A Multiagent Based Strategy for Detecting Attacks in Databases in a Distributed Mode

Cristian Pinzón[1], Yanira De Paz[2], Javier Bajo[1]

[1]Departamento Informática y Automática, Universidad de Salamanca,

Plaza de la Merced s/n 37008, Salamanca, Spain

[2]Universidad Europea de Madrid, Tajo s/n 28670, Villaviciosa de Odón, Spain

cristian_ivanp@usal.es; yanira@usal.es; jbajope@usal.es

Abstract This paper presents a distributed hierarchical multiagent architecture for detecting SQL injection attacks against databases. It uses a novel strategy, which is supported by a Case-Based Reasoning mechanism, which provides to the classifier agents with a great capacity of learning and adaptation to face this type of attack. The architecture combines strategies of intrusion detection systems such as misuse detection and anomaly detection. It has been tested and the results are presented in this paper.

Keywords: Multi-agent, SQL injection, Security database, case-based reasoning, IDS.

1 Introduction

The exponential growth of the computer network and the increase in the interconnection between networks has extended the offer of new services within the cyberspace [1]. The information volume with a sensitive value for the organizations is stored on information structures denominated databases and this information generally is transmitted across computer network. Databases are the core of many information systems, reason for which databases are increasingly coming under large number of attacks. Every day are founded new vulnerabilities in security systems intended to protect databases. These vulnerabilities are used by hackers in order to carry out attacks on the stored data. A special intrusion type within of da-

tabases is the SQL injection attack, which occurs when the intended effect of a SQL sentence is changed by inserting SQL keywords or special symbols [2].

Nowadays, the majority of approaches had addressed the problem of SQL injection attack from a centralized perspective, such as the one described by [3] and [2]. However, the solutions are limited to solve only a part of the problem. Regarding this, other approaches had implemented strategies based on intrusion detection systems in order to block a SQL injection attack, such as [4] and [5]. These proposals have as main drawbacks the highest error rate and a limited capacity of learning and adapting when changes occur in the patterns of attacks.

Our proposal aims the SQL injection attacks in a distributed, dynamic and flexible mode. This proposal is founded in a hierarchical multiagent architecture using agents based on the BDI (Belief, Desire and Intention) model [6]. Agents are typically integrated into multiagent systems or agent societies, exchanging information and resolving problems in a distributed way [7]. Agents can be characterized through their capacities such as autonomy, reactivity, pro-activity, social abilities, reasoning, learning and mobility [6]. Our proposal incorporates classifier agents supported by a Case-based reasoning mechanism (CBR) [8] that includes a mixture of neural networks capable of making short term predictions [9]. Our multi-agent architecture is adequate to block the SQL injection attack, because it is designed for working in distributed and dynamic environments.

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 details of the multiagent architecture, the different levels of the architecture, the interaction possibilities and communication between the agents; 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 Attacks description

The impact of a SQL injection attack in a database has many consequences within of the organization and individuals. Personal, financial and legal information is compromised when this type attack is carried out. 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 on the original SQL command that will be executed at the database layer of an application [10], [2]. The results of this attack can produce unauthorized handling of data, retrieval of confidential information, and in the worst possible case, taking over control of application server. One particular inconvenient of the SQL injection attack is the biggest number of variants. Some strategies can be extremely complex due to the high number of variables that they can generate, thus making their detection very difficult.

Some approaches based on firewall and intrusion detection system (IDS) are a few effective due the strategy of detection, which requires an updated patterns database. Other approaches more specifics to face SQL injection attacks are founded in a technique of string analysis, some carrying out static analysis such as JSA (Java String Analyzer) [3]. Other more complex using dynamic and hybrid analysis is AMNESIA (Analysis and Monitoring for Neutralizing SQL Injection Attacks) [2]. These approaches generally have as main drawback that aim just one part of the problem, 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

Several approaches based on artificial techniques and hybrid systems propose a novel alternative. Web Application Vulnerability and Error Scanner (WAVES) [11] 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 transactions are used during the training phase prior to monitoring and classifying malicious accesses. Rietta [5] put forward an IDS at the application layer using an anomaly detection model. Finally, Skaruz [12] proposes the use of a recurrent neural network (RNN). The detection problem is became a time serial prediction problem. Generally, this approaches present as main problem, generating a large number of false positive and false negative. In the case of the IDSs systems, they are unable to recognize unknown attacks because they depend on a signature database that requires a dynamic updating.

