International Journal of Innovative Computing, Information and Control
Volume **6**, Number **4**, April **2010**

ICIC International ©2010 ISSN 1349-4198
pp. 1–10

SCMAS: A DISTRIBUTED HIERARCHICAL MULTI-AGENT ARCHITECTURE FOR BLOCKING ATTACKS TO DATABASES

Javier Bajo[1], Juan M. Corchado[2], Cristian Pinzón[2], Yanira De Paz[2], Belén Pérez-Lancho[2]

[1]Universidad Pontificia de Salamanca
Compañía 5, 37002, Salamanca, Spain
jbajope@upsa.es

[2]Departament of Computer Science
University of Salamanca
Plaza de la Merced s/n, 37008, Salamanca, Spain
corchado@usal.es, cristian_ivanp@usal.es, yanira@usal.es, lancho@usal.es

Received June 2008; revised December 2008

Abstract. *One of the main attacks on databases is the SQL injection attack which causes severe damage both in the commercial aspect and the confidence of users. This paper presents a novel strategy for detecting and preventing SQL injection attacks consisting of a multi-agent based architecture called SCMAS. The SCMAS architecture is structured in hierarchical layers and incorporates SQLCBR agents with improved learning and adaptation capabilities. The SQLCBR agents presented within this paper have been specifically designed to classify SQL injection attacks and to predict the behaviour of malicious users. These agents incorporate a new technique based on a mixture of neural networks and a technique based on a temporal series. This paper begins with a detailed explanation of the SCMAS architecture and the SQLCBR agents. The results of their application to a case study are then presented and discussed*

Keywords: Multi-agent, Case based Reasoning, Security database, SQL injection, Intrusion Detection System

1. Introduction.

For several years, databases have been a key element of the technology components in organizations. Nevertheless, security is a serious problem for databases and it has become a complex task due to continuous threats and the emergence of new vulnerabilities [7]. In addition, the recent emergence of mobile technologies such as the Personal Digital Assistant (PDA), Smart Phone and laptop computer, as well as greater interconnection of networks across wireless networks, have caused a revolution in the supply of services [27]. This new philosophy of communication allows users to access information anywhere and anytime. The problem of open environments is the complexity of providing full protection. Over the last years, one of the most serious security threats around databases has been the SQL Injection attack [25]. In spite of being a well-known type of attack, the SQL injection remains at the top of the published threat list. The solutions proposed so far seem insufficient to block this type of attack because the vast majority are based on centralized mechanisms [26, 29, 30] with little capacity to work in distributed and dynamic environments. Furthermore, the detection and classification mechanisms proposed by these solutions lack the learning and adaptation capabilities for dealing with attacks and variations of the attacks that may appear in the future.

This study presents SQL-CBR Multi-agent System (SCMAS), a distributed hierarchical multi-agent architecture for blocking database attacks. SCMAS proposes a novel strategy to block SQL injection attacks through a distributed approach based on the capacities of the SQLCBR agents, which are a particular type of CBR-BDI agent [3, 17]. The philosophy of multi-agent systems allows SQL injection attacks to be dealt with from the perspective of the elements of communication, ubiquity and autonomous computation, and from the standpoint of a global distributed system. Every component in SCMAS interacts and cooperates to achieve a global common goal. SCMAS presents a hierarchical organization structured by levels or layers of agents. This hierarchical structure distributes roles and tasks for the detection and prevention of SQL injection attacks. The agents of each level are assigned specific tasks which they can execute regardless of their physical location.

Agents can be characterized by their capacities in areas such as autonomy, reasoning, reactivity, social abilities, pro-activeness, and mobility, among others. These capacities provide a great advantage for offering solutions in highly dynamic and distributed environments. Many activities in areas such as networks security, e-commerce, telecommunications, etc., are carried out by multi-agent systems that have been successfully implemented [2, 5, 14, 43]. The ability to execute agents on mobile devices such as PDAs, Smart phones and notebook computers makes them particularly suitable for detecting SQL injection attacks in distributed environments. The SQLCBR agents are CBR-BDI agents [17] integrated in the SCMAS architecture; their internal structure and capacities are based on mental aptitude [23]. The use of SQLCBR agents with advanced capabilities for analyzing and predicting SQL attacks is the main feature of the architecture. These agents are characterized by the integration of a CBR mechanism (Case-Based Reasoning) [3] in a deliberative BDI Agent. This mechanism provides the agents with a greater level of adaptation and learning capacity, since CBR systems make use of past experiences to solve new problems [17]. This is very effective for blocking SQL injection attacks as the mechanism uses a strategy based on anomaly detection [37].

The SQLCBR agents designed within the framework of this research are located on the upper levels of the hierarchical architecture. There are two types of CBR-BDI agents that incorporate two novel strategies for both the classification of SQL injection attacks and the prediction of negative behaviors by users. The first type of SQLCBR agent is a classifier agent called "Anomaly", which analyzes SQL queries and then classifies them according to whether the query is defined as attack or not attack to the database. This classifier agent incorporates a novel mechanism in the adaptation stage of his CBR cycle based on a mixture of neural networks. Neural networks are an effective method of classification [11] for problems of this type since current methods such as the Bayesian method [28], Exponential Regression model [35], Polynomial Regression Model [35] or Lineal Regression model [35] not only provide solutions more slowly, but solutions that are less effective compared to neural networks. Using a mixture of neural networks we can merge two networks that use neurons with distinct activation functions. As a result of this process, the Classifier agent demonstrates a remarkable improvement in the performance of the classifier since it accounts for the assessment carried out for the two neural networks and avoids conflictive cases that the networks cannot resolve on their own. The second type of SQLCBR agent is the Forecaster agent, which incorporates a mechanism to predict the behavior based on a time series technique. This mechanism uses the history of requests made by a user to predict current behavior.

The aim of this paper is to describe the SCMAS architecture and present the preliminary results obtained after the implementation of an initial prototype. These results demonstrate each of the following: the effectiveness of the solution in minimizing and predicting attacks, a higher performance obtained by distributing the workload among the available nodes in the architecture, a greater capacity for learning and adaptation provided by the SQLCBR agents, and the flexibility to be adapted to many scenarios in which information is susceptible to an SQL injection attack. The remainder of this paper is structured as follows: Section 2 presents the problem that has prompted most of this research work; Section 3 describes the SCMAS architecture, different agents incorporated in the architecture, and the cooperation and communication among them; Section 4 outlines the two types of SQLCBR agents used for detecting and preventing attacks. Section 5 describes a case study based on a medical register database. Finally, the final results and conclusion are presented in section 6.

