idMAS-SQL: Intrusion Detection Based on MAS to Detect and Block SQL Injection through Data Mining

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

[1]Faculty of Computer Systems Engineering, Technological University of Panama
Building Nº3, Campus «Dr. Víctor Levi Sasso»,
Universidad Tecnológica Ave. Panamá City, Panamá
cristian_ivanp@usal.es

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

[3]Department of Civil Engineering, University of Burgos,
C/ Francisco de Vitoria s/n, 09006, Burgos, Spain
ahcosio@ubu.es

Corresponding Author:

Name: Javier Bajo
Tlfn: +34 639771985
Fax: +34 923277101
Email: jbajope@upsa.es
Address: Compañía 5, 37002, Salamanca, Spain

Abstract.

This study presents a multiagent architecture aimed at detecting SQL injection attacks, which are one of the most prevalent threats for modern databases. The proposed architecture is based on a hierarchical and distributed strategy where the functionalities are structured on layers. SQL-injection attacks, one of the most dangerous attacks to online databases, are the focus of this research. The agents in each one of the layers are specialized in specific tasks, such as data gathering, data classification, and visualization. This study presents two key agents under a hybrid architecture: a classifier agent that incorporates a Case-Based Reasoning engine employing advanced algorithms in the reasoning cycle stages, and a visualizer agent that integrates several techniques to facilitate the visual analysis of suspicious queries. The former incorporates a new classification model based on a mixture of a neural network and a Support Vector Machine in order to classify SQL queries in a reliable way. The latter combines clustering and neural projection techniques to support the visual analysis and identification of target attacks. The proposed approach was tested in a real-traffic case study and its *experimental results, which validate the performance of the proposed approach, are presented in this paper.*

Keywords: Intrusion Detection, SQL Injection Attacks, Data Mining, CBR, SVM, Neural Networks, Multiagent System

1. Introduction

Currently, one of the most dangerous and common threats to databases and Web applications is the SQL injection attack. It typically involves malicious modifications of the user SQL input either by adding additional clauses or by changing the structure of an existing clause [46]. SQL injection enables attackers to access, modify, or delete critical information in a database without proper authorization [45]. In spite of being a well-known type of attack, the SQL injection remains at the top of the published list of security threats [35]. The solutions proposed so far [6], [47], [48], [5], [60], [32], [64] seem insufficient to prevent and block this type of attack because these solutions lack the learning and adaptation capabilities for dealing with 0-day (previously unseen) attacks as well as new or future variations of attacks. Furthermore, the vast majority of these solutions are based on centralized mechanisms, with little capacity to work in distributed and dynamic environments.

Taking previous research one step further, this study presents idMAS-SQL (Intrusion Detection Based on MAS to Detect and Block SQL Injection), a hybrid solution based on a distributed architecture (multiagent system - MAS) [65] capable of detecting and blocking SQL Injection Attacks. The philosophy of multi-agent systems makes it possible to deal with SQL injection attacks from the perspective of the elements of communication, ubiquity and autonomous computation, and from the standpoint of a global distributed system. Every component in idMAS-SQL interacts and cooperates to achieve a global common goal: the detection and prevention of ongoing intrusions in a database. idMAS-SQL presents a hierarchical organization structured by layers of agents, which distributes roles and tasks for the detection and prevention of SQL injection attacks. The agents at each level are assigned specific tasks which they can execute regardless of their physical location, due to their own capabilities.

idMAS-SQL has evolved from the SC-MAS architecture [2] which follows the strategy of an Intrusion Detection System (IDS) by means of a distributed approach based on the idCBR agent capabilities. idCBR agents are a particular type of CBR-BDI agent [3], [16] whose internal structure and capacities are based on mental aptitude [33]. These agents are characterized by the integration of a CBR (Case-Based Reasoning) mechanism [3] in a deliberative BDI Agent. This mechanism provides the agents with a greater level of adaptation and learning capability given that CBR systems use past experiences to solve new problems [16]. This is very effective for blocking SQL injection attacks as the mechanism uses a strategy based on anomaly detection [55], modelling the normal/legal SQL queries.

The main innovations of idMAS-SQL are the incorporation of a new classification strategy which is based on Data Mining in idCBR agents, and of an agent with special capabilities for the visualization and subsequent analysis of data. idCBR agents are specially designed to incorporate a mixture of classification through an Artificial Neural Network (ANN) and a Support Vector Machine (SVM). Through the use of this mixture, it is possible to take advantage of both strategies in order to classify SQL queries in a more reliable way. The use of idCBR agents with advanced capabilities for analyzing and predicting SQL attacks is one of the main features of the architecture. Furthermore, the incorporation of a visualizer agent provides human experts with a very useful tool for analyzing those cases which are classified as suspicious by the idCBR agent and which require validation by an expert. The Visualizer Agent is a type of agent equipped with advanced capabilities for data visualization through unsupervised projection models. Specifically, it combines a clustering technique for the selection of similar requests with a neural model for the reduction of dimensionality, which permits visualisation in 2D or 3D.

The remainder of the paper is structured as follows: Section 2 presents the problem that has prompted most of this research work. Section 3 introduces the topic of visualization techniques for information security and the projection models applied in this study. Section 4 presents the proposed MAS architecture in detail. Section 5 explains the internal structure of the two most important agents in this architecture. Finally, Sections 6 and 7 present the experimental results and conclusions after having tested the proposed approach.

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, executed at the database layer of an application [35]. Different attack techniques exist which include the use of SQL Tautologies, Logic errors/Illegal Queries, Union Queries and Piggy-back Queries. Other more advanced techniques use injections based on interference and alternative codification [35].

(1)	`SELECT * FROM tblUsers WHERE id = 1 or 1=1 AND user LIKE "%root%"`
(2)	`SELECT IF( user = 'root', BENCHMARK(1000000,MD5( 'x' )),NULL) FROM login declare @q varchar(8000); select @q = 0x73656c65637420404076657273696f6e; exec(@q).` Generating results: *‘select @@version'*

The first query bases its strategy on adding an expression that is always true to the where-clause of a select statement (tautologies). The second query masks the injection by using a type of codification such as ASCII (American Standard Code for Information Interchange) or a codification in Hexadecimal format.

The cause of 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 [35].

Different strategies have been presented as a solution to the problem of SQL injection attacks [35], with special attention given to strategies based on IDSs [6], [47], [48], [5], [60], [32], [64]. One approach based on anomaly detection was proposed by [6], applying a clustering strategy to group similar queries and isolate queries which are considered malicious. The main disadvantage of this approach is its high computational overhead which would affect a real-time detection. Kemalis and Tzouramanis propose the SQL-IDS (SQL Injection Detection System) [47] which uses security specifications to capture the syntactic structure of SQL queries generated by the applications. The main limitation of this approach is the computational cost when comparing the new query with the predefined structure at runtime.

In [48] 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 HTTP requests are extracted to be used later in the training phase and to determine the threshold to be used in the evaluation phase. Bertino, Kamra and Early [48] propose an anomaly detection mechanism applied using data mining techniques. The main problem of this approach is finding 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 [60]. This 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 technique used is characterization, which involves deducing the type of attack associated with a malicious request. A low computational overhead is generated;

however, it is susceptible to generating false positives. The ID3 algorithm, presented by Garcia, Monroy and Quintana [32], proposes the detection of attacks targeted at web applications. The ID3 algorithm is used to detect and filter malicious SQL strings. This approach presents a significant percentage of incorrect classifications. Valeur, Mutz, and Vigna [64] 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. Visualization for Information Security

As they are considered a viable approach to information seeking, visualisation techniques have been applied to massive datasets for many years [1]. Visual inspection of network traffic patterns is presented as an alternative for managing a crucial aspect of network monitoring [4] because its chief aim is to provide security personnel with a synthetic representation of the network situation. In performing this task, visualisation tools can:

- Assist security personnel in detecting anomalies and potential threats through an intuitive display of the progression of network traffic.
- Deal easily with highly heterogeneous and noisy data such as the data required for network monitoring and intrusion detection (ID) [18].
- Provide network managers with automated support and motivate their effectiveness by taking advantage of the ability of the human eye to extrapolate normal traffic patterns and detect anomalies. As stated in [8], "*a picture is worth a thousand packets*" or "*a picture is worth a thousand log entries*" [54].
- Help network managers diagnose performance issues or understand communication patterns between nodes.
- Serve as tools that are complementary to other security mechanisms.

The monitoring task that detects intrusive or anomalous events can be achieved by visualising data at different levels of abstraction: network nodes, intrusion alerts, packet-level data, communication content, log files, and so on. In other words, different data from various security tools can be visualised.

Visualisation tools rely on the human ability to recognize different features and to detect anomalies through graphical devices [1]. Human vision can rapidly locate, discover, identify, and compare objects; all essential tasks in the network monitoring process, considering the overwhelming amount of information and raw traffic data that must be processed [8].

The underlying operational assumption of this approach is mainly grounded in the ability to render high-dimensional traffic data in a consistent yet low-dimensional representation. Therefore, security visualisation tools have to map high-dimensional feature data into a low-dimensional space for presentation. One of the main assumptions of the visualization task in the present study is that neural projection models will prove themselves to be satisfactory for the purpose of SQL query visualisation through dimensionality reduction, as they previously have been for some other attacks [12, 38].