The multi-agent architecture aims the problem from a distributed context more dynamic and flexible to work according to the device used to execute a SQL injection attack. The incorporation of a CBR technique to the classifier agents allows them to offer a robust learning and adaptable solution that respond to an unlimited number of variants of SQL injection attacks. Finally, a combination of strategies based on intrusion detection system that covers misuse detection and anomaly detection grants to the architecture a highest level performance in the tasks of classification. It is important to highlight that the combination of these strategies reduce at a minimum the false positives and false negatives.

3 The Agent Based Architecture

Agents are characterized by their autonomy; which gives them the ability to work independently in real-time environments [13]. Agents are especially adequate to work in dynamic and distributed environments when they are integrated into multiagent systems [7]. Environments especially complex for protecting are the application that working with a relational database to provide information to different users. A distributed multiagent architecture presents a great capacity of error reco-

vering, allowing using autonomous agents for decision making and adapting a new variant of a SQL injection attack.

Several architectures have been proposed to build deliberative agents, most of them based on the BDI (Belief, Desire, Intention) model [6]. In this model, the intern structure of the agents and their election capacities are based on mental aptitudes [14]. The main characteristic of the architecture proposes in this paper is the incorporation of CBR-BDI deliberative agents, which are capable protecting databases of the applications by means of SQL queries classification. CBR-BDI classifier agents use an intrusion detection techniques (anomaly detection) [15], identifying elements of SQL injection in the database queries.

Our proposal is a distributed hierarchical multiagent architecture integrated for 4 levels with distinct BDI agents. The hierarchical structure allows distributing tasks on levels of the architecture, defining specific responsibilities, even though the interaction and communication between the agents is continuous in order to request services and delivering results. The architecture presented in figure 1 shows the four levels and BDI agents organized according to their roles.

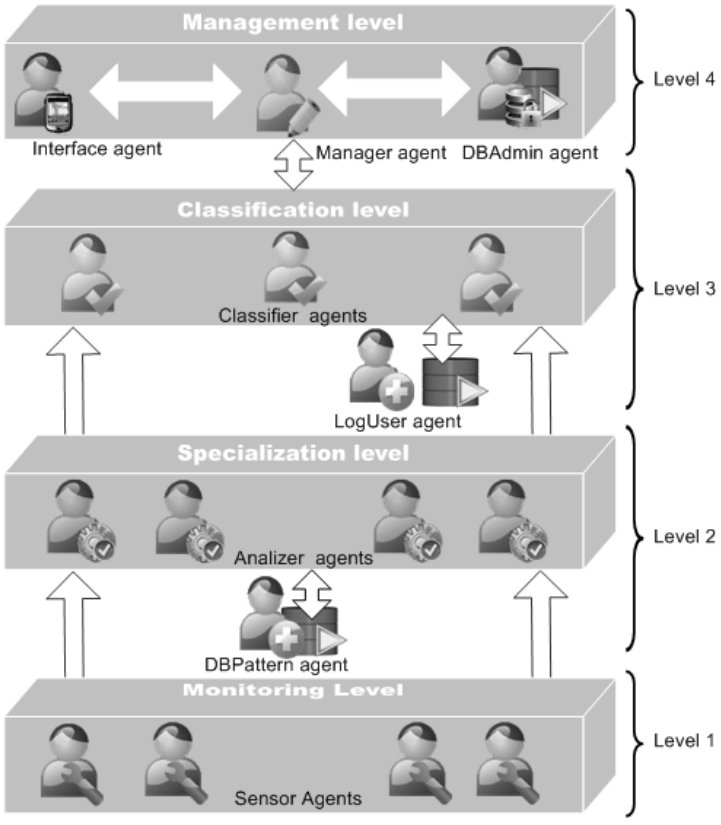

Fig. 1. Multi-agent architecture with BDI agents

Next, the details of the functionality of each agent located at each level of the architecture are described:

- Sensor agents: These agents are located at the level 1 of the architecture. They are incorporated at each device with access to database. They have been assigned 3 functions: the capture of datagrams, the ordering of the TCP fragments for extracting the SQL query string, and the syntactic analysis. The tasks of the Sensor agents end when the results (the SQL string transformed by the analysis, the result of the analysis and the user data) are sent to the next level of the architecture.
- Analyzer agents: These agents are located at the level 2. They receive the transformed SQL string, the result of the analysis and user data. This information is assigned to a specific agent. The Analyzer agent receives the data sent by the Sensor agent and then it executes searching process and compares known SQL string patterns. These SQL string patterns are stored in a previously built database. The Analyzer agent works in coordination with the DbPattern agent for searching and saving SQL string patterns on the database. The creation of the Analyzer agents is dynamic and it depends on the workload at a given time. When an Analyzer agent is created it receives an instance number. The Analyzer agent finishes its task when it sends results to the next level. The results consist of the SQL string transformed, the result of the analysis, the user data and the result by misuse detection.
- DbPattern Agent: this agent is located at the level 2. It is the responsible to save the new SQL string patterns on the database and search for patterns when the Analyzer agent requests it.
- Classifier agents: These agents are in charge of carrying out the task of classification of SQL queries. These agents are located at the level 3, and implements a technique based on anomaly detection. A classifier agent incorporates a CBR mechanism, which allows the evaluation of the new SQL query through the search of similar cases stored in the memory cases, looking for a solution to the new problem. At this level there exist *n* instances of the classifier agents and their creation is dynamic and depends of the workload at a given time. Classifier agents work together with the LogUser agent and Manager agent for carrying out their tasks.
- LogUser agent: this agent is located at the level 3, records the actions of the user and searches for the user profile (the historical profile and the user statistics) when it is requested by a Classifier agent.
- Manager agent: this agent works at the level 4. It has many tasks: monitors, evaluates and coordinates the classifiers agents. It ensures the capacity for errors recovering and the distribution of the workload within the architecture. An expert through of the Manager agent allows evaluating the results of the classification and the final decision that have been carried out. Moreover, this agent is in charge of managing the alert of attacks and the coordination of the required actions to take over an attack when it has been detected. A Classifier

agent is promoted to be a Manager agent when the Manager agent fails. This agent is selected by means of a voting method between the classifier agents.

- DbAdmin agent: this agent is also located at the level 4. It is in charge of executing the query on the database and obtaining the results.
- Interface agent: this agent is located at the level 4. The Interface agent allows the interaction of the user of the security system with the architecture. The interface agent communicates the details of an attack to the security personnel when an attack is detected. This agent has the capacity to work on mobile devices such as PDAs, mobile telephones and laptops. This capacity allows a ubiquitous communication. Moreover, the interface agent provides an access mechanism for carrying out certain adjustments in the configuration of the architecture and the verification of the status of the Manager agent.

4 Classifier Model of SQL Injection Attacks

The CBR-BDI classifier agent [16] 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 very appropriate for its use in dynamic environments. Working with this type of systems, the key concept is the case". 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. A CBR cycle consists of four steps: retrieve, reuse, revise and retain.

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

(a) Problem Description that describes the initial information available for generating a plan. As can see in table 1, the problem description 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. As can see in the table 1, it contains 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		Solution fields	
IdCase	Integer	Idcase	Integer
Sesion	Session	Classification_Query	Integer
User	String		
IP_Adress	String		

Query_SQL	Query_SQL
Affected_table	Integer
Affected_field	Integer
Command_type	Integer
Word_GroupBy	Boolean
Word_Having	Boolean
Word_OrderBy	Boolean
Numer_And	Integer
Numer_Or	Integer
Number_literals	Integer
Number_LOL	Integer
Length_SQL_String	Integer
Cost_Time_CPU	Float
Start_Time_Execution	Time
End_Time_Execution	Time
Query_Category	Integer

In the following section, the performance of the classifying system is described in detail. The proposed mechanism is responsible to classify SQL database queries made by users. When a user makes a new request, it is checked for pattern matching. This being the case, it is automatically identified as an attack. In order to identify the rest of the SQL attacks, the mechanism uses CBR, which must have a memory of cases dating back at least 4 weeks, and store the variables showed in the Table 1.