2. Database Threat and Security Revision.

A database allows users to access their data through customized applications. The growth of the Internet and the World Wide Web are two determining factor that have contributed to the evolution of accessing databases [21]. These systems have become a source for the global spread and exchange of data. As such, databases have played a crucial role in storing huge volumes of data. On the other hand, wireless access has enabled a great interconnection among devices and unrestricted data access to all types of data. As a result of the decentralization of information, it has been necessary to address new issues about privacy and information security. As a consequence of the increase in the number of incidents of unauthorized data access, information security is considered a critical issue within the strategic policies of organizations. For this reason a variety of security strategies for protecting information systems and databases from outsider attacks have been explored. Such strategies include firewalls, filters, authentication, communication transport encryption, intrusion detection, auditing, monitoring, honeypots, security tokens, biometric devices, sniffers, active blocking, file level security analysis or demilitarized zone. However, current security measures seem to be insufficient. The more recent types of attacks target the application layer and the database systems, but the protection mechanisms are unable to detect them. Organizations are hit hard when a malicious user bypasses or violates protective measures to steal, modify or destroy sensitive information.

SQL injection attacks are a potential threat at the application layer. The Structure Query Language (SQL) constitutes the backbone of many Database Management Systems, especially relational databases. It carries out information handling and database management, but it also facilitates building a type of attack which results extremely lethal. Table 1 enumerates some of the most commonly used techniques in SQL injection attacks. Although this type of attack has been the subject of many studies; it continues to be one of the most frequent attacks over the Internet.

A SQL injection attack harms an organization through financial losses, loss of the confidence on behalf of the customers, suppliers and business partners, and disruption of the internal and external operations of the organization. A SQL injection attack takes place when a hacker changes the semantic or syntactic logic of a SQL text string by inserting SQL keywords or special symbols within the original SQL command that will be executed at the database layer of an application [25]. 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 [25].

Web applications are the main target of this type of attack. However, even though the most common attack method is through a HTTP (HyperText Transport Protocol) protocol request, other methods are vulnerable to a SQL injection attack. The applications on wireless mobile devices execute SQL queries on the database. These queries are transmitted through insecure transmission channels allowing them to be monitored, captured and changed by a hacker. Finally, a recent vulnerability has arisen through the

Table 1. Classification of well-known techniques used in SQL injection Attacks

Type of injection	Description	Example
SQL Tautologies	The condition will always be true. The classic example of the login form in a web application.	`SELECT * from TblUser where Username = '' or 1=1 -- and pass= ''`
Union Query	It is possible for a hacker to use a union in order to obtain control of the query and retrieve some of its results.	`SELECT * FROM TblSupplier WHERE NameSupplier = '' UNION ALL SELECT * From TblConsumer WHERE 1 = 1`
piggy-backed Queries	A new query is injected within the original query without changing the logic of the first one.	`SELECT * FROM TblUser WHERE UserName = ''; DROP TABLE TblUser -- 'AND pass =''`
Injection based on inference	When there are no error messages that allow information to be retrieved and the attack to continue, the hacker sends queries crafted at the database for inferring answers where the value can be true or false. Other variants are based on inferences according to response time.	`DECLARE @s varchar(8000); SELECT @s = db_name(); if (ascii(substring(@s, 1, 1)) {&} ( power(2, 0))) > 0 waitfor delay '0:0:5'`
Stores Procedures	Generates a false sense of security when storing procedures are implemented inadequately. Allows an extension of functionalities of the databases. It is possible for the hacker to achieve an interaction with the server's operating system.	`Simplequoted.asp? city=seattle';EXEC master.dbo.xp_cmdshell 'cmd.exe dir c:`
Alternative codification	Standard Code for Information Interchange) or a codification in Hexadecimal format.	`DECLARE @q varchar(8000); SELECT @q = 0x73656c65637420 404076657273696f6e; EXEC(@q). Resultado:'SELECT @@version'`

use of a RFID label. This new technology has presented a security hole which can be exploited by a SQL injection attack and cause great damage [40].

There have been many proposed solutions for SQL injection attacks, including some Artificial Intelligence techniques. One of the approaches is DIDAFIT [34] which can efficiently identify anomalous accesses to the database. This technique works by fingerprinting legitimate access patterns of database transactions, and using them to identify database intrusions. Its greatest disadvantage lies in the need to use a set of highly conditioned fingerprints. Huang, et al. [29] propose WAVES (Web Application Vulnerability and Error Scaner). This solution is based on a black-box technique. WAVES is a web crawler that identifies vulnerable points, and then builds attacks that target those points based on a list of patterns and attack techniques. WAVES monitors the response from the application and uses a machine learning technique to improve the attack methodology. WAVES cannot check all the vulnerable points like traditional penetration testing. What's more, the strategy used by the intrusion detection systems has even been implemented in some SQL injection attacks. Valeur et al. [44] present an IDS approach that uses a machine learning technique based on a dataset of legal transactions. These are used during the training phase prior to monitoring and classifying malicious accesses. Generally, IDS systems depend on the quality of the training set since a poor training set would result in a large number of false positives and negatives. Rietta [41] proposed an IDS system at the application layer using an anomaly detection model which assumes certain behaviour of the traffic generated by the SQL queries; specifically, elements within the query (sub-queries, literals, keyword SQL). It also applies general statistics and proposes grouping the queries according to SQL commands and then comparing them against a previously built model. The SQL query that deviates from the normal profile is rejected.

The proposals based on intrusion detection depend on a database, which requires continual updating in order to detect new attacks. Garcia et al. [22] propose an IDS for protecting a Web application that makes use of ID3, a well known classifier that builds a decision tree from a fixed set of examples. Each input example is a Web application query; it has several attributes and belongs to a class identified as either attack or normal. The most important drawback of this solution is that it is non-incremental, which would require the decision tree to be built on a regular basis, using an updated attack signature database. Finally, Skaruz & Seredynski [42] propose the use of a recurrent neural network (RNN). The detection problem becomes a time serial prediction problem. The main problem with this approach is the large number of false positives and false negatives.