To date, most researches on ID have approached it from a classification standpoint, such as [26, 50, 59, 63]. They perform a 2-class classification of network traffic: normal/anomalous in anomaly-based ID, and intrusive/non-intrusive in misuse-based ID. From a different standpoint, this study proposes visualisation techniques for the detection of attacks. It is worth emphasizing that this proposal entails the visualisation of SQL queries for detecting attacks (that is, visualisation for ID) and not the visualisation of IDS alerts or logs (that is, visualisation of ID) as others have done [44].

The visualisation-based approach to ID relies on the following ideas [9]:

- Anomalous situations can be identified by their "visual signature". Visual fingerprints are frequently visible despite the visual noise of background traffic.
- Some stealthy attacks are resistant to detection by classification-based IDSs, but are readily visible using appropriate visualisations.
- Visualisation techniques require little resources and are remarkably resistant to overload caused by high volumes of network traffic.
- The completeness of visualisation-based IDSs is supposedly higher than that of classification-based IDSs when facing 0-day attacks.

Unlike other security tools, IDSs need to be monitored to make the most of their benefits [19]. The huge number of alerts that are usually generated by IDSs (including many false positives and negatives) is a hindrance to permanent (24h.) monitoring, mainly due to economical costs. Visualisation-based IDSs can ease this task by providing an easily understandable snapshot of the network status, thus reducing the time needed for ID.

Visualisation techniques take advantage of the outstanding capabilities of the human visual system to detect patterns and anomalies in visual representations of abstract data [52]. Another advantage of visualisation is that it transforms the task of analysing network data from a perceptually serial process (by reading textual data) to a perceptually parallel process (by presenting more concepts) [25]. Because of this, the visualisation approach to ID implies several advantages:

- Attack visualisation can provide fresh insight into the analysed data, allowing the deduction of new hypotheses usually lost in complex analysis [18]. Consequently, 0-day attacks can be easily detected.
- "*Visualisation tools need to be designed so that anomalies can be easily flagged for later analysis by more experienced analysts*" [34]. Visualisation for ID can help in training security personnel with no previous experience in security tasks, as well as reducing the time spent by more experienced personnel.
- For effective analysis, network data must be correlated with several variables. This requires dealing with highly heterogeneous, complex, and noisy data. The visualisation approach simplifies this problem by presenting the traffic situation in an intuitive way, as visual images can give perceptual clues to the administrators [18].
- Attack visualisation can be much faster than other anomaly detection approaches [18]. As it also reduces the time and effort of reviewing security logs, it implies a great reduction of the time (and hence resources) required for ID.

Although many ID tools have begun to incorporate advanced graphical user interfaces, most of them fail to provide an intuitive and comprehensive visualisation of network traffic. To identify intrusions, one has to look for "interesting" structures and for "abnormal" or unusual data. Although this process cannot be precisely detailed in a general and objective way, "*one usually can recognize unusual data when one sees them*" [53].

One of the main drawbacks of the visualisation approach to ID is that even if equipped with the "perfect" visualisation technique, security personnel will make mistakes in detecting intrusions. This is a consequence of relying on human abilities, which are affected by a range of factors such as time pressure, fatigue, boredom, and so on.

In this study, some statistical and unsupervised neural projection models were applied for visualization-based ID of SQL injection attacks. These models are described in the following subsections.

3.1 Principal Component Analysis

Principal Component Analysis (PCA) [41], [58] describes the variation in multivariate data in terms of a set of uncorrelated variables, in decreasing order of importance, each of which is a linear combination of the original variables. Using PCA it is possible to find a smaller group of underlying variables that describe the data, with the result that the first few components of such a group might explain most of the variation in the original data. It should be noted that even if we are able to characterize the data with a few variables, it does not mean that an interpretation will ensue. This statistical technique may be performed by using connectionist models [56], [61], [29].

3.2 Exploratory Projection Pursuit

Exploratory Projection Pursuit (EPP) [27] is a statistical technique for solving the complex problem of identifying structure in high-dimensional data. It involves low-dimensional data projections in which structure is identified by eye and requires an index of "interestingness" by which each projection is measured. Subsequently, the data is transformed by optimizing this index in order to examine the projections of greatest interest in greater detail. From a statistical point of view, the most interesting directions are those which are as non-Gaussian as possible. Typical random data set projections are usually Gaussian [23], so identification of the most interesting features in the data calls for further investigation of these "interesting" directions. As in the case of PCA, this statistical technique may be implemented by using connectionist models [43], [42], [13], [30].

While PCA is focused on the identification of the largest variance directions, EPP looks for higher order statistics, such as skewness or kurtosis.

3.3 Curvilinear Component Analysis

Curvilinear Component Analysis (CCA) [22] is a nonlinear dimensionality reduction method. It was developed as an improvement on the Self-Organizing Map (SOM) [49]. It tries to circumvent the limitations inherent in some previous linear models such as PCA. Its output is not a fixed lattice but a continuous space able to take the shape of the submanifold in the dataset (input space).

The CCA is based on a self-organised neural network performing two tasks: a vector quantization of the submanifold, and a nonlinear projection of these quantising vectors toward an output space, providing a revealing view of the way in which the submanifold unfolds. Quantization and nonlinear mapping are separately performed by two layers of connections.

In the vector quantization step, the input vectors are forced to become prototypes of the distribution by using competitive learning and the regularization method proposed in [21]. Thus, this step, which is intended to reveal the submanifold of the input data, regularly quantizes the space covered by the data, regardless of the density. Euclidean distances between these input vectors are considered, as the output layer has to build a nonlinear mapping of the input vectors.

Since a perfect matching is not possible at all scales when the manifold is "unfolding", a weighting function is introduced, yielding the quadratic cost function:

$$E = \frac{1}{2}\sum_{i}\sum_{j \neq i}\left(X_{ij} - Y_{ij}\right)^{2} F\left(Y_{ij}, \lambda_{y}\right) \qquad \textbf{(1)}$$

As regards its goal, the projection part of CCA is similar to other nonlinear mapping methods in that it minimizes a cost function based on interpoint distances in both input and output spaces. Instead of moving one of the output vectors (y_i) according to the sum of the influences of every other y_j, CCA proposes pinning down one of the output vectors (y_i) "temporarily", and moving all other y_j around, disregarding any interaction between them. Accordingly, the proposed "learning" rule can be expressed as:

$$\Delta y_j = \alpha(t) F\left(Y_{ij}, \lambda_y\right)\left(X_{ij} - Y_{ij}\right)\frac{y_j - y_i}{Y_{ij}} \quad \forall j \neq i \qquad \textbf{(2)}$$

The main advantages of CCA, in comparison to other methods such as stochastic gradient descent or steepest gradient descent, are:

- The proposed rule (Δy) is much lighter than a stochastic gradient from a computational standpoint.
- The average of the output vector updates is proportional to the opposite of the gradient of the cost function (E). On the other hand, it can temporarily produce increases in E, which eventually allows the algorithm to escape from local minima of E. The research in [22] showed that the CCA method implies a lower final cost in comparison with gradient methods.

CCA is able to perform dimensionality reduction and represent the intrinsic structure of given input data without any previous knowledge of distribution of the analysed dataset.

Compared with other previous projection algorithms, the CCA method is more general, reliable, and faster at capturing input data structure.

3.4 Cooperative Maximum Likelihood Hebbian Learning

Cooperative Maximum Likelihood Hebbian Learning (CMLHL) is based on the EPP neural model called Maximum Likelihood Hebbian Learning (MLHL) [13, 31]. The main difference between these two models is that CMLHL includes lateral connections [10, 11] derived from the Rectified Gaussian Distribution (RGD) [62]. The RGD is a modification of the standard Gaussian distribution in which the variables are constrained to be non-negative. More precisely, CMLHL includes lateral connections based on the mode of the cooperative distribution that is closely spaced along a nonlinear continuous manifold. By including these lateral connections, the resulting network can find the independent factors of a dataset in a way that captures some type of global ordering in the dataset.

Considering an N-dimensional input vector $\mathbf{x}$, an N-dimensional output vector $\mathbf{y}$ and with W_{ij} being the weight (linking input j^{th} to output i^{th}), CMLHL can be expressed as:

Feed-forward step:

$$y_i = \sum_{j=1}^{N} W_{ij} x_j , \forall i \qquad \mathbf{(3)}$$

Lateral activation passing:

$$y_i(t+1) = [y_i(t) + \tau(b - Ay)]^+ \qquad \mathbf{(4)}$$

where A (described below) is a matrix used to modify the response to the data, and b is the bias parameter

Feedback step:

$$e_j = x_j - \sum_{i=1}^{M} W_{ij} y_i , \forall j \qquad \mathbf{(5)}$$

Weight change:

$$\Delta W_{ij} = \eta . y_i .sign(e_j) | e_j |^p \qquad \mathbf{(6)}$$

where η is the learning rate, τ is the "strength" of the lateral connections, and p a parameter related to the energy function [10, 13, 31].

A is a symmetric matrix used to modify the response to the data whose effect is based on the relation between the distances among the output neurons. It is based on the cooperative distribution, but to speed learning up, it can be simplified to:

$$A(i,j) = \delta_{ij} - \cos\left(2\pi(i-j)/M\right) \qquad (7)$$

where δ_{ij} is the Kronecker delta.

