are easy to detect. However other strategies can be extremely complex due to the high number of variables that they can generate, thus making their detection very difficult. Some examples of these strategies are inference mechanisms, data storage procedures, and alternative encoding.

Traditional security mechanisms such as firewalls or IDSs are not very efficient in detecting and preventing these types of attacks. Other approaches based on string analysis, along with dynamic and static analyses such as AMNESIA (Analysis and Monitoring for Neutralizing SQL Injection Attacks) [2], have the disadvantage of addressing just one part of the problem, and therefore deliver only a partial solution. Moreover, the approaches based on models for detecting SQL injection attacks are very sensitive. With only slight variations of accuracy, they generate a large number of false positive and negatives.

Some innovative proposals are incorporating artificial intelligence and hybrids systems. Web Application Vulnerability and Error Scanner (WAVES) [10] uses a black-box technique which includes a machine learning approach. Valeur [4] presents an IDS approach which 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. Rietta [11] proposed an IDS at the application layer using an anomaly detection model which assumes certain behaviour of the traffic generated by the SQL queries; that is, elements within the query (sub-queries, literals, keyword

SQL). It also applies general statistics and proposes grouping the queries according to SQL commands and then comparing them against a previously built model. The SQL query that deviates from the normal profile is rejected. Finally, Skaruz [12] proposes the use of a neural network. The detection problem becomes a time serial prediction problem. All of these approaches have the problem of producing a large number of false positive and false negative. In the case of the IDS systems, they are unable to recognize unknown attacks because they depend on a signature database.

Our technique is based on BDI agents that incorporate a CBR mechanism. This is a novel approach in detecting and preventing SQL injection attacks. This strategy, which is depicted in the following sections, offers a robust and flexible solution for confronting SQL injection attacks, thus improving and surpassing previous approaches.

3 Classifier Agent Internal Structure

Agents are characterized by their autonomy; which gives them the ability to work in independently and real-time environments [13]. Because of this and other capacities they have, agents are being integrated into security approaches, such as is the case with IDS. Some applications of these systems can be found in [14], [15]. However, the use of agents in these systems is geared towards the retrieval of information in distributed environments, thus taking advantage of their mobility capacity. The classifier agent depicted in this work is the core of a multi-agent architecture, focused on detecting and preventing SQL injection attacks. The classifier agent interacts with other complementary agents, which are dedicated to tasks such as monitoring traffic, pattern matching, manage and interacting with both the user and the database. This agent is in charge of classifying SQL queries by means of an anomaly detection approach.

In our work, the agents are based on a BDI model in which beliefs are used as cognitive aptitudes, desires as motivational aptitudes, and intentions as deliberative aptitudes in the agents [5]. However, in order to focus on the problem of SQL injection, it was necessary to provide the agents with a greater capacity of learning and adaptation, as well as, a greater level of autonomy than a pure BDI model possesses. This is possible by providing the classifier agents with a CBR mechanism [7], which allows them to "reason" on their own and adapt to changes in the patterns of attacks. Working with this type of systems, the key concept is that of "case". A case is defined as a previous experience and is composed of three elements: a description of the problem that depicts the initial problem; a solution that describes the sequence of actions performed in order to solve the problem; and the final state, which describes the state that has been achieved once the solution is applied. To introduce a CBR motor into a BDI agent, we represent CBR system cases using BDI and implement a CBR cycle which consists of four steps: retrieve, reuse, revise and retain [7][16].

The classifier CBR-BDI [5] agent analyses a new query at the data base layer, executing a CBR cycle. In the retrieval phase, the cases that present a description similar to that of the new problem are recovered. The case description is based on elements of the SQL query that are extracted through a syntactic analyze process applied to the SQL text string. The process of retrieving the cases involves a similarity algorithm.

Once the similar cases have been recovered, the reuse phase starts adapting the solution in order to obtain the best case output for the given object of study. In this phase, two neural networks are responsible for receiving the variables as input data in order to produce an output response. This is done by a mixture of neural networks which use both sigmoidal activation functions and tangential functions. The next phase of the CBR cycle is the revise phase in which an expert evaluates the proposed solution. Finally, the retain phase is carried out where the system learns from its own experiences from each of the previous phases and updates the database with the solution obtained in the query classification.

As has been mentioned above, the classifier CBR-BDI agent is the core of a multi-agent architecture geared towards to detect and classify SQL injection attack in distributed environments. Figure 1 shows the classifier CBR-BDI agent in the multi-agent architecture.