Other strategies based on string analysis techniques and the generation of dynamic models have been proposed as solutions to SQL injection attacks. The Java String Analysis (JSA) library [12] provides a mechanism for generating models of Java strings. JSA performs a conservative string analysis of an application and creates automata that express all the possible values that a specific string can have at a given point in the application. This technique does not target SQL injection attacks, but it is important because other approaches use the library to generate middle forms of models. JDBC Checker [24] is a technique for statically checking the type correctness on dynamically generated SQL queries. This technique was not intended to detect and prevent general SQL injection attacks, but can be used to prevent attacks that take advantage of type mismatches in a dynamically generated query string. Wassermann & Su [46] propose an approach that uses a static analysis combined with automated reasoning. The technique verifies that the SQL queries generated in the application usually do not contain a tautology. The technique detects only SQL injections that insert a tautology in the SQL queries, but can not detect other types of SQL injections attacks. Halfond and Orso[26] propose AMNESIA (Analysis and Monitoring for Neutralizing SQL Injection Attacks). This approach uses a static analysis to build models of the SQL queries that an application generates at each point of access to the database. In the dynamic phase, AMNESIA captures all the SQL queries before they are sent to the database and checks each query against the statically built models. Queries that violate the model are classified as SQL injection attacks. AMNESIA depends on the accuracy of static analysis. With only slight variations of accuracy, it generates a large number of false positives and negatives. Kosuga et al. [30] proposes SANIA (Syntactic and Semantic Analysis for Automated Testing against SQL Injection). SANIA captures queries between Web applications and databases. It automatically generates crafted attacks according to the syntax and semantic of vulnerable points. SANIA uses a syntactic analysis tree of the query to evaluate the security of the points. The biggest drawback of SANIA is that it has a significant rate of false positives.

There are other main query development paradigms proposed as solutions to SQL injection attacks. SQLrand [8] provides a framework that allows developers to create SQL queries by using randomized keywords instead of the normal SQL keywords. A proxy between the web application and the database server captures SQL queries and de-randomizes the keywords. The SQL keywords injected by an attacker are not usually constructed by the randomized keywords, so the tainted SQL strings would have a syntax error. SQLrand depends on a secret key to modify keywords, and its security relies on hackers not being able to discover this key. Additionally it requires the application developer to rewrite code. SQL DOM [36] and Safe Query Objects [13] use an encapsulation of database queries to avoid SQL injection attacks. These techniques change the process of building SQL strings to a systematic method that uses a type-checked API. The API is able to systematically apply coding best practices such as input filtering and close-fitting type checking of user input. Although effective, both of these techniques have the disadvantage of requiring developers to learn and use a new programming paradigm or query-development process. Overall there has been a surge in interest in overcoming SQL injection attacks through new solutions. However, the approaches dealing with this type of attack have been limited to centralized models with little flexibility, scalability and effectiveness.

Table 2 compares SCMAS with current artificial intelligence-based models, specifically those that focus on automated learning. Among the previously mentioned models, those with an insufficient or complete lack of similarity with the classification strategy implemented in our solution were omitted from the comparison process.

Table 2. Comparison of selected models approaches vs. SCMAS

	Low et al., 2002	Huang, et al., 2003	Valeur et al., 2005	Rietta, 2006	Garcia et al., 2006	Skaruz et. al 2007	SCMAS
High-Complexity Attacks	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Type of Detection	Misuse	Anomaly / Misuse	Anomaly	Anomaly	Anomaly	Anomaly	Anomaly / Misuse
Distributed Approach	No	No	No	No	No	No	Yes
Learning ability	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Adaptive ability	No	No	No	No	No	No	Yes
Balances the Workload	-	-	No	No	No	No	Yes
Tolerance to Failure	-	-	-	-	-	-	Yes
Scalability	Yes	Yes	-	-	-	-	Yes
Positive / Negative False	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Time Response	Nearly Real time	-	Nearly Real time	-	-	-	Nearly Real time
Ubiquity	No	No	No	No	No	No	Yes

According to the results shown in Table 2, SCMAS outperforms the other models with respect to:

- Type of Detection: SCMAS was strategically designed to exploit the strong points of the most recognized and applied detection techniques.

- Distributed Approach: SCMAS is based on a multi-agent architecture that can execute tasks derived from the classification process in a distributed way.
- Adaptive ability: SCMAS includes two types of intelligent SQLCBR agents that were designed to learn and adapt to changes in attack patterns, new attacks, and types of user behaviour.
- Balances the Workload: SCMAS was designed to distribute the classification task load throughout the various layers of the hierarchical architecture.
- Tolerance to Failure: SCMAS has a hierarchical design that can facilitate error recovery through the instantiation of new agents.
- Scalability: SCMAS is capable of growing (by means of the instantiation of new agents) according to the needs of its environment.
- Ubiquity: SMCAS provides a ubiquitous alert mechanism to notify security personnel in the event of an attack.

Some aspects of SCMAS, such as response time or the required initial learning curve, can be considered a disadvantage when compared to other solutions. Nevertheless, SCMAS provides a much more efficient classification once the system acquires experience, and a reasonably low response time.

The SCMAS architecture presents novel characteristics that have not heretofore been considered in previous approaches. The next section presents the SCMAS architecture in greater detail.

3. SCMAS: A Solution based on Multi-agent System.

The agents handle capacities such as autonomy, social abilities, reasoning, learning, and mobility, among others [47]. One of the main features of agents is their ability to carry out cooperative and collaborative work when they are grouped into multi-agent systems to solve problems in distributed manner [14, 18]. Recent software applications implemented in the electronic commerce, industry and health sectors, among others [2, 5, 14] are suited to work in dynamic environments with ubiquitous access to information, and require the use of mobile technologies. As such, the problem of providing protection against SQL injection attacks requires a different approach. A solution based on a multi-agent architecture presents the most suitable features for resolving the problem. The present study proposes the use of a distributed and hierarchical architecture depicted in Figure 1 to detect and block SQL injection attacks. It is based on a totally innovative approach since there is no known architecture with these characteristics for resolving the problem of SQL injection attacks.

As depicted in Figure 1, the distributed resolution of problems balances the workload, facilitates the recovery from error conditions, and also avoids centralized traffic. While requests in current environments are carried out from several devices, it is preferable for specialized agents to monitor requests from various strategic points and avoid having all requests go directly to the database. Similarly, the analysis, classification, and prediction capabilities, among others, are distributed in a layered structure, where 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, SCMAS architecture uses a model based on a hierarchical model that reduces the complexity of tasks such as monitoring devices and users, classifying user requests, predicting behavior, evaluating the final solution etc. Distributing the functionality at each level, while maintaining each level independent, allows new changes to be easily adapted. Each level of architecture holds a collection of agents with well-defined roles that allow their tasks and responsibilities to be defined. The architecture has been divided into 4 levels so that the specific tasks are assigned according to the degree of complexity. Figure 1, illustrates the SCMAS architecture with each level and the respective agents. The details of each type of agent located at the different levels of SCMAS architecture are presented in Table 3.

