CBRid4SQL: A CBR Intrusion Detector for SQL Injection Attacks

Cristian Pinzón[1,2], Álvaro Herrero[3], Juan F. De Paz[1], Emilio Corchado[1], and Javier Bajo[1]

[1] Departamento de Informática y Automática, Universidad de Salamanca, Plaza de la Merced s/n, 37008, Salamanca, Spain
{cristian_ivanp,fcofds,escorchado,jbajope}@usal.es
[2] Universidad Tecnológica de Panamá, A.P: 0819-07289, Panamá, Rep. De Panamá
[3] Department of Civil Engineering, University of Burgos, Spain
C/ Francisco de Vitoria s/n, 09006 Burgos, Spain
ahcosio@ubu.es

Abstract. One of the most serious security threats to recently deployed databases has been the SQL Injection attack. This paper presents an agent specialised in the detection of SQL injection attacks. The agent incorporates a Case-Based Reasoning engine which is equipped with a learning and adaptation capacity for the classification of malicious codes. The agent also incorporates advanced algorithms in the reasoning cycle stages. The reuse phase uses an innovative classification model based on a mixture of a neuronal network together with a Support Vector Machine in order to classify the received SQL queries in the most reliable way. Finally, a visualisation neural technique is incorporated, which notably eases the revision stage carried out by human experts in the case of suspicious queries. The Classifier Agent was tested in a real-traffic case study and its experimental results, which validate the performance of the proposed approach, are presented here.

Keywords: SQL Injection, Intrusion Detection, CBR, SVM, Neural Networks.

1 Introduction

Over recent years, one of the most serious security threats around databases has been the SQL Injection attack [1]. In spite of it being a well-known type of attack, the SQL injection remains at the top of the published threat list. The solutions proposed so far [2], [3], [4], [5], [6], [7], [8] seem insufficient to prevent and block this type of attack because these solutions lack the learning and adaptation capabilities for dealing with attacks and their possible variations in the future. In addition, the vast majority of solutions are based on centralized mechanisms with little capacity to work in distributed and dynamic environments.

This study presents the intelligent agent CBRid4SQL (a CBR Intrusion Detector), capable of detecting attacks based on SQL code injection. CBRid4SQL is an agent specially designed following the strategy of an Intrusion Detection System (IDS) and is defined as a Hybrid Artificial Intelligence System (HAIS). This agent is the principal component of a distributed hierarchical multi-agent system aimed at detecting attacks in dynamic and distributed environments.

The CBRid4SQL agent is a CBR agent [9] that is characterized by the integration of a CBR (Case-Based Reasoning) mechanism. This mechanism provides the agents with a greater level of adaptation and learning capability, since CBR systems make use of past experiences to solve new problems [9]. This is very effective for blocking SQL injection attacks as the mechanism uses a strategy based on anomaly detection [10].

Additional to the incorporated CBR motor in the CBRid4SQL agent's internal structure, an integrated mixture through an Artificial Neural Network (ANN) and a Support Vector Machine (SVM) are used as a mechanism of classification. Through the use of this mixture, it is possible to exploit the advantages of both strategies in order to classify the SQL queries in a more reliable way.

Finally, to assist the expert in the making of decisions regarding those queries classified as suspicious, a visualization mechanism is proposed which combines clustering techniques and neural models to reduce the dimensionality based on unsupervised learning. The rest of the paper is structured as follows: section 2 presents the problem that has prompted most of this research work. Section 3 explains the internal structure of the CBRid4SQL agent used as a classifier agent. Finally, the conclusions and experimental results of this work are presented in section 4.

2 SQL Injection Attacks

An SQL injection 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 which is executed at the database layer of an application [1]. Different attack techniques exist which include the use of SQL Tautologies, Logic errors / Illegal queries, union query and piggy-backed queries. Other more advanced techniques use injection based on interference and alternative codification [1]. The cause of the SQL injection attacks is relatively simple: an inadequate input validation on the user interface. As a result of this attack, a hacker can be responsible for unauthorized data handling, retrieval of confidential information, and in the worst possible case, taking over control of the application server [1].

Different strategies have been presented as a solution to the problem of SQL injection attacks [1], with special attention given to strategies based on IDSs [2], [3], [4], [5], [6], [7], [8]. One approach based on anomaly detection was proposed by [2], applying a clustering strategy to group similar queries and isolate queries which are considered malicious. The main disadvantage of this approach is in its high computational overhead which would affect a real-time detection. Kemalis and Tzouramanis propose SQL-IDS (SQL Injection Detection System) [3] that uses security specifications to capture the syntactic structure of the SQL queries generated by the applications. The main limitation of this approach is the computational cost while comparing the new query with the predefined structure at runtime.