The application of CMLHL, initially in the field of artificial vision [10, 11], and subsequently to other interesting topics [14, 37, 39, 40], has proven that this model can successfully perform data visualisation.

3.5 Multidimensional Scaling

Multidimensional Scaling (MDS) [17] is a dimensionality reduction technique used for representing data. It involves finding a graphic representation in low dimensionality which is as close as possible to the original data. Two types of MDS exist: working with original values, or using ranges to represent the order of values in place of distances. If original magnitudes are used, it is called metric MDS, otherwise, it is denominated non-metric MDS. Starting with a matrix of distances, the values are sorted, representing a connection that indicates which elements are closest. What MDS tries to create is a new set of variables which maintain the same order of the initial variables based on a new matrix of distances.

4. idMAS-SQL: A Multi-Agent Architecture to Detect and Block SQL Injection

Given some of the above mentioned capabilities, a multi-agent solution fits the challenge of detecting and blocking SQL-injection attacks. In keeping with this idea, the present study proposes the use of a multi-agent architecture, (see Fig. 1) to focus on SQL injection attacks. It is based on an innovative approach since there is no known architecture with these characteristics for detecting SQL injection attacks.

The distributed resolution of problems balances the workload, facilitates recovery from error conditions, and also avoids centralized traffic. The analysis, classification and decision making capabilities, among others, are distributed throughout several layers in the proposed idMAS-SQL architecture, as depicted in Fig. 1. The agents that make up the architecture are assigned specific roles to perform their tasks. Moreover, the distribution greatly simplifies the capacity to recover from errors or failures because if an agent fails, it is immediately replaced without affecting the other agents at the same level or in other levels. Additionally, the proposed architecture is based on a hierarchical model that reduces the complexity of tasks such as monitoring and capturing user requests, classifying user requests, evaluating the final solution, etc. Distributing the functionality at each level, while maintaining each level independently, allows new changes to be easily adapted. Each level of the architecture houses a collection of agents with well-defined roles that allow their tasks and responsibilities to be clearly specified. The architecture has been divided into three levels so that the specific tasks are assigned according to the degree of complexity. Fig 1 depicts the idMAS-SQL architecture with each level and the respective agents.

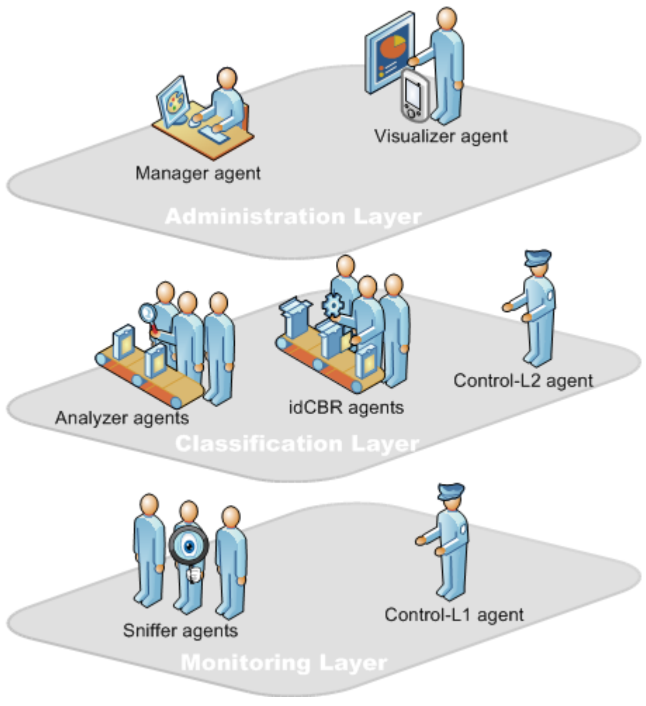


Fig. 1. idMAS-SQL architecture showing the different layers and their respective agents.

idMAS-SQL is presented as an evolution of the SC-MAS architecture [2] that proposed a novel strategy to identify and block SQL injection attacks through a distributed approach based on the capacities of idCBR agents, which are a particular type of CBR-BDI agents [15]. As well as the CBR-BDI agents, an agent with visualisation capabilities is incorporated to assist the expert in decision making regarding queries that are classified as suspicious. To do so, a visualization mechanism is proposed which combines clustering techniques and neural models, based on unsupervised learning, to reduce dimensionality.

The different types of agents located at the different levels of the idMAS-SQL architecture can be described as:

- **Sniffer**. This type of agent is located in the monitoring layer and is responsible for capturing datagrams, ordering TCP fragments to extract the request's SQL string and executing a syntactic analysis of the request's SQL string. There can be more than one Sniffer agent depending on the amount of workload.
- **Control-L1**. This is one of the agents that execute control and communication functions in the lower layers of the architecture. It is located in the monitoring layer, and all communication from this layer is administered by the agent. Its functions include: receiving data from the Sniffer agent and assigning the Analyzer agent to the task of searching for patterns of attacks; reporting to the administration layer the detection of any intrusion during the process of comparing attack signatures; and supervising the workload of the layer in order to request, from the administration layer, the creation or elimination of Sniffer or Analyzer agent requests.
- **Analyzer**. This type of agent is located in the classification layer. Its function includes matching patterns of known attacks; a database with previously built patterns allows this task. There can be more than one Analyzer agent depending on the amount of workload.

- **idCBR**. This type of agent is also located in the classification layer and is a core component of the architecture as it carries out a classification of SQL strings through detection anomalies. It integrates a case based reasoning (CBR) mechanism. In the **reuse phase** of the CBR cycle it applies a mixture of neural networks to generate a classification (legal, illegal or suspicious). There can be more than one idCBR agent depending on the amount of workload.
- **Control-L2**. This is the second type of agent for carrying out control and communication functions. All of the incoming and outgoing communication of the classification layer is administered by the Control-L2 agent. Its functions include: receiving processed data from the Monitoring layer and assigning a specific classifier agent to execute the task of classification; reporting the detection of intrusions to the administration layer once the classification process is completed; and supervising the workload of the layer to request, from the administration layer, the creation or elimination of requests from the idCBR agents.
- **Visualizer**. This agent is the main novelty of the architecture proposed in this research. Also located in the Administration layer, this agent facilitates the interaction between security personnel and the architecture. It applies different projection models for visualizing SQL-related data. Consequently, SQL injection attacks can be visually identified. In addition to this function, the Visualizer agent dictates the rules and actions for when an intrusion is detected, given that its principal task is to block (not execute) queries identified as anomalous. It also facilitates the implementation of adjustments in the architectural setup. Finally, it is equipped with the ability to run on mobile devices to facilitate the task of monitoring.

- **Manager**. This agent is responsible for the evaluation and coordination of the overall architectural operation. It administers the directory of the architecture's operative agents through communication with the idCBR agents at each layer.

The following section provides a detailed description of the two types of key agents in intrusion detection.

5. Agents for Detecting SQL Injection Attacks

The idCBR and Visualizer agents, presented in this study, interact with other agents within the idMAS-SQL architecture. These other agents carry out tasks related to message capturing, syntactic analysis, and administration. The idCBR and Visualyzer agents execute complementary tasks to determine the reliability of SQL queries.

The idCBR agent is a type of BDI agent that incorporates a CBR engine. This paradigm 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.

The Visualizer agent is an agent that is equipped with the capability of visualization, which helps the security expert to resolve any user requests that are classified as suspicious by the idCBR agent. This agent incorporates projection models used as tools to identify and remove correlations between problem variables, which enable us to carry out dimensionality reduction, and visualization or exploratory data analysis. In this study, some statistical and unsupervised neural projection models, specifically PCA [58], CCA [22], CMLHL [10], and MDS [17] were applied.

Below, the mechanisms incorporated in the internal structures of the idCBR and Visualizer agents are presented in detail.

5.1 idCBR Agent

This subsection presents the idCBR agent, with special emphasis on its internal structure and the classification mechanism of SQL attacks. This mechanism combines the advantages of CBR-BDI 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.

When working with CBR systems, the key concept is that of “case”. A case is defined as a previous experience and is composed of three elements: a description 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.

In terms of CBR, the case is composed of elements of the SQL query described as follows: (a) Problem Description, which describes the initial information available for generating a plan. The problem description consists of: case identification, user session and SQL query elements. (b) Solution, which describes the action carried out in order to solve the problem description, in this case, prediction models. (c) Final State, which describes the state achieved after the solution has been applied. The fields defining a case are listed in Table 1. Additionally, the information related to the prediction models used is also stored in the Models Memory.