The complexity of the components of the architecture increases with each level of hierarchy. The entities at each layer use the functionalities that have been provided to them by entities at the bottom layer. The functions have been divided according to the SCMAS levels. The operational tasks are performed at layer 1 and include: capture traffic generated by user requests across different devices, retrieve the SQL string from the capture, and execute a syntax analysis of the SQL string. Layer 2 contains the tactical strategic tasks for the detection and classification of user requests. The entities at layer 3, the administration layer, carry out tasks related to determining the classification of requests. Additionally, they coordinate the overall functioning of the architecture and determine which measure to take when an attack has been detected. Finally, the tasks of interaction with the user are performed in the top layer of the architecture called the interaction layer, which provides an interface to access services of the architecture.

3.1. **Cooperation of agents for task execution.** In SCMAS architecture each agent is equipped with the ability and resources to achieve its own goals. However, to achieve the overall goal of the architecture (classifying user requests and predicting the behavior of the hacker) it is necessary to have cooperation and communication among the architecture's agents. This subsection addresses the way in which the types of agents within SCMAS cooperate and collaborate. Different situations in which agents work together to achieve a particular goal are: detection by misuse detection (FingerPrint, Sensor and DBPattern agents cooperate by applying misuse detection to accomplish a task); classification of the user requests (FingerPrint and Anomaly agents cooperate by accomplishing tasks related to the classification of user requests); resolving user requests (Manager, Anomaly, Interface and LogUser cooperate to resolve the task of the classifying user requests); responding to user requests (Manager, DB and Response cooperate to provide an answer to user requests); and behavior analysis and blocking of malicious users (Manager, Sensor and Forecaster agents cooperate in the analysis of user behavior for blocking attacks by hackers). Below, an example of cooperation explains how Misuse detection resolves a detection task.

As shown in Figure 2 and Table 4, the cooperation to detect attacks based on misuse detection involves the Sensor agent, which belongs to the bottom layer, and the DBPattern and FingerPrint agents, which are located at level 2. Figure 2 illustrates how the Sensor agent gets the requested SQL string, analyzes it and sends the results to the FingerPrint. The FingerPrint agent carries out a pattern matching. It requires the DBPattern agent to search the database and retrieve patterns similar to the SQL query being analyzed.

3.2. **Communication among agents.** In distributed environments where the agents are applied as solutions, it is essential to provide the necessary mechanisms for communication among the agents so they can perform their tasks efficiently. The cooperation among agents is highly dependent on the efficiency of the communication mechanisms which support the interaction. As part of the architecture that is proposed in this paper, communication mechanisms were identified at each level of the architecture (intra-layer communication), and a global communication among levels for the execution of tasks (communication inter-layer). The SCMAS architecture considers the use and control of mobile devices, laptop computers and workstations, and takes the use of wireless networks into account.

The SCMAS architecture was developed following the recommendations made by FIPA (Foundation for Intelligent Physical Agents) [20]. The physical transfer of messages is

Table 3. SCMAS Architecture's agents

Agent types	Archictecture Level	Quantity	Abilities / Tasks
Sensor	1	n=number of host	Located in each of the devices with access to the database. They have 3 specific functions: a) Capture datagrams launched by the devices. b) Order TCP fragments to extract the request's SQL string. c) Provide Syntactic analysis of the request's SQL string.
FingerPrint	2	n=workload available	Gets the results supplied by the Sensor agent. Its function includes a pattern matching of known attacks. A database with previously built patterns allows this task.
DBPattern	2	1	Cooperates with the FingerPrint agent in the retrieval of patterns from the database. It is also responsible for updating and adding new patterns to the patterns database.
Anomaly	2	n=workload available	A core component of the architecture, it carries out a classification of SQL strings through detection anomalies. It integrates a case based reasoning (CBR) mechanism. In the reuse stage of the CBR cycle it applies a mixture of neural networks to generate a classification (legal, illegal or suspicious).
Manager	3	1	Responsible for the decision-making, evaluation and coordination of the overall operation of architecture. Evaluates the final decisions of classifications and manages attack alerts and coordinates the actions necessary when an attack is detected. Decisions are based on a method of voting among the Anomaly agents.
Forecaster	3	n=workload available	Another core agent in the architecture. Responsible for predicting an attack based on user behavior. Using a technique based on moving averages integrated into the reuse phase of the CRB cycle, the agent is able to predict user behavior and determine an attack against the database.
LogUser	3	1	It is responsible for updating the profile of application users. The requests made to the database are registered, and user profiles are labeled based on the historical behavior. Cooperates with the Anomaly agent to update the users- log.
DB	3	1	It is responsible for executing queries to the database once the requests are classified as legal, and getting the results.
Interface	4	1	It facilitates the interaction between a human expert in charge of security and architecture. It is equipped with the ability to run on mobile devices to achieve direct communication with security personnel whenever an attack is detected. It also facilitates the implementation of adjustments in the setup of the architecture.
Response	4	1	Responsible for formatting and delivering the results of request in device interfaces. The results of valid requests are sent to users; however, invalidated requests are rejected and notified with a warning message.

