A Distributed Hierarchical Multi-agent Architecture for Detecting Injections in SQL Queries

Cristian Pinzón[1], Juan F. De Paz[1], Álvaro Herrero[2], Emilio Corchado[1], Javier Bajo[1]

Abstract. SQL injections consist in inserting keywords and special symbols in the parameters of SQL queries to gain illegitimate access to a database. They are usually identified by analyzing the input parameters and removing the special symbols. In the case of websites, due to the great amount of queries and parameters, it is very common to find parameters without checking that allow bad-intentioned users to introduce keywords and special symbols. This work proposes a distributed architecture based on multi-agent systems that is able to detect SQL injection attacks. The multi-agent architecture incorporates cased-based reasoning, neural networks and support vector machines in order to classify and visualize the queries, allowing the detection and identification of SQL injections. The approach has been tested and the experimental results are presented in this paper.

Keywords: SQL injection, Database Security, Intrusion Detection Systems, Multi-agent Systems, Case-based Reasoning, Unsupervised Projection Models.

1 Introduction

A potential security problem of databases is the SQL injection attack. This attack takes place when a hacker changes the semantic or syntactic logic of an SQL text string by inserting SQL keywords or special symbols within the original SQL command. The SQL query will then be executed at the database layer of an appli-

[1] Departamento Informática y Automática, Universidad de Salamanca. Plaza de la Merced s/n, 37008, Salamanca, Spain. {cristian_ivanp,fcofds, escorchado, jbajope}@usal.es.

[2] Department of Civil Engineering. University of Burgos. C/ Francisco de Vitoria S/N, 09006, Burgos, Spain. ahcosio@ubu.es

cation [1], [6], being extremely dangerous in the case of online applications as the answer to the query will be available through a web browser. 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 the application server.

Nowadays, this type of attack has been handled from distinct perspectives. The string analysis [7] has been the support of many others approaches such as [1] and [8], which carried out a more complete analysis applying a dynamic and hybrid treatment over the SQL string. In other cases, computational intelligence techniques have been applied to face the SQL injection attack, such as [9], [2], [3] with WAVES (Web Application Vulnerability and Error Scanner). These approaches apply machine learning techniques based on a dataset of legal transactions and artificial neural networks. Usually, many approaches present a poor performance, with high error rates (both false positive and false negative rates). The performance of misuse-based intrusion detection systems depend on the database, which requires a continue update in order to detect new attacks.

The proposal presented in this work tackles the SQL injection attack problem through a distributed hierarchical multi-agent architecture to detect SQL attacks in queries. The key component is the intelligent agent CBRid4SQL (a Case-Based Reasoning Intrusion Detector), capable of detecting attacks based on SQL code injection. CBRid4SQL is an agent that addresses the SQL injection problem from the Intrusion Detection standpoint by combining different Computational Intelligence techniques. This is the principal component of a distributed hierarchical multi-agent system aimed at detecting a wide range of attacks in dynamic and distributed environments. CBRid4SQL is a CBR agent [13] characterized by the integration of several techniques within the CBR mechanism. This mechanism provides the agents with a great level of adaptation and learning capability, since CBR systems make use of past experiences to solve new problems [13]. This is very effective for blocking SQL injection attacks as the mechanism uses a strategy based on anomaly detection [14]. The multi-agent system incorporates classification and visualization techniques in the different phases of the reasoning cycle.

The rest of the paper is structured as follows: section 2 focuses on the details of the proposed multiagent architecture while section 3 comprehensively explains the integrated classification model. Finally, section 4 describes how the proposed agent has been tested in the frame of a multi-agent system and presents the obtained results.

2 A Multi-agent Architecture for the Detection of SQL Injection

The agents are characterized through their capacities such as autonomy, reactivity, pro-activity, social abilities, reasoning, learning and mobility [4]. One of the main features of agents is their ability to carry out cooperative and collaborative work,

when they are grouped into multi-agent systems to solve problems in a distributed way [11]. These features make the agents suitable to face the SQL injection attack problem. A distributed hierarchical multi-agent system presents a great capacity for the distribution of tasks and responsibilities, error recovering, adaptation to new changes and high level of learning. These factors are key to achieve a robust and efficient solution. One main innovation of the proposed architecture is the use of a CBR agent [5], which presents a great capacity of learning and adaptation. This CBR mechanism additionally incorporates a mixture of a neural network [10] and support vector machine (SVM) [12] in order to identify SQL injections.