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

Problem description fields		**Solution fields**	
IdCase	Integer	Idcase	Integer
Session	Session	Classification_Query	Integer
User	String		
IP_Address	String		
Query_SQL	Query_SQL		
Affected_table	Integer		
Affected_field	Integer		
Command_type	Integer		
Word_GroupBy	Boolean		
Word_Having	Boolean		
Word_OrderBy	Boolean		
Number_And	Integer		
Number_Or	Integer		
Number_literals	Integer		
Number_LOL	Integer		
Length_SQL_String	Integer		
Start_Time_Execution	Time		
End_Time_Execution	Time		
Query_Category	Integer		

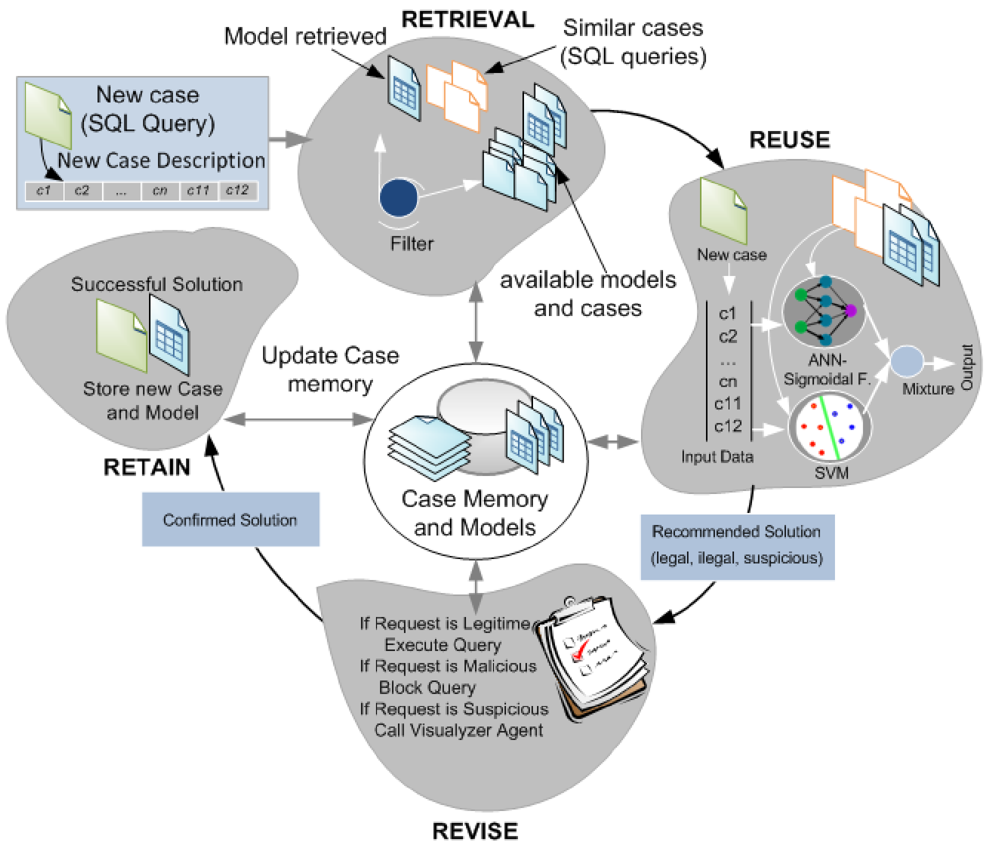


Fig. 2. CBR cycle and classification mechanism of the idCBR agent.

Fig. 2 depicts the different stages applied in the reasoning cycle. To summarize, the retrieval stage involves a selection of queries sorted by type and by the memory's classification models. In the reuse phase, an MLP and an SVM [57] are applied simultaneously to carry out the prediction of the new query. During learning, the memory information regarding the cases and models is updated.

The different stages of the CBR reasoning cycle associated with the system are described as follows.

- **Retrieve**: it is divided 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 MLP mlp_r and svm_r associated with the recovered cases are retrieved. The recovery of these memory models improves 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.
- **Reuse**: it begins 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 reducing 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, there is one neuron in the output layer. The sigmoid activation function has been selected for the different layers. Taking into account the activation function f_j, the calculation of output values is given by the following expression:

$$y_j^p = f_j\left(\sum_{i=1}^{N} w_{ji}(t)\, x_i^p(t) \; + \theta_j\right) \qquad \textbf{(8)}$$

The outputs correspond to x^r. As the neurons exiting from the hidden layer of the neural network contain sigmoid 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 [51].

At the same time that the estimation through the use of neuronal networks is performed, an 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 [10]. SVM also separates the element classes which are not linearly separable. In order to do so, the space of initial coordinates is mapped in a high dimensionality space through the use of functions. Given 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. To do so, a series of non-linear functions called kernels is used.

Let us consider a set of patterns $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ where x_i is a vector of the dimension *n*. The aim 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 $\Phi(T) = \{(\Phi(x_1), y_1), (\Phi(x_2), y_2), \ldots, (\Phi(x_m), y_m)\}$ that, depending on the selected function $\Phi(x)$, could be linearly separable. To carry out the classification, this equation sign is studied [13]:

$$class(x_k) = sign\left(\sum_{i=1}^{m} \lambda_i y_i \Phi(x_i)\Phi(x_k) + b \right) \qquad \textbf{(9)}$$

where λ_i is a Lagrange multiplier, y_i output value for the x_i, b constant.

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 a weighted average in function of the error rate of each one of the techniques. Before calculating the average, the values are normalized to the interval [0, 1]. As SVM provides positive and negative values and those of greater magnitude, the calculation could affect the final value in greater measure if it is not redimensioned.

- **Revise**: for those cases detected as suspicious, with output values determined experimentally in the interval [0.35, 0.6], a review by a human expert is performed. To facilitate the interaction of the human expert, a sophisticated mechanism based on visualization techniques was incorporated in the Administration layer. This mechanism allows the security expert to manage suspicious cases with greater precision. In cases with queries clearly classified as attacks, these are rejected, and the queries clearly classified as legitimate are allowed to run on the database. The suspicious queries are rejected and subsequently validated in the revise phase for a subsequent execution.
- **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.

5.2 Visualizer Agent

This section presents an agent especially designed to resolve user requests that have been classified as suspicious. Its main function is to complement the classification of SQL attacks through visualization facilities. As a result, this agent improves the classification performance of the idMAS-SQL agent.

This agent visualizes all the previously classified queries, highlighting those most similar to the new suspicious query. The selection of similar cases is carried out through the use of a neuronal Growing Cell Structures (GCS) [28] network, that distributes the previously stored cases in meshes and selects the mesh in which the new case is found. To visualize the cases (those in the selected mesh), the dimensionality of data is reduced by means of a projection model. The information is represented and the associated queries are recovered with the retrieved mesh, as shown in Fig. 5. For the purpose of facilitating the revise phase, CART [7] is applied to extract the relevant field in the probes that have been removed. The information about the fields can help us to understand the reasons that queries are classified as legal or illegal.

6. Experimental Results

A comprehensive set of experiments was designed and carried out to check the proposed approach. As a result, a sample web application with access to a MySQL 5.0 database was developed. After creating the database, legal queries were sent from the designed user interfaces. These requests were filtered to avoid redundancy and only legal SQL queries were gathered to generate the dataset. In the case of malicious queries, the dispatch of the queries was automated using the agent SQLMAP0.5 [20].

This tool is able to fingerprint an extensive DBMS back-end, retrieve remote DBMS databases, usernames, tables, and columns, enumerate entire DBMS, read system files, and much more, taking advantage of web application programming security flaws that lead to SQL injection vulnerabilities. Although the SQLMap 0.5 tool generates a wide variety of malicious queries by using different strategies of attack, these queries were also filtered to remove any similar SQL string previously stored.

For the classification process and application of the projection models, the SQL strings were syntactically analyzed, storing the fields in the dataset as listed in Table 2.

Table 2. Dataset fields obtained from the syntactic analysis of SQL queries.

Field	Description	Type (*Values)*
Affected_table	Number of *tables* affected by the query	Int (*n tables*)
Affected_field	Number of *fields* affected by the query	Int (*n fields*)
Command_type	Type of declared *command* in the query	Int (*0-3*)
Word_GroupBy	Number of repetitions of *Group By* clause	Int (*n clause*)
Word_Having	Number of repetitions of *Having* clause	Int (*n clause*)
Word_OrderBy	Number of repetitions of *Order By* clause	Int (*n clause*)
Number_And	Number of repetitions of the *And* Operator	Int (*n ops*)
Number_Or	Number of repetitions of the *Or* Operator	Int (*n ops*)
Number_literals	Number of *Literal* in the SQL string	Int (*n literals*)
Number_LOL	Number of declared Expressions *Literal-Operator-Literal* in the SQL String	Int (*n exprs*)
Length_SQL_String	Length of the SQL String	Int (*n chars*)

To analyze the successful rates, different classifiers were applied: Bayesian Network, Naive Bayes, AdaBoost M1, Bagging, DecisionStump, J48, JRIP, LMT, Logistic, LogitBoost, MultiBoosting AdaBoost, OneR, SMO, and Stacking. The software used for the experiments includes the libraries provided by WEKA [36] and the R script language. This software was used to compare the different classifiers, whose names come from those listed in the libraries. These different classifiers were applied to 705 previously classified queries (437 legal, 268 attacks). The consecutive process for carrying out the output test was the following: select one of the cases, extract it from the set, construct the model starting from the remaining cases and classify 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 3.

Table 3. Percentage of hits for the different classifiers.