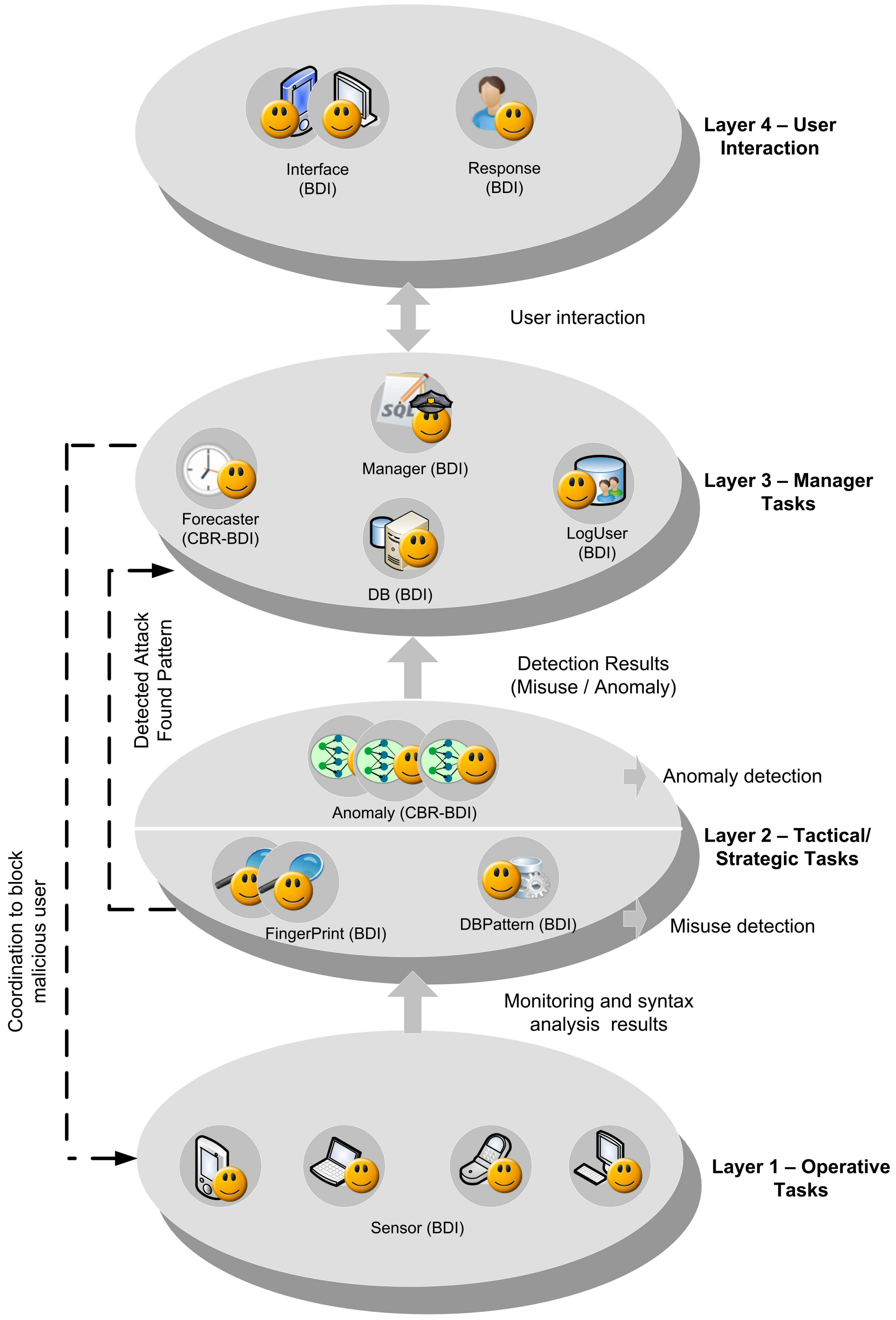

Interface (BDI)
Response (BDI)
Layer 4 – User Interaction
User interaction
Manager (BDI)
Forecaster (CBR-BDI)
LogUser (BDI)
DB (BDI)
Layer 3 – Manager Tasks
Detected Attack Found Pattern
Coordination to block malicious user
Detection Results (Misuse / Anomaly)
Anomaly (CBR-BDI)
Anomaly detection
Layer 2 – Tactical/ Strategic Tasks
FingerPrint (BDI)
DBPattern (BDI)
Misuse detection
Monitoring and syntax analysis results
Sensor (BDI)
Layer 1 – Operative Tasks

Figure 1. SCMAS Architecture - Levels and agents.

carried out using HTTP (HyperText Transport Protocol) and MTP (Message Transport Protocol) as transport protocols. Another component in communication agents is the communication language. In SCMAS architecture, communication among agents is carried out through the exchange of messages. Messages in SCMAS architecture are based on the standard FIPA CAL, which is based on the speech act theory in which messages

Table 4. Cooperation-Detection based on Misuse detection

Agents	Description
Sensor - FingerPrint - DBPattern	• Sensor agents send their results (transformed SQL string, syntax analysis data, and user data) to a specific FingerPrint agent. The FingerPrint executes a pattern matching with known SQL attack patterns. • The FingerPrint agent requests a set of SQL patterns from the DBPattern agent, which manages the pattern database. The DBPattern agent retrieves and sends the set of patterns to the FingerPrint. It is also possible for the DBPattern agent to update the pattern database.

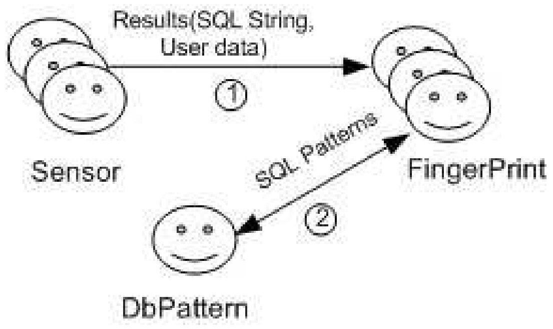
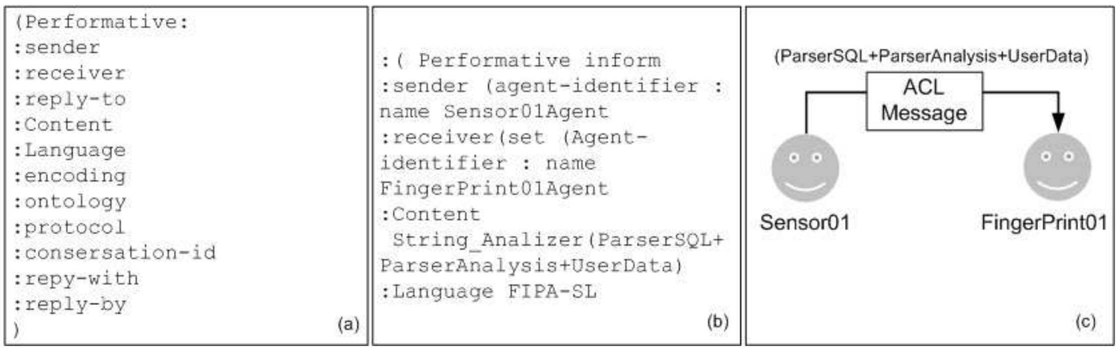


Figure 2. Cooperation to resolve the task based on misuse detection.

are considered communicative acts [10]. Figure 3 presents a model of messages exchanged between agents in the SCMAS architecture. Figure 3 (a) presents a FIPA CAL message showing attributes defined and standardized by FIPA. It is important to note that not all attributes are required in a FIPA CAL message. Figure 3 (b) provides an example of a message relayed between a Sensor agent located on the bottom layer of the architecture and a Fingerprint agent located on level 2. The sensor agent provides the monitoring results by sending the results of the SQL string's syntax analysis and user data. Figure 3 (c) makes a graphic representation of the message exchanged between Sensor01 agent and FingerPrint01 agent.



```
(Performative:
:sender
:receiver
:reply-to
:Content
:Language
:encoding
:ontology
:protocol
:consersation-id
:repy-with
:reply-by
)
```

(a)

```
:( Performative inform
:sender (agent-identifier :
name Sensor01Agent
:receiver(set (Agent-
identifier : name
FingerPrint01Agent
:Content
 String_Analizer(ParserSQL+
ParserAnalysis+UserData)
:Language FIPA-SL
```

(b)

Figure 3. Model of the messages exchanged between agents in the SCMAS architecture.