The types of agents within the architecture are described as follows:

- **Sensor agents**: Located in each of the devices accessing the database. They have 3 specific functions: a) the capture of datagrams launched by the devices. b) Order TCP fragments to extract the request's SQL string. c) Syntactic analysis of the request's SQL string. The duties of the agent Sensor end when the results (the SQL string transformed by the analysis, the result of the analysis of the SQL string and the user data) are sent to the next agent at the hierarchy of the classification process.
- **FingerPrint agents**: The numbers of agents FingerPrint depend on the workload at a given time. An agent FingerPrint receives the information of a Sensor agent and executes a pattern matching known attacks stored at a previously built database. The FingerPrint agent finishes its task when it sends its results to the Anomaly agent. The results of the FingerPrint agent consist of the SQL string transformed by the analysis, the result of the analysis of the SQL string, the user data and the results achieved by pattern matching.
- **CBRid4SQL agents**: These agents are based on the CBR model. They are the key component of the classification process. Their strategy is based on a case-based reasoning mechanism that incorporates a mixture of neural networks. These agents retrieve those past cases that are the most similar to the new case to be classified, train the neural networks with the recovered cases and generate the final classification for the new case. The result of the classification is sent to the Manager agent for the evaluation.
- **Manager agent**: This is the agent responsible for decision-making, evaluation and coordination of the overall operation of the architecture. It evaluates the final decisions for classifications, manages alerts of attacks and coordinates the necessary actions when an attack is detected.
- **Interface agent**: This 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. Its ability to work on mobile devices allows a ubiquitous communication to manage the alerts immediately.

Fig. 1 depicts the hierarchical multi-agent architecture showing different types of agents in charge of the classification of SQL queries.

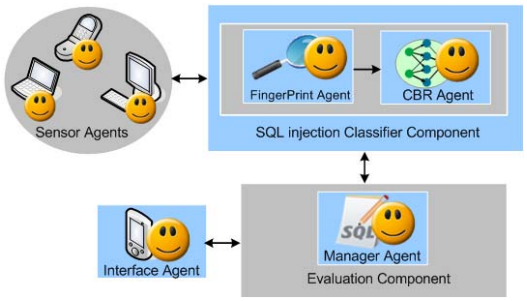

Fig. 1. Description of the distributed hierarchical multi-agent architecture.

3 Classifier CBR Agent

In this section the CBRid4SQL agent is presented, with special attention paid to its internal structure and the classification mechanism of SQL attacks. This mechanism combines the advantages of CBR systems, such as learning and adaptation, with the predictive capabilities of a combination integrated by ANNs and SVMs. The use of this combination of techniques is based on the possibility of using two classifiers together to detect suspicious queries in the most reliable way possible.

In terms of CBR, the case is composed of the following elements of an SQL query: (a) Problem Description that describes the initial information available for generating a plan. The problem description consists of: case identification, user session and SQL query elements. (b) Solution that describes the action carried out in order to solve the problem description, in this case, prediction models. (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
Length_SQL_String	Integer
Cost_Time_CPU	Float
Start_Time_Execution	Time
End_Time_Execution	Time
Query_Category	Integer

A Multilayer Perceptron (MLP) and an SVM are applied simultaneously to carry out the prediction of the new query. Subsequently, a new inspection is performed which can be done automatically or by a human expert. In the case of the query resulting as suspicious, further inspection is carried out manually by a human expert. At this stage the most similar cases will be selected by means of a Growing Cell Structure (GCS) network [15], and then visualized by a dimensionality reduction technique which employs the neuronal model called Cooperative Maximum Likelihood Hebbian Learning (CMLHL) [16]. As a result of such visualization, the human expert will graphically see the relationship between the suspicious query and the recovered queries. During learning, memory information regarding the cases and models will be updated. The different stages of the CBR reasoning cycle associated with the system are comprehensively described in the following sections.

4.1. Retrieve

The retrieval phase consists of two phases; case retrieval and model retrieval. The case retrieval is performed by using the *Query_Category* attribute which retrieves queries from the case memory which were used for a similar query in accordance with the attributes of the new case. Subsequently, the models for the MLP and associated with the recovered cases are retrieved. The recovery of these memory models allows the improvement of the system's performance so that the time necessary for the creation of such models is considerably reduced, mainly in the case of the ANN training.