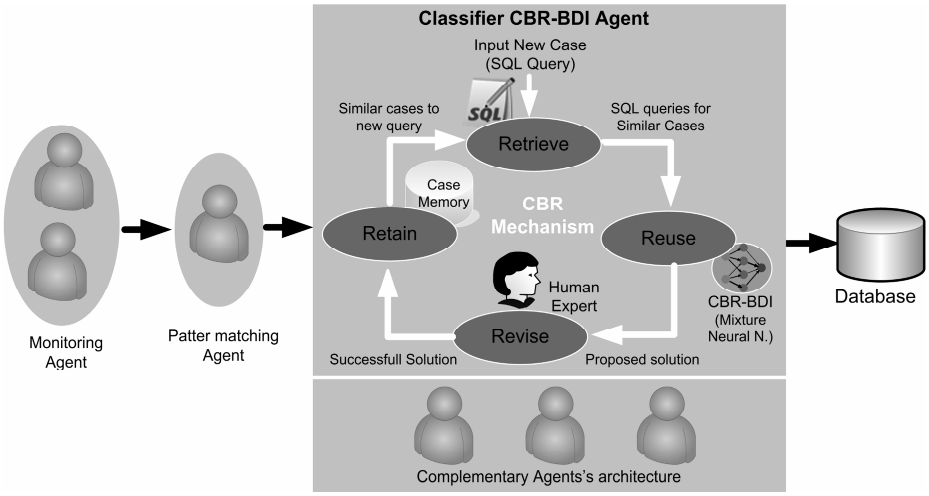

Fig. 1. CBR-BDI agent in the multi-agent architecture

All the SQL queries are filtered by the classifier CBR-BDI agent. The query considered legal is executed on the database otherwise the query considered malicious is rejected. In the next section, the classification mechanism used in the reuse phase by the agent is presented.

4 Mechanism for the Classification of SQL Injection Attacks

The classifier agent presented in section 3 incorporates a case-based reasoning system that allows the prevention and detection of anomalies by means of a prediction model based on neural networks, configured for short-term predictions of intrusions by SQL injections. This mechanism uses a memory of cases which identifies past experiences with the corresponding indicators that characterize each of the attacks. This paper presents a novel classification system that combines the advantages of the CBR systems, such as learning and adaptation, with the predictive capabilities of a mixture of

neural networks. These features make the system innovative for this type of attack, and make it very appropriate for its use in different environments. The proposed mechanism is responsible for classifying SQL database requests made by users. When a user makes a new request by means of a SQL query, it is checked by an agent in charge of detection by pattern matching. If it is found an attack firm on the SQL string, then the SQL query is labelled as suspicious. Independently of the results of the pattern matching, the SQL query is send to a CBR-BDI agent. This query is transformed to a new case of input for a CBR mechanism. The case is constituted by elements extracted of the SQL string (Affected_Table, Affected_Field, Command_Type, Word_GroupBy, Word_OrderBy, Word_Having, Number_And, Number_Or, Number_Literales, Number_LOL, Length_SQL_String, Cost_Time_CPU, Start_time, End_time, Query_Category). The CBR mechanism needs a memory of cases dating back at least 4 weeks for the training stage of the neural networks.

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 request, 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 request. The cases recovered are used to train the mixture of neural networks implemented in the recovery phase; the neural network with the sigmoidal function is trained with the recovered cases that were an attack or not, whereas the neural network with hyperbolic function is trained with all the recovered cases (including the suspects). A preliminary analysis of correlations is required to determine the number of neurons of the input layer of the neuronal networks. Additionally, it is to normalize the data (i.e., all data must be values in the interval [0.1]). The data used to train the mixture of networks must not be correlated. With the cases stored after eliminating correlated cases, the entries for training the mixture of networks are normalized. It is considered to be two neural networks. The result obtained using a mixture of the outputs of the networks provides a balanced response and avoids individual tendencies (always taking into account the weights that determine which of the two networks is more optimal).

La mixture of the neural networks intents to solve the individual tendencies by use an only neural network. If one only network with a sigmoidal activation function is used, then the result provided by the network would tend to be attack or not attack, and no suspects would be detected. On the other hand, if only one network with a hyperbolic tangent activation is used, then a potential problem could exist in which the majority of the results would be identified as suspect although they are being clearly attack or not attack. Figure 2 shows the mixture of the neural networks with their activation function.