Finally, to achieve interaction and cooperation among agents within the SCMAS architecture, a set of communication protocols were defined. Figure 4 presents two examples of the communication between two agents through a protocol diagram. Figure 4(a) shows the communication between the FingerPrint agent and the Pattern agent to request SQL patterns stored when the misuse detection is applied. Figure 4(b) shows the message sent by the FingerPrint agent when it sends the results generated by the capture of the SQL query, string analysis and user data.

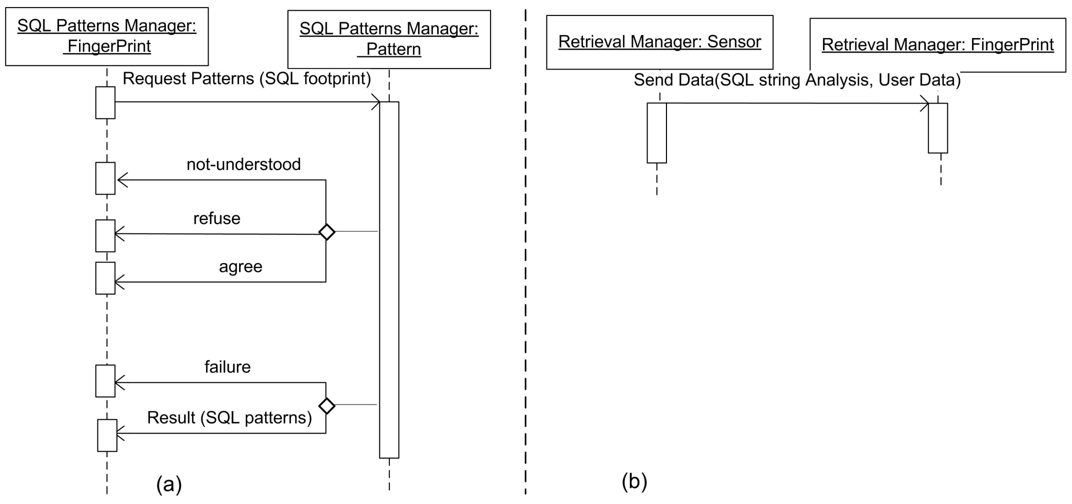

SQL Patterns Manager: FingerPrint
SQL Patterns Manager: Pattern
Request Patterns (SQL footprint)
not-understood
refuse
agree
failure
Result (SQL patterns)
(a)
Retrieval Manager: Sensor
Retrieval Manager: FingerPrint
Send Data(SQL string Analysis, User Data)
(b)

FIGURE 4. Communication pattern during the exchange of a message between agents.

As the final step, elements to provide security in agent communication were taken into account. This resource was provided by a secure channel through the HTTPS protocol (Hypertext Transfer Protocol Secure) [39]. Moreover, the internal-communication among agents of SCMAS architecture was secured by means of JADE-S [6], a JADE plug-in that supports user authentication and agents, encryption and message signature.

4. **SCMAS Agents with Reasoning Capabilities.** The application of agents and multi-agent systems facilitates taking advantage of agent capabilities, such as mobility, pro- activeness or social abilities, and the possibility of distributed problem solving. Within the context of an intelligent environment, agents must be able to respond to events, take initiative according to their goals, communicate with other agents, interact with users, and make use of past experiences to find the best ways to achieve goals. There are many architectures for constructing deliberative agents, and many of them are based on the Beliefs Desires Intentions (BDI) model [47]. In the BDI model, the internal structure of an agent and its capacity to choose is based on mental aptitudes, given that agent behavior is composed of beliefs, desires, and intentions. The beliefs represent its information state, what the agent knows about itself and its environment. The desires are its motivation state, what the agent is trying to achieve. And the intentions represent the agent's deliberative states. Intentions are sequences of actions which can be identified as plans. A BDI architecture has the advantage of being intuitive whereby identifying the process of making decisions and carrying them out are relatively simple tasks.

Case-based Reasoning (CBR) is a type of reasoning based on the use of past experiences [1]. The purpose of case-based reasoning systems is to solve new problems by adapting solutions that have been used to solve similar problems in the past. The fundamental concept when working with case-based reasoning is the concept of case. A case can be defined as a past experience, and is composed of three elements: a problem description which describes the initial problem, a solution which provides the sequence of actions carried out in order to solve the problem, and the final state which describes the state achieved after the solution was applied. A case-based reasoning system manages cases (past experiences) to solve new problems. The way in which cases are managed is known as the case-based reasoning cycle. These systems execute the CBR cycle which consists of four sequential steps: retrieve, reuse, revise and retain [1].

The SQLCBR agents proposed in the framework of this research are a CBR-BDI type of agent specially adapted to resolve the SQL injection attack problem. These agents use the concept of CBR to gain autonomy and improve their problem-solving capabilities. The method proposed in [17] facilitates the incorporation of case-based reasoning systems as a deliberative mechanism within BDI agents, allowing them to learn and adapt themselves, lending them a greater level of autonomy than what is normally found in a typical BDI architecture [9]. Accordingly, SQLCBR-agents can reason autonomously and therefore adapt themselves to environmental changes. The case-based reasoning system is completely integrated within the agent architecture. The SQLCBR are classifier and predictor agents that incorporate a "formalism" that is easy to implement, in which the reasoning process is based on the concept of intention. Intentions can be seen as cases, which have to be retrieved, reused, revised and retained. A direct relationship between case-based reasoning systems and BDI agents can also be established if the problems are defined in the form of states and actions.

Case: <Problem, Solution, Result>	**BDI agent**
Problem: initial_state	Belief: state
Solution: sequence of <action,[intermediate_state]>	Intention: sequence of <action>
Result: final_state	Desire: set of <final_state>

SQLCBR agents implement cases as beliefs, intentions and desires which lead to the resolution of the problem. As described in [4, 14], each state of a CBR-BDI agent is considered as a belief, including the objective to be reached. The intentions are plans of actions that the agent has to carry out in order to achieve its objectives, which makes each intention an ordered set of actions. Each change from state to state is made after carrying out an action (the agent remembers the action carried out in the past, when it was in a specified state, and the subsequent result). A desire will be any of the final states reached in the past (if the agent has to deal with a situation that is similar to one from the past, it will try to achieve a result similar to the one previously obtained). The SQLCBR agents, which are explained in detail below, use these concepts to define a case structure for SQL injection problems, and include specific novel mechanisms in the different phases of the CBR cycle to improve the tasks of classification and prediction.