Method	Success rate	Method	Success rate	Method	Success rate
BayesNet	90.50	Naive Bayes	94.47	AdaBoostM1	94.33
Bagging	97.02	DecisionStump	84.82	J48	97.73
JRIP	98.16	LMT	98.30	Logistic	97.59
LogitBoost	96.45	MultiBoostAB	94.47	OneR	88.23
SMO	97.16	Stacking	61.99	idCBR	99.01

As shown in Table 3, the highest-performance is obtained by the idCBR, which has a success rate of 99.01. Figure 3 shows the ROC curves for the methods presented in table 3. As shown in Figure 3, the idCBR method presents the highest area under the curve (AUC), and can be consequently considered to be the method that provides the best results.

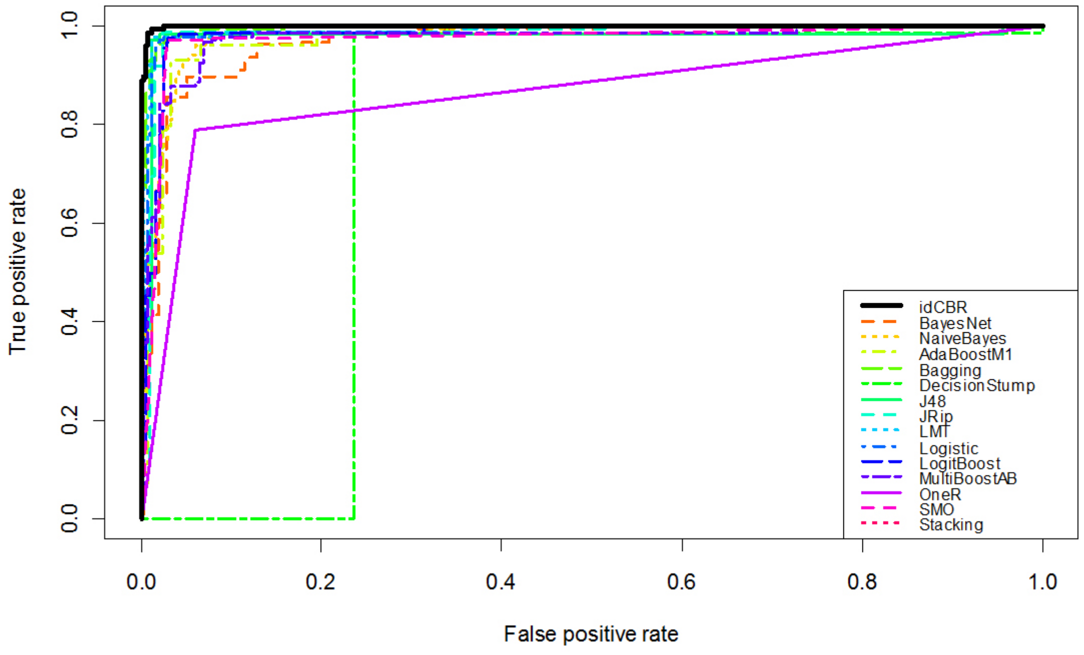


Fig. 3. ROC curves for the different classifiers shown in Table 3.

To analyze the reuse phase in depth, a mixture was carried out between all of the pairs of methods displayed in Table 3. The results are shown in Table 4, which is divided into two halves by the principal diagonal. The upper part of the principal diagonal contains the percentage of decisions for the combination of methods of the column file. In these values, estimations of the dubious cases are included. The part corresponding to the lower diagonal contains the percentage of cases that can be considered as suspicious among those classified. In the combination, it is clear that there is no method that exceeds the number of decisions of the proposed procedure: 99.01. The number of cases detected as suspicious, with an output between the values of 0.35 and 0.6, was limited to 6.

Table 4. Mixture of experts with combinations of the different classifiers. The upper values of the principal diagonal correspond to the percentage of elements successfully classified taking suspicious cases into account; the lower values to the percentage of cases detected as suspicious.

	BayesNet	**NaiveBayes**	**AdaBoostM1**	**AdaBoostM1**	**DecisionStump**	**J48**	**JRip**	**LMT**	**Logistic**	**LogitBoost**	**MultiBoostAB**	**OneR**	**SMO**	**Stacking**
BayesNet		93.48	94.47	95.89	87.66	98.01	98.16	97.16	96.60	96.03	93.90	88.23	97.16	89.79
NaiveBayes	3.40		95.18	96.03	93.62	98.01	98.30	97.30	96.74	96.03	95.46	88.23	97.16	94.04
AdaBoostM1	5.82	3.97		96.17	94.04	98.01	98.16	97.59	96.45	96.03	94.04	88.23	97.16	95.18
Bagging	2.55	2.13	0.85		96.45	98.01	98.16	97.59	97.87	97.02	96.45	88.23	97.16	97.30
DecisionStump	9.36	10.50	11.49	12.91		98.01	98.16	97.45	97.59	94.89	93.33	88.23	97.16	84.82
J48	2.84	2.55	2.41	0.71	13.76		98.16	97.87	98.30	98.01	98.01	98.01	98.30	97.73
JRip	3.55	3.26	2.55	1.13	14.18	1.13		97.87	98.30	98.16	97.87	90.92	98.16	98.16
LMT	4.68	4.40	3.12	2.55	15.46	1.56	1.42		97.45	97.87	97.02	95.32	97.16	98.30
Logistic	5.53	3.12	2.41	1.70	14.04	0.71	1.56	1.70		97.73	96.17	88.23	97.16	96.74
LogitBoost	6.24	2.41	2.41	1.56	12.48	0.71	0.99	2.13	2.55		96.17	88.23	97.16	96.74
MultiBoostAB	5.11	4.96	0.57	2.41	11.06	3.40	3.97	4.96	3.55	2.98		88.23	97.16	94.04
OneR	8.51	8.23	6.10	8.23	18.44	10.35	11.06	11.63	9.50	4.40	10.07		96.74	88.23
SMO	2.13	1.99	0.85	0.43	13.48	1.70	2.41	2.41	0.14	0.28	2.70	10.64		97.16
Stacking	9.22	3.40	9.22	3.40	52.06	0.43	0.00	0.85	3.40	10.92	5.11	0.00	0.00	

To evaluate the significance of the different techniques presented in Table 4, a cross validation was established following the Dietterich's 5x2- Cross-Validation Paired t-Test algorithm [24]. The value 5 in the algorithm name represents the number of replications of the training process, and value 2 is the number of sets into which the global set is divided. Thus, for each technique, the global dataset S was divided into two groups S_1 and S_2 as follows: $S = S_1 \cup S_2$ and $S_1 \cap S_2 = \phi$. The learning and estimation stages were then carried out. This process was repeated 5 times for each technique, and included the following steps: the classifier was trained using S_2 and was then used to classify S_2 and S_1. In a second step, the classifier was trained using S_1 and was then used to classify S_2 and S_1. The results obtained are shown in Table 5, where the columns represent the success rate obtained for S_1, S_2 (R_i-A trained with S_1) and S_1, S_2 (R_i-B trained with S_2) for each *i* repetition. The rows of Table 5 show the different classifiers previously listed in Table 3.

Table 5. Number of errors obtained using the training and cross validation 5x2.

	R_1-A		R_1-B		R_2-A		R_2-B		R_3-A		R_3-B		R_4-A		R_4-B		R_5-A		R_5-B		
	S_1	S_2	S_1	S_2	S_1	S_2	S_1	S_2	S_1	S_2	S_1	S_2	S_1	S_2	S_1	S_2	S_1	S_2	S_1	S_2	Average
BayesNet	319	317	313	316	322	303	317	325	317	312	317	321	318	316	318	318	314	315	312	303	316.25
NaiveBayes	333	333	334	332	336	328	329	334	333	333	334	332	333	334	334	331	330	333	334	332	332.62
AdaBoostM1	331	331	335	325	342	332	329	321	331	328	339	339	330	335	341	331	333	331	341	342	332.80
Bagging	345	345	346	344	349	341	340	342	344	345	346	342	340	339	348	344	345	343	347	345	343.93
DecisionStump	294	304	304	294	309	289	310	304	304	294	294	304	297	301	301	297	296	302	302	296	299.90
J48	345	342	351	341	349	343	340	339	347	340	345	342	348	342	349	339	347	340	348	344	344.01
JRip	348	344	338	335	349	336	336	341	346	344	346	344	342	340	346	340	343	342	344	344	342.27
LMT	352	347	353	336	352	341	345	344	345	343	353	343	349	341	353	343	351	341	350	345	346.35
Logistic	343	343	347	338	347	336	339	347	342	343	344	340	342	344	346	340	345	344	344	345	342.82
LogitBoost	344	343	338	331	343	340	338	340	336	336	342	338	340	336	343	342	342	339	342	343	339.60
MultiBoostAB	335	335	338	335	309	289	325	312	312	310	335	335	310	304	334	336	337	336	323	324	323.03
OneR	301	321	321	301	315	291	312	310	312	302	310	312	312	310	310	312	311	311	311	311	309.58
SMO	342	344	339	340	341	342	339	342	339	342	343	340	340	341	344	341	342	339	342	341	341.15
Stacking	208	229	229	208	217	220	220	217	219	218	218	219	226	211	211	226	218	219	219	218	218.36
idCBR	348	348	342	352	352	343	350	347	347	346	346	350	347	350	339	351	349	346	346	349	347.28

Once the results presented in Table 5 were obtained, a study on the significance of the different classification techniques was performed by applying the Mann-Whitney U-test. It was a non-parametric test in which it is not necessary to make assumptions on the data distribution, as in the t-test. The test determines two values: H_0 and H_1. H_0 shows whether the data in both groups presents the same distribution, whereas H_1 determines if there is difference in the distribution of the error distance data. The upper diagonal of table 6 shows the level of significance for the statistical test, so that if the value obtained is lower than the level of significance (commonly 0.05), we can conclude that the methods are different. In the upper diagonal of Table 6 it is possible to observe those methods in which differences were detected (in bold), and those in which no differences were detected (in red). Clearly, the error distribution for the idCBR approach differs from the error distribution for the rest of the methods, from which we can conclude that there does exist a difference with all but one (LMT) of the methods considered.