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 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).

With la mixture of the neural network we mean to detect attacks, so if one only network with a sigmoidal activation function was 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 was used, then a potential problem could exist in which the majority of the results would be identified as suspect although they were clearly attack or not attack. 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). 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 to decide 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 advantages of multiagent 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 was 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 2 are promising: it is possible to observe different techniques to predict attacks at the database layer and the errors associated with misclassifications. All the techniques presented in Table 2 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 0.5% of the cases.

Table 2. 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. With a mixture of two neural networks, the predictions are notably improved.

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

References

1. Kandula S, Singh S (2002) Argus: A distributed network-intrusion detection system. In: Proceedings of the 3rd International SANE'02 Conference. Netherlands
2. Halfond W, Orso A (2005) AMNESIA: Analysis and Monitoring for Neutralizing SQL-injection Attacks. In: 20th IEEE/ACM international Conference on Automated software engineering, pp. 174-183. ACM, New York
3. Christensen AS, Moller A, Schwartzbach MI (2003) Precise Analysis of String Expressions. In: 10th International Static Analysis Symposium, pp. 1-18. Springer-Verlag
4. Valeur F, Mutz D, Vigna G (2005) A Learning-Based Approach to the Detection of SQL Attacks. In: Conference on Detection of Intrusions and Malware and Vulnerability Assessment, pp. 123-140. Vienna
5. Rietta F (2006) Application layer intrusion detection for SQL injection. In: 44th annual Southeast regional conference, pp. 531-536. ACM, New York
6. Woolridge M, Wooldridge MJ (2002) Introduction to Multiagent Systems, John Wiley & Sons, Inc., New York
7. Corchado JM, Bajo J, Abraham A (2008) GerAmi: Improving Healthcare Delivery in Geriatric Residences, Vol. 23, pp. 19-25. Intelligent Systems, IEEE
8. Corchado JM, Laza R, Borrajo L, De Luis YA, Valiño M (2003) Increasing the Autonomy of Deliberative Agents with a Case-Based Reasoning System. Vol 3, pp. 101-118. International Journal of Computational Intelligence and Applications
9. Ramasubramanian P, Kannan A (2004) Quickprop Neural Network Ensemble Forecasting a Database Intrusion Prediction System. In: 7th International Conference Artificial on Intelligence and Soft Computing, Neural Information Processing. Vol 5, pp. 847-852
10. Anley C (2002) Advanced SQL Injection In SQL Server Applications. http://nextgenss.com/papers/advanced-sql-injection.pdf. Accessed 02 march 2008
11. Huang Y, Huang S, Lin T, Tsai C (2003) Web application security assessment by fault injection and behavior monitoring. In: 12th international conference on World Wide Web, pp. 148-159. ACM
12. Skaruz J, Seredynski F (2007) Recurrent neural networks towards detection of SQL attacks. In: 21th International Parallel and Distributed Processing Symposium, pp. 1-8. IEEE International
13. Carrascosa C, Bajo J, Julian V, Corchado JM, Botti V (2008) Hybrid multi-agent architecture as a real-time problem-solving model, Vol. 34, pp. 2-17. Expert Systems With Applications
14. Corchado JM, Pavón J, Corchado ES, Castillo LF (2004) Development of CBR-BDI Agents. Springer Berlin / Heidelberg, ed. Advances in Case-Based Reasoning
15. Abraham A, Jain R, Thomas J, Han, SY (2007) D-SCIDS: distributed soft computing intrusion detection system, Vol. 30, pp. 81-98. Journal of Network and Computer Applications
16. Pinzon C, De Paz Y, Cano R (2008) Classification Agent-Based Techniques for Detecting Intrusions in Databases. In: 3rd International Workshop on Hybrid Artificial Intelligence Systems