4.1. **SQLCBR Agent with Classification Capabilities.** The SQLCBR Anomaly agent incorporates a reasoning mechanism that allows it to prevent and detect SQL injection attacks. This novel prevention and detection technique is supported by a prediction model based on neural networks, which is configured for short-term predictions of intrusions. This mechanism uses a memory of cases which identifies past experiences with the corresponding indicators that characterize each of the attacks. This paper presents a novel classification system that combines the advantages of the CBR systems, such as learning and adaptation, with the predictive capabilities of a mixture of neural networks. SQLCBR agents are particularly suitable to be applied to classification problems in dynamic environments, such as the classification of SQL injection attacks, because they learn from past experiences and adapt to changes [31, 32]. The elements of the SQL query classification problem are represented as follows, using CBR terminology:

- Problem Description: Describes the initial situation (information available) before beginning with the classification process. As shown in Table 5, the problem description consists of a case identification, user session and SQL query elements.
- Solution: Describes the actions carried out in order to resolve the problem description. As shown, in Table 5, it contains the case identification and the applied solution.
- Final State: Describes the state achieved after the solution has been applied. It can take three possible values: attack when a SQL query is considered as malicious, not attack when the SQL query has been executed without malicious problems, or suspect when the SQL query has been executed but there still exist doubts about the real nature or intentions of the user. The multi-agent architecture incorporates the Manager agent, which allows an expert to evaluate the classification.

Table 5. Structure of the problem definition and solution for a 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		
Numer_And	Integer		
Numer_Or	Integer		
Number_literals	Integer		
Number_LOL	Integer		
Length_SQL_String	Integer		
Start_Time_Execution	Time		
End_Time_Execution	Time		
Query_Category	Integer		

The reasoning mechanism of the Anomaly agent is responsible for classifying the SQL database queries made by the users. When a user makes a new request, it is first checked by a pattern matching mechanism. This mechanism is based on a set of well-known patterns which are stored in a database that handles a significant number of signatures not allowed on the user level, such as symbol combination, binary and hexadecimal encoding, and reserved statement of language (union, execute, drop, revoke, concat, length, asc, chr etc.). If the FingerPrint agent detects a known signature using the pattern matching mechanism, the query is automatically identified as an attack and the classification process is complete. In order to identify other SQL attacks that do not match a known pattern, the Anomaly agent uses the CBR mechanism, which must have a memory of cases with the structure described in Table 5. The problem description for a case is obtained by a string analysis technique over the SQL query. This process can be easily understood through the following example: Let us suppose that a query with the following syntax: *"Select field1, field2, field3 from table1 where field1 =*

input1 and field2=input2” is received. If the fields input1 and input2 are used to bypass the authentication mechanism with the following input data: *“Select field1, field2, field3 from table1 where field1 =” or 9876 = 9876 -- ‘and field2=””*, then, the analysis of the SQL string would generate the result presented in Table 6, with the following fields: Affected_table(c1), Affected_field(c2), Command_type(c3), Word_GroupBy(c4), Word_Having(c5), Word_OrderBy(c6), Numer_And(c7), Numer_Or(c8), Number_literals(c9), Number_LOL(c10), Length_SQL_String(11), Query_Category(12). The fields Command_type and Query_Category have been encoded with the following nomenclature Command_Type: 0=select, 1=insert, 2=update, 3=delete; Query_Category: -1=suspicious, 0=illegal, 1=legal, 2=unassigned.

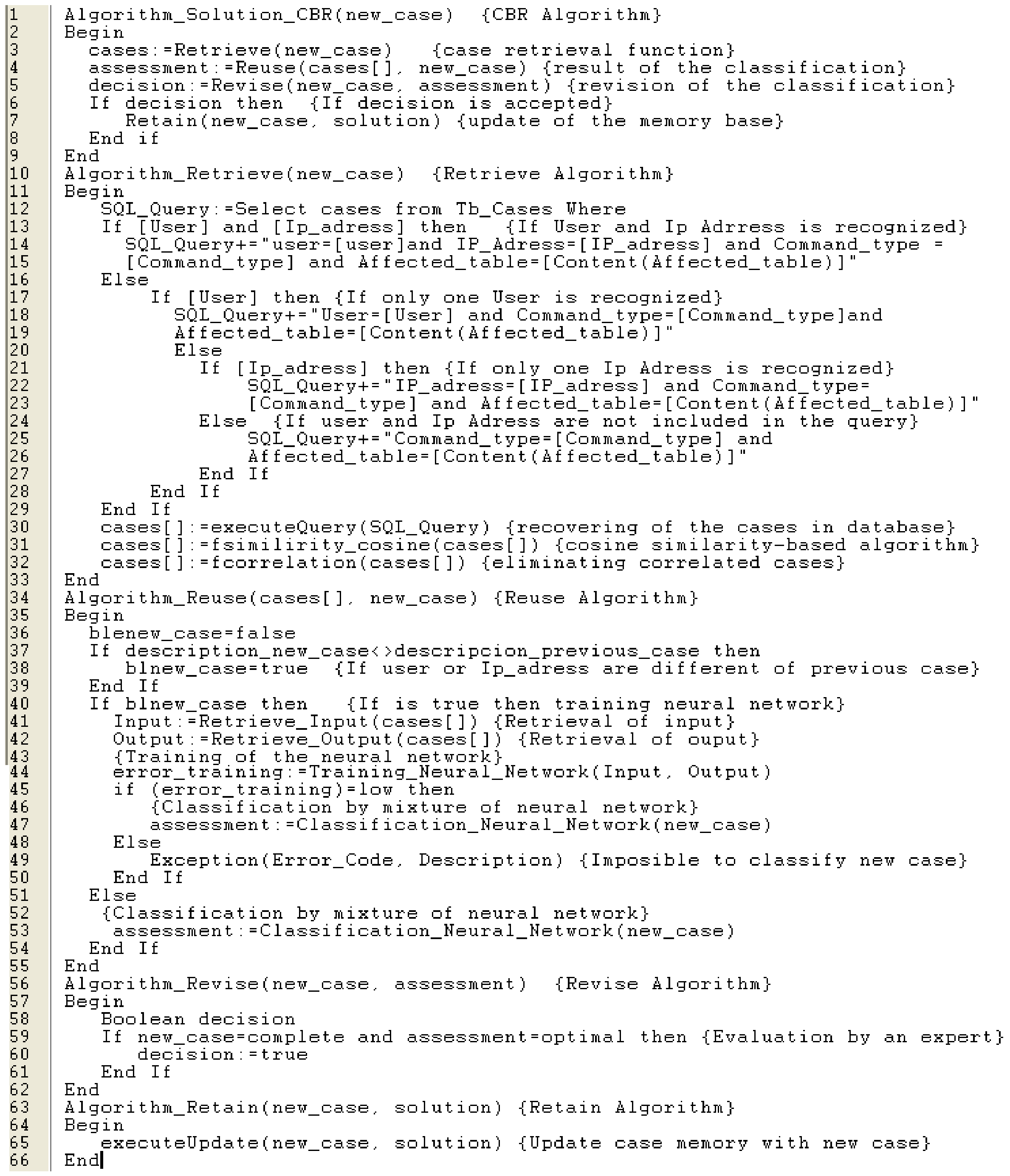
Table 6. SQL String transformed through the string analysis