The analysis of the cross validation is completed using the Dietterich's 5x2- Cross-Validation Paired t-Test [24]. The results obtained are shown in the lower diagonal of Table 6. It is possible to observe that the results are very similar to those previously shown in the upper diagonal (Mann-Whitney U-test). In this case, the only technique that provides results similar to idCBR (i.e., the difference is not significant) is LMT (the value obtained is higher than the level of significance: α).

Table 6. Mann-Whitney and Paired t-Test test for the significance of the differences. The upper diagonal contains the Mann-Whitney U-Test and the lower diagonal contains the t-Test.

	BayesNet	NaiveBayes	AdaBoostM1	Bagging	DecisionStump	J48	JRip	LMT	Logistic	LogitBoost	MultiBoostAB	OneR	SMO	Stacking	idCBR
BayesNet		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.054	0.001	0.000	0.000	0.000
NaiveBayes	0.000		**0.913**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	**0.796**	0.000	0.000	0.000	0.000
AdaBoostM1	0.000	**0.555**		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.087	0.000	0.000	0.000	0.000
Bagging	0.000	0.000	0.000		0.000	**0.935**	0.200	0.153	0.300	0.000	0.000	0.000	0.001	0.000	0.001
DecisionStump	0.000	0.000	0.000	0.000		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
J48	0.000	0.000	0.000	**0.946**	0.000		0.315	0.095	0.463	0.003	0.000	0.000	0.017	0.000	0.009
JRip	0.000	0.000	0.000	**0.038**	0.000	**0.098**		0.022	**0.703**	0.026	0.000	0.000	0.101	0.000	0.000
LMT	0.000	0.000	0.000	0.017	0.000	0.005	0.000		0.039	0.000	0.000	0.000	0.000	0.000	**0.506**
Logistic	0.000	0.000	0.000	**0.114**	0.000	**0.191**	**0.480**	0.001		0.003	0.000	0.000	0.015	0.000	0.000
LogitBoost	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.000	0.001		0.000	0.000	0.324	0.000	0.000
MultiBoostAB	0.019	0.012	0.009	0.000	0.000	0.000	0.000	0.000	0.000	0.000		0.003	0.000	0.000	0.000
OneR	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000		0.000	0.000	0.000
SMO	0.000	0.000	0.000	0.000	0.000	0.004	**0.141**	0.000	0.018	**0.041**	0.000	0.000		0.000	0.000
Stacking	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000		0.000
idCBR	0.000	0.000	0.000	0.004	0.000	0.021	0.000	**0.509**	0.001	0.000	0.000	0.000	0.000	0.000	

The information presented in Table 5 is graphically represented with a boxplot in Figure 4. It is possible to observe that the variability of the data is low for the idCBR technique, while the other techniques (e.g., LMT) present a high variability. Additionally, idCBR presents a higher success rate (although this fact cannot be determined statistically) due to the variability of the LMT algorithm. The best success rate is that of the idCBR method, as seen in Figure 4. The global average, without distinguishing between the test and the validation, is 692.5 (98.23%) for the LMT method and 694.6 (98.54%) for the proposed idCBR method. The number of successful classifications of the idCBR method increases as the memory of cases grows, while the success rate for the LMT remains constant, as shown in Table 3.

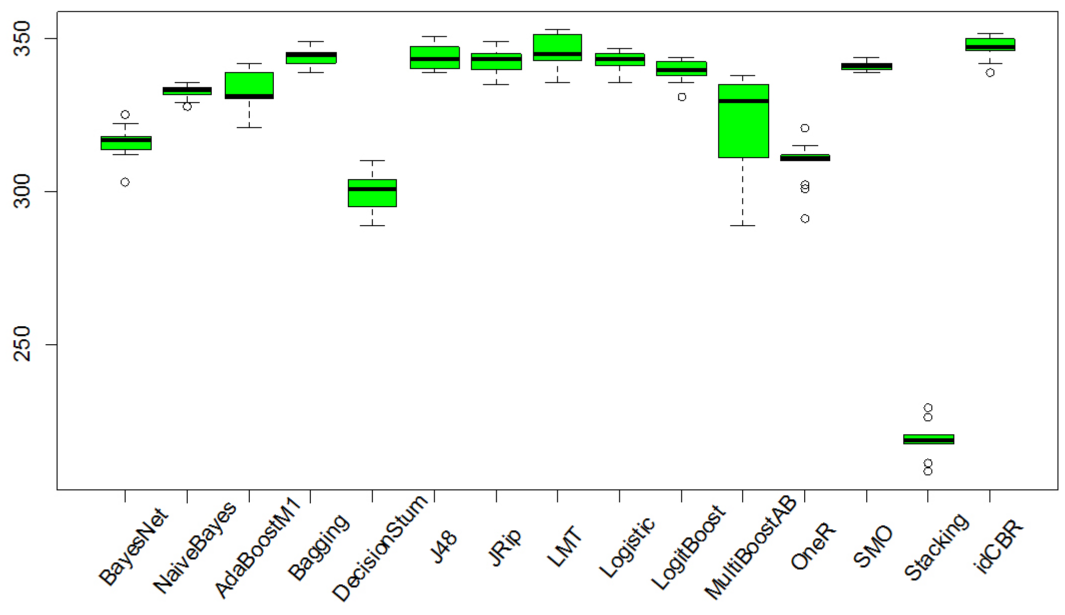


Fig. 4. Boxplots containing the information about the success rate for each technique considering the information shown in Table 5.

The same analysis was repeated using only the data in Table 5 representing classification errors in the test sets. In this case H_0, although close, was not rejected when comparing LMT to idCBR because the value obtained was 0.08. This fact indicates that the idCBR has a better generalization capacity than the LMT technique. Similarly, if the only errors selected were those related to the test, then the result obtained from applying the paired t-test was 0.13, which will not allow us to determine the difference between both methods, although it is the closest to being rejected.

To compare the execution times, we proceeded to perform a classification for 70,500 requests for each technique shown in Table 3. The results obtained are shown in Table 7. This table shows the total execution time (in seconds) for the 70,500 requests and the average execution time in milliseconds.

Table 7. Comparison of the execution time for the different techniques.

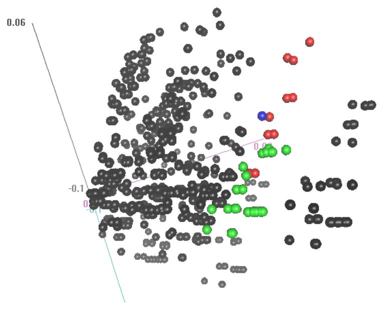
	Total Execution Time (s)	**Average Execution Time (ms)**
BayesNet	3.24	0.04595745
NaiveBayes	3.48	0.0493617
AdaBoostM1	2.78	0.03943262
Bagging	2.69	0.03815603
DecisionStump	2.64	0.03744681
J48	2.63	0.03730496
JRip	2.63	0.03730496
LMT	3.68	0.05219858
Logistic	3.1	0.04397163
LogitBoost	3.15	0.04468085
MultiBoostAB	3.2	0.04539007
OneR	3.13	0.04439716
SMO	3.58	0.05078014
Stacking	2.9	0.04113475
idCBR	5.35	0.07588652

In order to validate the revise phase, we proceeded to recover one of the suspicious queries and to carry out a visualization of the results. The query recovered is shown below:

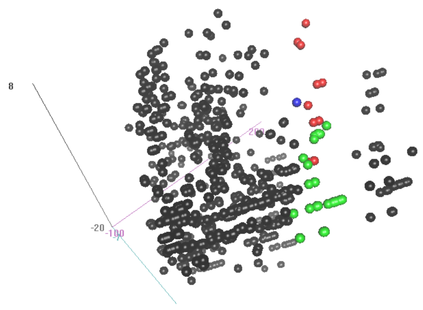
```
select id_cliente, count(*) from client_order, client where id_client = id and id = 'test' AND '1' = '2' and date between '2008-05-05 00:00:00' and '2008-05-05 23:23:23' group by id having city = 'Madrid' order by id
```

This query presents suspicious information because it contains a literal '1'='2' that could be considered an attack. As this content does not make sense in a legal query, it could be interpreted as an attack. The output value obtained by the idCBR agent was 0.541, therefore it will be classified as an attack, as originally proposed.