In [4] two types of SQL injection attacks are raised: tautology attacks and those based on the UNION operator. Through the syntactic analysis of SQL query strings, the data of the HTTP requests are extracted to later be used in the training phase and to determine the threshold to use in the evaluation phase. Bertino, Kamra and Early [4] propose an anomaly detection mechanism applying data mining techniques. The main problem of this approach is to find an adequate threshold to maintain a low rate of both false positives and false negatives. Another anomaly-based approach is proposed by Robertson, Vigna, Kruegel and Kemmerer [6]. The approach uses generalisation techniques to convert suspicious requests within abnormal signatures. These signatures are later used to group malicious requests which present similar characteristics. Another of the techniques used is characterization; deducing the type of attack associated with the malicious request. A low computational overhead is generated. However, it is susceptible to generating false positives. The algorithm ID3, presented by Garcia, Monroy and Quintana [7], proposes the detection of attacks targeted at web applications. The algorithm ID3 is used to detect and filter malicious SQL string. This approach presents a significant percentage of incorrect classifications. Valeur, Mutz, and Vigna [8] propose the use of anomaly detection through the generation of a series of models beginning with a set of recovered queries. At execution time, they monitor the applications in order to identify requests which are not associated with the aforementioned models.

3 An Agent for Detecting SQL Injection Attacks

Agents are characterized by their autonomy, which gives them the ability to work independently and in real-time environments [11]. The CBRid4SQL agent presented in this study interacts with other agents within the architecture. These agents carry out tasks related to capturing messages, syntactic analysis, administration, and user interaction. As opposed to the tasks for these agents, the CBRid4SQL agent executes classification SQL queries that we will subsequently define in greater detail.

CBR is a paradigm which is based on the idea that similar problems have similar solutions. Thus, a new problem is resolved by consulting the case memory to find a similar case which has been resolved in the past.

When working with this type of system, 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.

As previously mentioned, the CBRid4SQL agent is an specialization of a CBR agent which is the key component of a multi-agent architecture and is geared towards classifying SQL queries for the detection of SQL injection attacks. Below, a new classification mechanism incorporated in the internal structure of the CBRid4SQL agent is explained in detail.

3.1 CBRid4SQL 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 elements of the SQL Query described as follows: (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.

The fields defining a case are as follows: *IdCase, Session, User, IP_Address, Query_SQL, Affected_table, Affected_field, Command_type, Word_GroupBy, Word_Having, Word_OrderBy, Numer_And, Numer_Or, Number_literals, Number_LOL, Length_SQL_String, Start_Time_Execution, End_Time_Execution,* and *Query_Category*. Additionally, the information related to the prediction models used is stored as well.

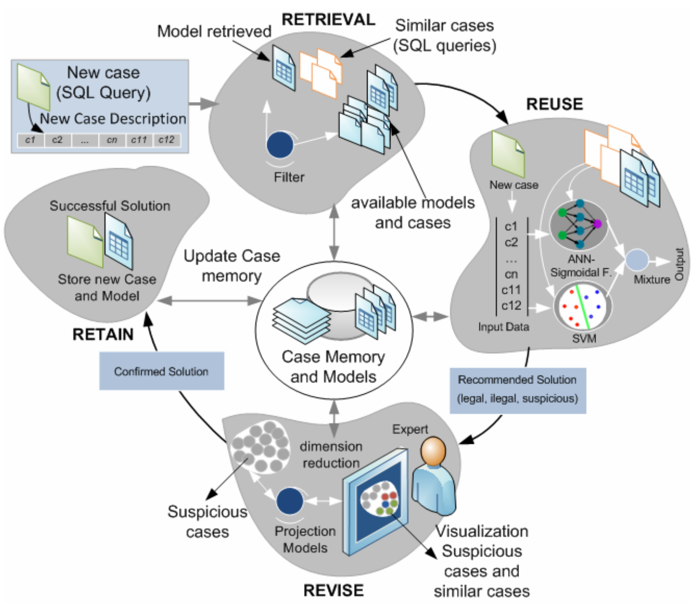

Fig. 1. CBR cycle and classification mechanism of the CBRid4SQL agent

In Figure 1, the different stages applied in the reasoning cycle can be seen. In summary, in the retrieval stage, there is a selection of queries sorted by type and by the memory's classification models. In the reuse phase, as seen in figure 1, 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 will be carried out manually by a human expert. At this stage the most similar cases will be selected by means of a Growing Cell Structures (GCS) network [12], visualized by a dimensionality reduction technique which employs the neuronal model called Cooperative Maximum Likelihood Hebbian Learning (CMLHL). As a result, 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. Below, the different stages of the CBR reasoning cycle associated with the system are described in more detail.