4.2. Reuse

The reuse phase initially considers the information of the retrieved cases and the recovered models of the MLP and the SVM. The combination of both techniques is fundamental in the reduction of the false negative rate. The inputs of the MLP an SVM are: *Query_SQL, Affected_table, Affected_field, Command_type, Word_GroupBy, Word_Having, Word_OrderBy, Numer_And, Numer_Or, Num-*

ber_literals, *Number_LOL*, and *Length_SQL_String*. The number of neurons in the hidden layer of the MLP is *2n+1*, where *n* is the number of neurons in the input layer. Finally, there is only one neuron in the output layer. The activation function selected for the different layers has been the sigmoid.

As the neurons exiting from the hidden layer of the neural network contain sigmoidal neurons with values between [0, 1], the incoming variables are redefined so that their range falls between [0.2-0.8].

At the same time as the estimation through the use of neuronal networks is performed, estimation is also carried out by the SVM application, a supervised learning technique applied to the classification and regression of elements. The algorithm represents an extension of nonlinear model [12].

The selected kernel function in this problem was polynomial. The values used for the estimation are dominated by decision values and are related to the distance from the points to the hyperplane.

Once the output values for the ANN and the SVM are obtained, the mixture is performed by way of a weighted average in function of the error rate of each one of the techniques. Before carrying out the average, the values are normalized to the interval [0,1], as SVM provides positive and negative values and those of greater magnitude, so that it could affect the final value in greater measure if it is not redimensioned.

4.3. Revise

The revise phase can be manual or automatic depending on the output values. The automatic review is given for non-suspicious cases during the estimation obtained for the reuse phase. For cases detected as suspicious, with output values determined experimentally in the interval [0.35, 0.6], a review by a human expert is performed. As CBR learns, the interval values are automatically adjusted to the smallest of the false negatives. The greater limit is constantly maintained throughout the iterations. The review consists of recovering those queries similar to the current one together with their previous classifications. To do so, a clustering technique (for the selection of similar requests) and a neuronal model (for the reduction of dimensionality) are combined to generate an informative visualisation in 2D or 3D.

The selection of similar cases is carried out through the use of a neuronal GCS network, the different cases are distributed in meshes and the mesh containing the new case is selected. To visualize the cases (those in the selected mesh), the dimensionality of data is reduced by means of the CMLHL neuronal model [16] which performs Exploratory Projection Pursuit by unsupervised learning.

4.4 Retain

The learning phase updates the information of the new classified case and reconstructs the classifiers offline to leave the system available for new classifications. The ANN classifier is reconstructed only when an erroneous classification is produced. In the case of a reference to inspection of suspicious queries, information and classifiers are updated when the expert updates the information.

4 Results and Conclusions

To check the performance of the proposed model, experiments were run over a dataset with 518 select, 89 update and 77 delete queries (both legal an illegal). A sample query is:
select * from request_client, client where id_client = id and id = 'test' AND ORD(MID((CONCAT(CHAR(52), CHAR(52))), 1, 1)) > 63 AND '1'='1' OR id_client= 1

The accuracy reached by the CBRidSQL agent is 99% compared to 90.5% of Bayesian Network, 94.5% de Naive Bayes and 87.2% of linear regressions.

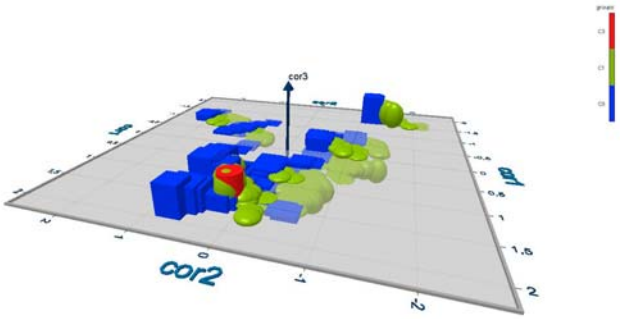

Fig. 2. Revision phase for a suspicious query.

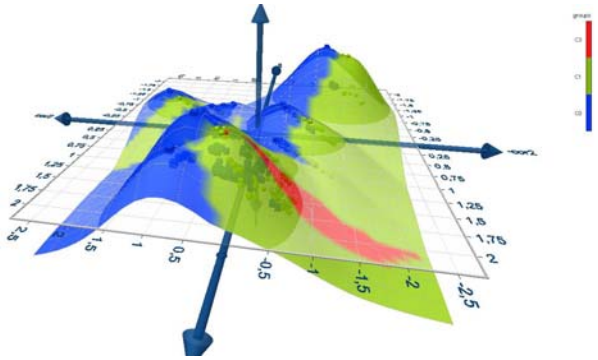

Fig. 3. Revision phase for a suspicious query with a surface map.