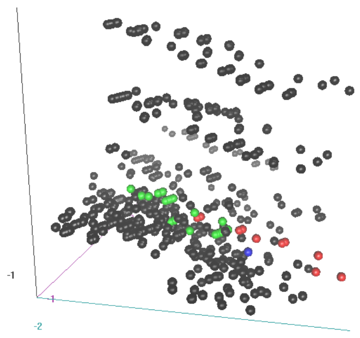
To carry out the revision, the two previously mentioned different techniques of dimensionality reduction were compared. Fig. 5 shows the visualizations provided by the different techniques, which represent the selected query together with the most similar queries. The stored queries are depicted in different colours: the sample query (detailed above) is shown in blue, legal queries in green, attacks in red, and non-recovered queries in grey.



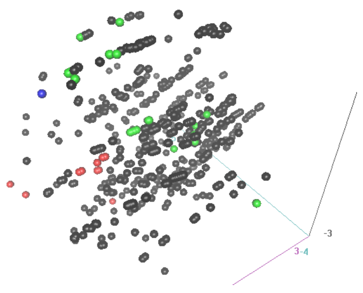
a) PCA projection



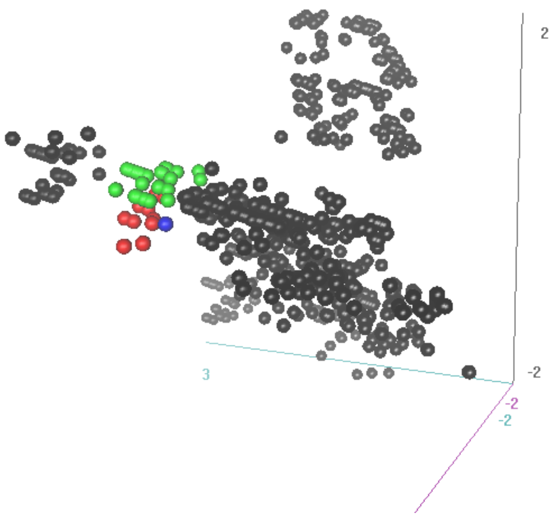
b) MDS projection



c) MLHL projection



d) CCA projection



e) CMLHL projection

Fig. 5. Representation of queries during the revise phase, provided by different dimensionality techniques.

As can be seen, the dimensionality reduction that provides the best results is that of CMLHL, since the retrieved similar cases are closer. Furthermore, it is easier for the naked eye to detect clusters with this procedure than with the others applied. Based on the results in Fig. 5.e, as provided by CMLHL, the query highlighted in blue can be seen closest to the queries in red, from which it is possible to conclude that the query is probably an attack.

Once it is confirmed that the recovered queries results are significant with regards to their proximity in the projection to the suspicious query, the queries are displayed so that the revision can be carried out. In this case, 39 similar queries were recovered. Two examples of the recovered queries are:

```
select id_cliente, count(*) from client_order, client where id_client = id and id = 'test' and date between '2008-05-05 00:00:00' and '2008-05-05 23:23:23' group by id having city = 'Madrid' order by id

select id_cliente, count(*) from client_order, client where id_client = id and id = 'test' AND ORD(MID((SELECT 4 FROM information_schema.TABLES LIMIT 0, 1), 2, 1)) > 1 AND '1'='1' and date between '2008-05-05 00:00:00' and '2008-05-05 23:23:23' group by id having city = 'Madrid' order by id
```

The first corresponds to a legal query while the second corresponds to an attack. In order to facilitate the revise phase, knowledge extraction is carried out to identify the variables that determine the relevant attributes that establish their classification. To this end, CART was applied, as it allows the selection of important attributes according to the rules shown in Fig. 6. Each one of the rows corresponds to a tree branch that contains the number of assigned nodes, the condition, the number of nodes correctly classified, the misclassified nodes, and the class they belong to. The probability of each one of the classes being assigned to the nodes (legal C0, illegal C1) is indicated within parentheses (Fig. 6).

```
n= 39

node), split, n, loss, yval, (yprob)

     * denotes terminal node

1) root 39 11 C0 (0.71794872 0.28205128)

  2) Number_literals< 4.5 30  2 C0 (0.93333333 0.06666667) *

  3) Number_literals>=4.5 9  0 C1 (0.00000000 1.00000000) *
```

Fig. 6. Rules of the Decision Tree generated by CART.

As can be seen, the field Number_literals is what determines if the queries are attacks within the recovered results in the revise phase. Therefore, if the presence of literals in the query is detected as an amount that could be reviewed as an attack, the dubious query displayed in Fig. 5 could be classified as an attack.

7. Conclusions and Future Work

The combination of different AI and Data-Mining paradigms allows the development of a hybrid IDS with characteristics such as the capacity for learning and reasoning, and flexibility and robustness, which make the detection of SQL injection attacks possible. The proposed idMAS-SQL architecture is capable of detecting these abnormal situations with lower error rates than other existing techniques. The idCBR agent is selected because of a mixture which allows the efficient detection of different types of queries, as shown in Table 2, and because it improves the results proposed by the mixture of other experts, as shown in Table 3. The idCBR agent incorporates both a neural network, which makes an estimation of the most atypical cases compared with those existing in the database, and the SVM, which behaves effectively for similar queries; hence the efficiency of both techniques.

The idCBR agent also provides a decision mechanism, based on CART, which facilitates the review of suspicious queries through the selection of similar queries and their visualization through projection models in the Visualizer agent. Visualization facilitates the expert's decision making through a graphical representation of similar queries. Different existing techniques for performing dimensionality reduction have been analyzed, and the conclusions demonstrate that CMLHL displays clearer projections which allow a simpler interpretation of results. Furthermore, the technique of knowledge extraction enables the recovery of relevant information which permits the grouping of queries, as shown in the example in Section 6.

The proposed architecture in general and the idCBR and Visualizer agents in particular, could easily be applied to the detection of other application-layer intrusions. Thus, further research will focus on the adaptation of idMAS-SQL to cover any potential vulnerabilities. To do so, the key features of the packets involved in the target intrusions must be selected, and relevant data must be gathered by the Sniffer agent. There is no need to redefine the techniques of idCBR and Visualizer agents, as the generalization capability of the comprised neural models allows its application to new problems.

Future work will be also based on the comparison to other neural and ensemble models for the visualization of anomalous cases.

Acknowledgements

This research has been partially supported by the Spanish Ministry of Science projects OVAMAH (TIN 2009-13839-C03-03) and MIDAS (TIN 2010-21272-C02-01), funded by the European Regional Development Fund, projects of the Junta of Castilla and León BU006A08 and JCYL-2002-05; Projects of the Spanish Government SA071A08, CIT-020000-2008-2 and CIT-020000-2009-12; the Professional Excellence Program 2006-2010 IFARHU-SENACYT-Panama. 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 Ministry.

References

[1] C. Ahlberg, B. Shneiderman, Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays, in: Readings in Information Visualization: using Vision to Think, Morgan Kaufmann Publishers Inc., 1999, pp. 244-250.

[2] J. Bajo, J.M. Corchado, C. Pinzón, Y.D. Paz, B. Pérez-Lancho, SCMAS: A Distributed Hierarchical Multi-Agent Architecture for Blocking Attacks to Databases, International Journal of Innovative Computing, Information and Control, 6 (2010) 3787–3817.

[3] J. Bajo, J. Vicente, J.M. Corchado, C. Carrascosa, Y.D. Paz, V. Botti, J.F.D. Paz, An execution time planner for the ARTIS agent architecture, Engineering Applications of Artificial Intelligence, 21 (2008) 769-784.

[4] R.A. Becker, S.G. Eick, A.R. Wilks, Visualizing Network Data, IEEE Transactions on Visualization and Computer Graphics, 1 (1995) 16-28.

[5] E. Bertino, A. Kamra, J. Early, Profiling Database Applications to Detect SQL Injection Attacks, in: Performance, Computing, and Communications Conference (IPCCC'2007), New Orleans, LA, USA 2007, pp. 449-458.

[6] C. Bockermann, M. Apel, M. Meier, Learning SQL for Database Intrusion Detection Using Context-Sensitive Modelling (Extended Abstract), in: 6th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA '09), Springer-Verlag, Berlin, Heidelberg, 2009, pp. 196-205.

[7] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth Inc., Belmont, CA, 358 (1984).

[8] G. Conti, Security Data Visualization: Graphical Techniques for Network Analysis, No Starch Press, 2007.

[9] G. Conti, K. Abdullah, Passive Visual Fingerprinting of Network Attack Tools, in: 2004 ACM Workshop on Visualization and Data Mining for Computer Security, ACM, Washington DC, USA, 2004, pp. 45-54.

[10] E. Corchado, C. Fyfe, Connectionist Techniques for the Identification and Suppression of Interfering Underlying Factors, International Journal of Pattern Recognition and Artificial Intelligence, 17 (2003) 1447-1466.

[11] E. Corchado, Y. Han, C. Fyfe, Structuring Global Responses of Local Filters Using Lateral Connections, Journal of Experimental & Theoretical Artificial Intelligence, 15 (2003) 473-487.

[12] E. Corchado, Á. Herrero, Neural Visualization of Network Traffic Data for Intrusion Detection, Applied Soft Computing, ("Accepted - In press") (2010).

[13] E. Corchado, D. MacDonald, C. Fyfe, Maximum and Minimum Likelihood Hebbian Learning for Exploratory Projection Pursuit, Data Mining and Knowledge Discovery, 8 (2004) 203-225.