The mixture provides a more efficient configuration of the networks, since the global result is determined by merging two filters. This way, if the two networks classify the user request as an attack, so too will the mixture; and if both agree that it is not an attack, the mixture will as well. If there is no concurrence, the system uses the result of the network with the least error in the training process or classifies it as a suspect. In the reuse phase the two networks are trained by a back-propagation algorithm for the same set of training patterns (in particular, these neural networks are named Multilayer Perceptron), using a sigmoidal activation function (which will take values in [0.1], where 0 = Illegal and 1 = legal) for a Multilayer Perceptron and a hyperbolic tangent activation function for the other Multilayer Perceptron (which take values in [-1.1], where -1 = Suspect, 0 = illegal and 1 = legal).

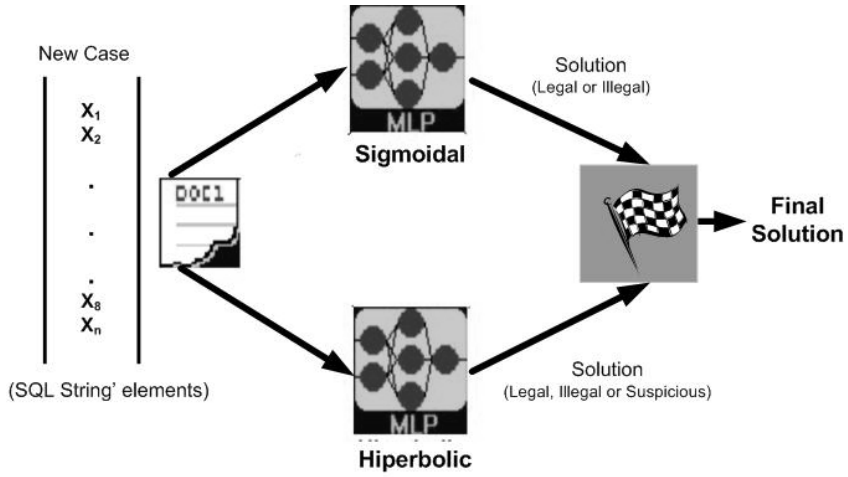

Fig. 2. Mixture of the Neural Networks

The response of both networks is combined, obtaining the mixture of networks denoted by y^2; where the superscript indicates the number of mixtured networks

$$y^2 = \frac{1}{\sum_{r=1}^{2} e^{-|1-r|}} \sum_{r=1}^{2} e^{-|1-r|} y^r \tag{1}$$

The number of neurons in the output layer for both Multilayer Perceptrons is 1, and is responsible for deciding whether or not there is an attack. The error of the training phase for each of the neural networks, can be quantified with formula (2), where P is the total number of training patterns.

$$Error = \frac{1}{P} \sum_{i=1}^{P} \left| \frac{Forecast_P - T\arg et_P}{T\arg et_P} \right| \tag{2}$$

The review stage is performed by an expert, and depending on his opinions, a decision is made as to 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.

5 Results and Conclusions

The problem of SQL injection attacks on databases supposes a serious threat against information systems. This paper has presented a new classification system for detecting SQL injection attacks which combines the advantages of multi-agent systems, such as autonomy and distributed problem solving, with the adaptation and learning capabilities of CBR systems. Additionally, the system incorporates the prediction capabilities that characterize neural networks. An innovative model has been presented that provides a significant reduction of the error rate during the classification of attacks. To check the validity of the proposed model, a series of test were

elaborated which were executed on a memory of cases, specifically developed for these tests, and which generated attack consults. The results shown in Table 1 are promising: it is possible to observe different techniques for predicting attacks at the database layer and the errors associated with misclassifications. All the techniques presented in Table 1 have been applied under similar conditions to the same set of cases, taking the same problem into account in order to obtain a new case common to all the methods. Note that the technique proposed in this article provides the best results, with an error in only 0.537% of the cases.