3.1.1 Retrieve

The retrieval phase is broken down into two phases; case retrieval and model retrieval. Case retrieval is performed by using the *Query_Category* attribute which retrieves queries from the case memory (C_r) which were used for a similar query in accordance with attributes of the new case c_n. Subsequently, the models for the multilayer perceptron mlp_r and svm_r 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 models will be considerably reduced, mainly in the case of the ANN training.

3.1.2 Reuse

The reuse phase is performed beginning with the information of the retrieved cases C_r and the recovered models mlp_r and svm_r. The combination of both techniques is fundamental in the reduction of the rate of false negatives. The inputs of the MLP are: *Query_SQL, Affected_table, Affected_field, Command_type, Word_GroupBy, Word_Having, Word_OrderBy, Numer_And, Numer_Or, Number_literals, Number_LOL,* and *Length_SQL_String*. The number of neurons in the hidden layer is *2n+1*, where *n* is the number of neurons in the input layer. Finally, at output, one neuron is had. The activation function selected for the different layers has been the sigmoid. Taking into account the activation function f_j, the calculation of output values are given by the following expression

$$y_j^p = f_j\left(\sum_{i=1}^{N} w_{ji}(t)\, x_i^p(t) \; + \theta_j\right) \tag{1}$$

The outputs correspond to x^r. 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]. This transformation is necessary because the network does not deal with values that fall outside of this range. The outgoing values are similarly limited to the range of [0.2, 0.8] with the value 0.2 corresponding to a non-attack and the value 0.8 corresponding to an attack. The network training is carried out through the error Backpropagation Algorithm [13].

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 models [14]. SVM also allows the separation of element classes which are not linearly separable. For this the space of initial coordinates is mapped in a high dimensionality space through the use of functions. Due to the fact that the dimensionality of the new space can be very high, it is not feasible to calculate hyperplanes that allow the production of linear separability. For this, a series of non-linear functions called kernels is used.

Let us consider a set of patterns $T = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$ where x_i is a vector of the dimension *n*. The idea is to convert the elements x_i in a space of high dimensionality through the application of a function, in such a way that the set of original patterns is converted into the following set $\Phi(T) = \{(\Phi(x_1), y_1), (\Phi(x_2), y_2), ..., (\Phi(x_m), y_m)\}$ that, depending on the selected function $\Phi(x)$, could be linearly separable. To carry out the classification, this equation sign is studied [15]:

$$class(x_k) = sign\left(\sum_{i=1}^{m} \lambda_i y_i \Phi(x_i)\Phi(x_k) + b\right) \tag{2}$$

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.

3.1.3 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 queries similar to the current one together with previous classifications. This combines a clustering technique for the selection of similar requests with a neuronal model for the reduction of dimensionality, which permits 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 in which the new case is found 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. Considering an N-dimensional input vector (x), and an M-dimensional output vector (y), with W_{ij} being the weight (linking input j to output i), then CMLHL can be expressed as:

Feed-forward step:
$$y_i = \sum^{\mathbf{N}} W_{ij} x_j, \forall i \tag{3}$$

Lateral activation passing:
$$y_i(t+1) = [y_i(t) + \tau(b - Ay)]^+ \tag{4}$$

Feedback step: $$e_j = x_j - \sum_{i=1}^{M} W_{ij} y_i , \forall j \quad \circ \tag{5}$$

Weight change: $$\Delta W_{ij} = \eta . y_i . sign\left(e_j\right) | \, e_j \, |^{p-1} \tag{6}$$

Where: η is the learning rate, τ is the "strength" of the lateral connections, b the bias parameter, p a parameter related to the energy function [14], [15] and A is a symmetric matrix used to modify the response to the data [14]. The effect of this matrix is based on the relation between the distances separating the output neurons.

Finally, the information is represented and the associated queries are recovered with the retrieved mesh, as can be seen in Fig. 2.