[14] E. Corchado, M.A. Pellicer, M.L. Borrajo, A MLHL Based Method to an Agent-Based Architecture, International Journal of Computer Mathematics (Accepted - In press), (2009).

[15] J.M. Corchado, R. Laza, Constructing Deliberative Agents with Case-Based Reasoning Technology, International Journal of Intelligent Systems, 18 (2003) 1227-1241.

[16] J.M. Corchado, R. Laza, Constructing deliberative agents with case-based reasoning technology, International Journal of Intelligent Systems, 18 (2003) 1227-1241.

[17] T.F. Cox, G. Ferry, Discriminant Analysis using Non-metric Multidimensional Scaling, Pattern Recognition, 26 (1993) 145-153.

[18] H. Choi, H. Lee, H. Kim, Fast Detection and Visualization of Network Attacks on Parallel Coordinates, Computers & Security, 28 (2009) 276-288.

[19] A. Chuvakin, Monitoring IDS, Information Security Journal: A Global Perspective, 12 (2004) 12 - 16.

[20] B. Damele, SQLMAP0.5 – Automated SQL Injection Tool, in, 2007.

[21] P. Demartines, Analyze de données par réseaux de neurones auto-organizés, in, Institut National Polytechnique de Grenoble, 1994.

[22] P. Demartines, J. Herault, Curvilinear Component Analysis: A Self-Organizing Neural Network for Nonlinear Mapping of Data Sets, IEEE Transactions on Neural Networks, 8 (1997) 148-154.

[23] P. Diaconis, D. Freedman, Asymptotics of Graphical Projection Pursuit, The Annals of Statistics, 12 (1984) 793-815.

[24] T.G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural Computation (1998) 1895-1923

[25] R.F. Erbacher, K.L. Walker, D.A. Frincke, Intrusion and Misuse Detection in Large-scale Systems, IEEE Computer Graphics and Applications, 22 (2002) 38-47.

[26] D. Fisch, A. Hofmann, B. Sick, On the Versatility of Radial Basis Function Neural Networks: A Case Study in the Field of Intrusion Detection, Information Sciences, 180 (2010) 2421-2439.

[27] J.H. Friedman, J.W. Tukey, A Projection Pursuit Algorithm for Exploratory Data-Analysis, IEEE Transactions on Computers, 23 (1974) 881-890.

[28] B. Fritzke, A Growing Neural Gas Network Learns Topologies, Advances in Neural Information Processing Systems, 7 (1995) 625-632.

[29] C. Fyfe, A Neural Network for PCA and Beyond, Neural Processing Letters, 6 (1997) 33-41.

[30] C. Fyfe, R. Baddeley, D.R. McGregor, Exploratory Projection Pursuit: an Artificial Neural Network Approach, in: Research Report/94/160, University of Strathclyde, 1994.

[31] C. Fyfe, E. Corchado, Maximum Likelihood Hebbian Rules, in: 10th European Symposium on Artificial Neural Networks (ESANN 2002), 2002, pp. 143-148.

[32] V.H. García, R. Monroy, M. Quintana, Web Attack Detection Using ID3, in: Workshop International Federation for Information Processing Santiago, Chile, 2006, pp. 323-332.

[33] M.P. Georgeff, A.L. Lansky, Reactive Reasoning and Planning, in: National Conference on Artificial Intelligence, American Association of Artificial Intelligence, 1987, pp. 677-682.

[34] J.R. Goodall, W.G. Lutters, A. Komlodi, The Work of Intrusion Detection: Rethinking the Role of Security Analysts, in: Americas Conference on Information Systems, 2004, pp. 1421–1427.

[35] W.G.J. Halfond, J. Viegas, A. Orso, A Classification of SQL-Injection Attacks and Countermeasures, in: IEEE International Symposium on Secure Software Engineering, Arlington, VA, USA, 2006.

[36] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA Data Mining Software: An Update, ACM SIGKDD Explorations Newsletter, 11 (2009) 10-18.

[37] Á. Herrero, E. Corchado, Mining Network Traffic Data for Attacks through MOVICAB-IDS, in: Foundations of Computational Intelligence, Springer, 2009, pp. 377-394.

[38] Á. Herrero, E. Corchado, P. Gastaldo, R. Zunino, Neural Projection Techniques for the Visual Inspection of Network Traffic, Neurocomputing, 72 (2009) 3649-3658.

[39] Á. Herrero, E. Corchado, L. Sáiz, A. Abraham, DIPKIP: A Connectionist Knowledge Management System to Identify Knowledge Deficits in Practical Cases, Computational Intelligence, 26 (2010) 26-56.

[40] Á. Herrero, M. Navarro, E. Corchado, V. Julián, RT-MOVICAB-IDS: Addressing Real-Time Intrusion Detection, Future Generation Computer Systems, ("Accepted - In press") (2011).

[41] H. Hotelling, Analysis of a Complex of Statistical Variables into Principal Components, Journal of Education Psychology, 24 (1933) 417-444.

[42] A. Hyvarinen, Complexity pursuit: Separating interesting components from time series, Neural Computation, 13 (2001) 883-898.

[43] A. Hyvärinen, New Approximations of Differential Entropy for Independent Component Analysis and Projection Pursuit, in: 1997 Conference on Advances in Neural Information Processing Systems, MIT Press, 1998, pp. 273 - 279.

[44] T. Itoh, H. Takakura, A. Sawada, K. Koyamada, Hierarchical Visualization of Network Intrusion Detection Data, IEEE Computer Graphics and Applications, 26 (2006) 40-47.

[45] M. Junjin, An Approach for SQL Injection Vulnerability Detection, in: Sixth International Conference on Information Technology: New Generations (ITNG '09), IEEE Computer Society, Washington, DC, USA, 2009, pp. 1411--1414.

[46] A. Kamra, E. Bertino, G. Lebanon, Mechanisms for database intrusion detection and response, in: 2nd SIGMOD PhD workshop on Innovative database research (IDAR'2008), ACM, New York, NY, USA, 2008, pp. 31--36.

[47] K. Kemalis, T. Tzouramanis, SQL-IDS: a specification-based approach for SQL-injection detection, in: ACM symposium on Applied computing (SAC'2008), ACM, Fortaleza, Ceara, Brazil, 2008, pp. 2153-2158.

[48] M. Kiani, A. Clark, G. Mohay, 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, DC, USA, 2008, pp. 47-55.

[49] T. Kohonen, The Self-Organizing Map, Proceedings of the IEEE, 78 (1990) 1464-1480.

[50] G. Kou, Y. Peng, Z. Chen, Y. Shi, Multiple Criteria Mathematical Programming for Multi-class Classification and Application in Network Intrusion Detection, Information Sciences, 179 (2009) 371-381.

[51] Y. LeCun, L. Bottou, G.B. Orr, K.R. Müller, Efficient backprop in: Neural Networks, Tricks of the Trade, Springer Verlag, 1998, pp. 546.

[52] K.-L. Ma, Visualization for Security, ACM SIGGRAPH Computer Graphics, 38 (2004) 4-6.

[53] D.J. Marchette, Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint, Springer-Verlag New York, Inc., 2001.

[54] R. Marty, Applied Security Visualization, Addison-Wesley Professional, 2008.

[55] S. Mukkamala, A.H. Sung, A. Abraham, Intrusion detection using an ensemble of intelligent paradigms, Journal of Network and Computer Applications, 28 (2005) 167--182.

[56] E. Oja, Neural Networks, Principal Components, and Subspaces, International Journal of Neural Systems, 1 (1989) 61-68.

[57] S. Pang, T. Ban, Y. Kadobayashi, N. Kasabov, Personalized Mode Transductive Spanning SVM Classification Tree, Information Sciences, 181 (2011) 2071-2085.

[58] K. Pearson, On Lines and Planes of Closest Fit to Systems of Points in Space, Philosophical Magazine, 2 (1901) 559-572.

[59] S.T. Powers, J. He, A Hybrid Artificial Immune System and Self Organising Map for Network Intrusion Detection, Information Sciences, 178 (2008) 3024-3042.

[60] W. Robertson, G. Vigna, C. Kruegel, R.A. Kemmerer, Using Generalization and Characterization Techniques in the Anomaly-Based Detection of Web Attacks, in: 13th Annual Network and Distributed System Security Symposium (NDSS'2006), San Diego, CA, USA, 2006.

[61] D. Sanger, Contribution Analysis: a Technique for Assigning Responsibilities to Hidden Units in Connectionist Networks, Connection Science, 1 (1989) 115-138.

[62] H.S. Seung, N.D. Socci, D. Lee, The Rectified Gaussian Distribution, Advances in Neural Information Processing Systems, 10 (1998) 350-356.

[63] T. Shon, J. Moon, A Hybrid Machine Learning Approach to Network Anomaly Detection, Information Sciences, 177 (2007) 3799-3821.

[64] F. Valeur, D. Mutz, G. Vigna, A Learning-Based Approach to the Detection of SQL Attacks, in: Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA), Vienna, Austria, 2005, pp. 123-140.

[65] M. Wooldridge, Introduction to MultiAgent Systems, John Wiley & Sons, 2002.