Table 1. Results obtained after testing different classification techniques

Forecasting Techniques	Successful (%)	Approximated Time (secs)
CBR-BDI Agent (mixture NN)	99.5	2
Back-Propagation Neural Networks	99.2	2
Bayesian Forecasting Method	98.2	11
Exponential Regression	97.8	9
Polynomial Regression	97.7	8
Linear Regression	97.6	5

As shown in Table 2, the Bayesian method is the most accurate statistical method since it is based on the likelihood of the events observed. But it has the disadvantage of determining the initial parameters of the algorithm, although it is the fastest of the statistical methods. Taking the errors obtained with the different methods into account, after the CBR-BDI Agent together with the mixture of neural networks and Bayesian methods we find the regression models. Because of the non linear behaviour of the hackers, linear regression offers the worst results, followed by the polynomial and exponential regression. This can be explained by looking at hacker behaviour: as the hackers break security measures, the time for their attacks to obtain information decreases exponentially. The empirical results show that the best methods are those that involve the use of neural networks and, if we consider a mixture of two neural networks, the predictions are notably improved. These methods are more accurate than statistical methods for detecting attacks to databases because the behaviour of the hacker is not linear. The solution presented is an innovative approach to detect SQL injection attack, when it is based on multi-agent system, CBR mechanism and a sophisticated technique using a mixture of neural networks.

Acknowledgments. This development has been partially supported by the Spanish Ministry of Science project TIN2006-14630-C03-03.

References

1. Anley, C.: Advanced SQL Injection In SQL Server Applications (2002), http://www.nextgenss.com/papers/advanced-sql-injection.pdf
2. Halfond, W., Orso, A.: AMNESIA: analysis and monitoring for neutralizing SQL-injection attacks. In: ASE 2005: 20th IEEE/ACM international Conference on Automated software engineering, pp. 174–183. ACM, New York (2005)

3. Wassermann, G., Gould, C., Su, Z., Devanbu, P.: Static Checking of Dynamically Generated Queries in Database Applications. ACM Transactions on Software Engineering and Methodology 16, 14 (2007)
4. Valeur, F., Mutz, D., Vigna, G.: A Learning-Based Approach to the Detection of SQL Attacks. In: Julisch, K., Krügel, C. (eds.) DIMVA 2005. LNCS, vol. 3548, pp. 123–140. Springer, Heidelberg (2005)
5. Corchado, J.M., Pavón, J., Corchado, E.S., Castillo, L.F.: Development of CBR-BDI Agents. In: Advances in Case-Based Reasoning. Springer, Heidelberg (2004)
6. Woolridge, M., Wooldridge, M.J.: Introduction to Multiagent Systems. John Wiley & Sons, New York (2002)
7. Corchado, J.M., Laza, R., Borrajo, L., De Luis, Y.A., Valiño, M.: Increasing the Autonomy of Deliberative Agents with a Case-Based Reasoning System. International Journal of Computational Intelligence and Applications 3(1), 101–118 (2003)
8. Fdez-Riverola, F., Iglesias, E.L., Daz, F., Méndez, J.R., Corchado, J.M.: SpamHunting: An instance-based reasoning system for spam labelling and filtering. Decision Support System 43(3), 722–736 (2007)
9. Ramasubramanian, P., Kannan, A.: Quickprop Neural Network Ensemble Forecasting a Database Intrusion Prediction System. Neural Information Processing 5, 847–852 (2004)
10. Huang, Y., Huang, S., Lin, T., Tsai, C.: Web application security assessment by fault injection and behavior monitoring, pp. 148–159. ACM, New York (2003)
11. Rietta, F.: Application layer intrusion detection for SQL injection. In: 44th annual Southeast regional conference, pp. 531–536. ACM, New York (2006)
12. Skaruz, J., Seredynski, F.: Recurrent neural networks towards detection of SQL attacks. In: Parallel and Distributed Processing Symposium, 2007. IPDPS 2007, pp. 1–8. IEEE International, Los Alamitos (2007)
13. Carrascosa, C., Bajo, J., Julian, V., Corchado, J.M., Botti, V.: Hybrid multi-agent architecture as a real-time problem-solving model. Expert System with Application 34, 2–17 (2008)
14. Kussul, N., Shelestov, A., Sidorenko, A., Skakun, S., Veremeenko, Y.: Intelligent multiagent information security system, Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. In: Proceedings of the Second IEEE International Workshop, pp. 120–122 (2003)
15. Abraham, A., Jain, R., Thomas, J., Han, S.Y.: D-SCIDS: distributed soft computing intrusion detection system. J. Netw. Comput. Appl. 30, 81–98 (2007)
16. Corchado, J.M., Bajo, J., Abraham, A.: GerAmi: Improving Healthcare Delivery in Geriatric Residences. Intelligent Systems 23, 19–25 (2008)