3.1.4 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 Experimental Results and Conclusions

A sample web application with access to a MySQL 5.0 database was developed to check the proposed approach. Once the database had been created, legal queries were sent from the designed user interfaces. In the case of malicious queries, the dispatch of the queries was automated using the agent SQLMAP0.5 [17]. This tool is able to fingerprint an extensive DBMS back-end, retrieve remote DBMS databases and so on.

To analyze the successful rates, a test of the classification of queries was conducted, taking into account the following classifiers: Bayesian Network, Naive Bayes, AdaBoost M1, Bagging, DecisionStump, J48, JRIP, LMT, Logistic, LogitBoost, MultiBoosting AdaBoost, OneR, SMO, Stacking. The different classifiers were applied to 705 previously classified queries (437 legal, 268 attacks). The consecutive process to carry out the output test was the following: selecting one of the cases, extracting it from the set, conducting the model starting from the remaining cases and classifying the extracted case. This process is repeated for each one of the cases and techniques in order to analyze each query without it being used to build the model. The final result of the classification can be seen in Table 1.

Table 1. Total number of hits for the different classifiers

Method		Method		Method	
BayesNet	638	Naive Bayes	666	AdaBoostM1	665
Bagging	684	DecisionStump	598	J48	689
JRIP	692	LMT	693	Logistic	688
LogitBoost	680	MultiBoostAB	666	OneR	622
SMO	685	Stacking	437	CBRid4SQL	**698**

As can be seen in Table 1, the higest-performance system is CBRid4SQL, which has a success rate of 698/705. The number of queries detected as suspicious was limited to 7 being one of those shown below:

select pedido_cliente.id_pedido, linea, codigo, nombre, precio from pedido_lineas, pedido_cliente, producto where pedido_cliente.id_pedido = pedido_lineas.id_pedido and producto.codigo = pedido_lineas.codigo and pedido_cliente.id_pedido = 1 OR 1 = 1 order by fecha desc

This query represents an attack on the database since the presence of OR 1=1implies the retrieval of a number of records not associated with requests from the client.

The value obtained by the ANN for this query was 0.28. However SVM deemed that the output value was 0.66. The mixture gave an output value of 0.47, which is in the range of suspicious queries. If the ANN had been applied alone it would have considered this query as a valid one. However, the SVM would have considered it as an attack. The mixture deemed suspicious and a review would be carried out manually.

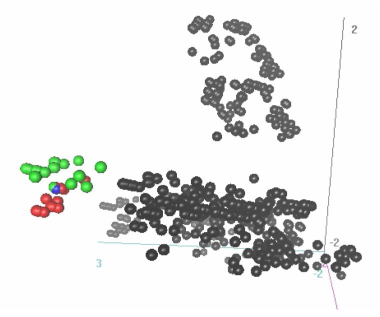

Fig. 2. SQL queries recovered in the revise stage

During the manual review similar queries are recovered and dimensionality is reduced. In Figure 2 the obtained results to be shown to the human expert can be seen. The most similar queries are coloured: queries that correspond as legal are shown in green, attacks are in red and current queries are in blue. Non-recovered queries are shown in black. A set of queries different from the rest is recovered, both normal and abnormal. An example of each of these is shown below.

select pedido_cliente.id_pedido, linea, codigo, nombre, precio from pedido_lineas, pedido_cliente, producto where pedido_cliente.id_pedido = pedido_lineas.id_pedido and producto.codigo = pedido_lineas.codigo and pedido_cliente.id_pedido = 1 AND ORD(MID((CONCAT(CHAR(55), CHAR(55))), 1, 1)) > 63 order by fecha desc

select pedido_cliente.id_pedido, linea, codigo, nombre, precio from pedido_lineas, pedido_cliente, producto where pedido_cliente.id_pedido = pedido_lineas.id_pedido and producto.codigo = pedido_lineas.codigo and pedido_cliente.id_pedido = 1 AND 1 = 1 order by fecha desc

The first of the queries is a clear attack, while second of the queries could also present uncertainties due to the presence of the literal *1=1*. Being that a query more restrictive than the original, it would retrieve the same values or less, which would not be a very intelligent for an attack. In any case, the system considers it as such and provides an output value of 0.66 and it is thus also filtered a priori, but this should not be worrying. This is one of the false positives that the system presents within the 7 existing in the 705.