In Figs. 2 and 3 it can be seen the visualization of a suspicious query applying the CMLHL model. The suspicious query (C2) is shown in red, the legal queries (C0) are depicted in blue and the illegal queries (C1) in red. In Fig. 2, the axes represent the variables in low dimensionality. In Fig. 3 it is represented the same information with a surface map. In both images, it can be seen that the suspicious query is represented with the illegal queries, and therefore the query would be classified as an attack.

SQL injection attacks on databases are a serious threat against information systems. As conclusions of this work, we can state the this paper presents a distributed hierarchical multi-agent architecture incorporating a novel type of agent based on the CBR model specially designed for detecting and blocking such attacks. This CBR agent handles a great adaptation and learning capacities using a CBR mechanism. In addition, it incorporates the prediction capabilities that characterize neural networks and SVM. As a result, an innovative and robust solution is presented allowing a significant reduction of the error rate during the classification of attacks and a different way to tackle SQL injection attacks using a distributed and hierarchical approach.

The empirical results show that the best methods are those that involve the use of neural network and SVM and, if it is considered a mixture, the predictions capabilities are remarkably improved. Besides, the revision phase through a neural visualization eases the classification of suspicious queries.

Acknowledgments. This research is partially supported through the Junta de Castilla and León project BU006A08, Business intelligence for production within the framework of the Instituto Tecnológico de Castilla y León (ITCL) and the Agencia de Desarrollo Empresarial (ADE), projects of the Spanish Ministry of Science and Innovation TIN 2009-13839-C03-03**,** CIT-020000-2009-12 (funded by the European Regional Development Fund) and TIN2010-21272-C02-01 (funded by the European Regional Development Fund). The authors would also like to thank the vehicle interior manufacturer, Grupo Antolin Ingenieria S.A., within the framework of the project MAGNO2008 - 1028.- CENIT Project funded by the Spanish Government**.**

References

1. Halfond, W., Orso, A.: 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 (2005)
2. Valeur, F., Mutz, D., Vigna, G.: 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 (2005)
3. Rietta, F.: Application layer intrusion detection for SQL injection. In: 44th annual Southeast regional conference, pp. 531-536. ACM, New York (2006)
4. Woolridge, M., Wooldridge, M. J.: Introduction to Multiagent Systems, John Wiley & Sons, Inc., New York (2002)
5. Laza, R., Pavon, R., Corchado, J. M.: A Reasoning Model for CBR_BDI Agents Using an Adaptable Fuzzy Inference System. In: 10th Conference of the Spanish Association for Artificial Intelligence, Vol 3040, pp. 96-106. Springer (2003)
6. Anley, C.: Advanced SQL Injection In SQL Server Applications, http://www.ngssoftware.com/papers/more_advanced_sql_injection.pdf (2002)
7. Christensen, A. S.; Moller, A., Schwartzbach, M. I.: Precise Analysis of String Expressions. In: 10th International Static Analysis Symposium, pp. 1-18. Springer-Verlag (2003)
8. Su, Z., Wassermann, G.: The essence of command injection attacks in web applications. In: 33rd Annual Symposium on Principles of Programming Languages, pp. 372-382. ACM Press, New York (2006).
9. Huang, Y.; Huang, S.; Lin, T., Tsai, C.: Web application security assessment by fault injection and behavior monitoring. In: 12th international conference on World Wide Web, pp. 148-159. ACM (2003)
10. Ramasubramanian, P., Kannan, A.: 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 (2004)
11. Corchado, J. M., Bajo, J., Abraham, A.: GerAmi: Improving Healthcare Delivery in Geriatric Residences, Vol. 23, pp. 19-25. Intelligent Systems, IEEE (2008)
12. Vapnik VN (1999) An overview of statistical learning theory. IEEE Transactions on Neural Networks. 10: 988–999.
13. Corchado, J.M., Laza, R. (2003) Constructing deliberative agents with case-based reasoning technology. International Journal of Intelligent Systems 18: 1227-1241
14. Mukkamala, S., Sung, A.H., Abraham, A. (2005) Intrusion detection using an ensemble of intelligent paradigms. Journal of Network and Computer Applications 28(2): 167-182
15. Fritzke, B. (1995) A Growing Neural Gas Network Learns Topologies. (ed) Advances in Neural Information Processing Systems 7. MIT Press.
16. Herrero, Á., Corchado, E., Sáiz, L., Abraham, A. (2010) DIPKIP: A Connectionist Knowledge Management System to Identify Knowledge Deficits in Practical Cases. Computational Intelligence 26(1): 26-56