The combination of different paradigms of AI allows the development of a HAIS with characteristics such as the capacity for learning and reasoning, flexibility and robustness which make the detection of SQL injection attacks possible. The proposed CBRid4SQL agent is capable of detecting these abnormal situations with low error rates compared with other existing techniques, as demonstrated in Table 1. It also provides a decision mechanism which eases the review of suspicious queries through the selection of similar queries and their visualization using neuronal models.

Acknowledgments. This development has been partially supported by the Spanish Ministry of Science and Technology project OVAMAH: TIN 2009-13839-C03-03, Junta of Castilla and León (JCyL): [BU006A08], the project of the Spanish Ministry of Education and Innovation [CIT-020000-2008-2] and [CIT-020000-2009-12], and Grupo Antolin Ingenieria, S.A., within the framework of project MAGNO2008 - 1028.- CENIT also funded by the same Government Ministry and The Professional Excellence Program 2006-2010 IFARHU-SENACYT-Panama.

References

1. Halfond, W.G.J., Viegas, J., Orso, A.: A Classification of SQL-Injection Attacks and Countermeasures. In: Proceedings of the IEEE International Symposium on Secure Software Engineering, Arlington, VA, USA (2006)
2. Bockermann, C., Apel, M., Meier, M.: Learning SQL for Database Intrusion Detection Using Context-Sensitive Modelling (Extended Abstract). In: Flegel, U., Bruschi, D. (eds.) DIMVA 2009. LNCS, vol. 5587, pp. 196–205. Springer, Heidelberg (2009)
3. Kemalis, K., Tzouramanis, T.: SQL-IDS: a specification-based approach for SQL-injection detection. In: Proceedings of the 2008 ACM symposium on Applied computing (SAC 2008). ACM, New York (2008)
4. Kiani, M., Clark, A., Mohay, G.: Evaluation of Anomaly Based Character Distribution Models in the Detection of SQL Injection Attacks. In: Third International Conference on Availability, Reliability and Security (ARES 2008). IEEE Computer Society, Washington (2008)
5. Bertino, E., Kamra, A., Early, J.: Profiling Database Applications to Detect SQL Injection Attacks. In: Proceedings of the Performance, Computing, and Communications Conference, IPCCC 2007 (2007)
6. Robertson, W., Vigna, G., Kruegel, C., Kemmerer, R.A.: Using Generalization and Characterization Techniques in the Anomaly-Based Detection of Web Attacks. In: 13th Annual Network and Distributed System Security Symposium, NDSS 2006 (2006)
7. García, V.H., Monroy, R., Quintana, M.: Web Attack Detection Using ID3. In: International Federation for Information Processing (2006)
8. 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)

9. Corchado, J.M., Laza, R.: Constructing deliberative agents with case-based reasoning technology. International Journal of Intelligent Systems 18, 1227–1241 (2003)
10. Mukkamala, S., Sung, A.H., Abraham, A.: Intrusion detection using an ensemble of intelligent paradigms. Journal of Network and Computer Applications 28(2), 167–182 (2005)
11. Carrascosa, C., Bajo, J., Julian, V., Corchado, J.M., Botti, V.: Hybrid multi-agent architecture as a real-time problem-solving model. Expert Systems with Applications 34(1), 2–17 (2008)
12. Fritzke, B.: A Growing Neural Gas Network Learns Topologies. In: Advances in Neural Information Processing Systems, vol. 7. MIT Press, Cambridge (1995)
13. LeCun, Y., Bottou, L., Orr, G.B., Müller, K.R.: Efficient BackProp. In: Orr, G.B., Müller, K.-R. (eds.) NIPS-WS 1996. LNCS, vol. 1524, p. 9. Springer, Heidelberg (1998)
14. Corchado, E., Fyfe, C.: Connectionist Techniques for the Identification and Suppression of Interfering Underlying Factors. International Journal of Pattern Recognition and Artificial Intelligence 17(8), 1447–1466 (2003)
15. Corchado, E., MacDonald, D., Fyfe, C.: Maximum and Minimum Likelihood Hebbian Learning for Exploratory Projection Pursuit. Data Mining and Knowledge Discovery 8(3), 203–225 (2004)
16. Herrero, Á., Corchado, E., Sáiz, L., Abraham, A.: DIPKIP: A Connectionist Knowledge Management System to Identify Knowledge Deficits in Practical Cases. Computational Intelligence 26(1), 26–56 (2010)
17. Damele, B.: SQLMAP0.5 – Automated SQL Injection Tool (2007)