c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12
1	3	0	0	0	0	1	1	2	1	81	0

When the Anomaly SQLCBR agent receives a new problem description, it initiates a new CBR cycle to achieve a solution. Figure 5 shows the algorithm containing the four steps of the CBR cycle:

- Retrieve: The first phase of the CBR cycle consists of recovering past experiences from the memory of cases, specifically those with a problem description similar to the current request. In order to do this, a cosine similarity-based algorithm is applied, allowing the recovery of those cases which are at least 90% similar to the current problem description.
- Reuse: The recovered cases are then input for the second phase of the CBR cycle, the reuse phase. In the reuse phase the similar cases recovered are used to train the mixture of neural networks. Once the mixture has been trained, it is able to classify the current problem. The result obtained using a mixture of the outputs of the networks provides a balanced response and avoids individual tendencies (always taking into account the weights that determine which of the two networks is more optimal). Because the mixture of neural networks is composed of two different networks, there are some considerations to take into account in the training process: (i) the neural network with neurons based on the sigmoidal function is trained with the recovered cases that were classified as attack or not attack, whereas the neural network with neurons based on hyperbolic function is trained with all the recovered cases (including those of identified as suspicious). (ii) Moreover, a preliminary analysis of correlations is required to determine the number of neurons of the input layer of the neuronal networks. The data used to train the mixture of networks must not be correlated. Avoiding correlated data allows network topologies to be reduced. After removing correlations, only the input variables that are not correlated to each other remain, which is reflected in a smaller number of neurons in the input layer, thus lowering the training time [19]. (iii) Additionally, it is necessary to normalize the data (i.e., all data must be values in the interval [0,1]) after deleting correlated cases. If data are not normalized, an overflow is produced in the outputs of neural networks, causing an error. The cause of the error is the use of the hyperbolic tangent and sigmoidal function as activation functions.
- Revise: An expert evaluates the solution proposed for those cases where the mixture of networks generates output values in the interval [-0.6,-0.4] [0.4,0.6]. The remaining cases are automatically stored.

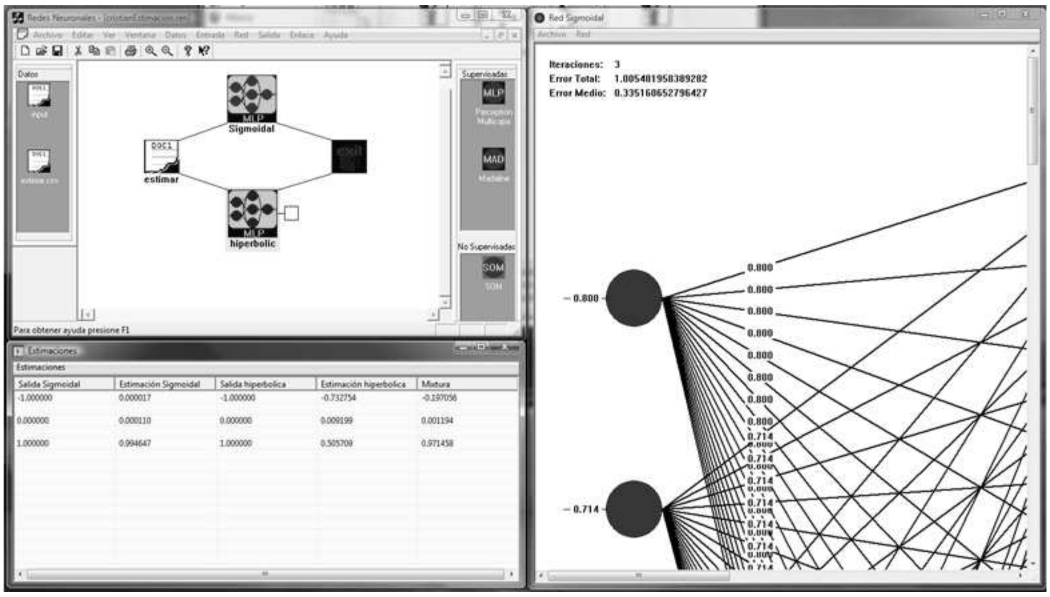
- Retain: The system learns from its own experiences from each of the previous phases and updates the database with the solution obtained in the query classification. All cases marked as efficient in the revise phase are stored.



```
Algorithm_Solution_CBR(new_case)  {CBR Algorithm}
Begin
  cases:=Retrieve(new_case)   {case retrieval function}
  assessment:=Reuse(cases[], new_case) {result of the classification}
  decision:=Revise(new_case, assessment) {revision of the classification}
  If decision then  {If decision is accepted}
     Retain(new_case, solution) {update of the memory base}
  End if
End
Algorithm_Retrieve(new_case)  {Retrieve Algorithm}
Begin
   SQL_Query:=Select cases from Tb_Cases Where
   If [User] and [Ip_adress] then   {If User and Ip Adrress is recognized}
     SQL_Query+="user=[user]and IP_Adress=[IP_adress] and Command_type =
     [Command_type] and Affected_table=[Content(Affected_table)]"
   Else
       If [User] then {If only one User is recognized}
         SQL_Query+="User=[User] and Command_type=[Command_type]and
         Affected_table=[Content(Affected_table)]"
         Else
           If [Ip_adress] then {If only one Ip Adress is recognized}
               SQL_Query+="IP_adress=[IP_adress] and Command_type=
               [Command_type] and Affected_table=[Content(Affected_table)]"
           Else  {If user and Ip Adress are not included in the query}
               SQL_Query+="Command_type=[Command_type] and
               Affected_table=[Content(Affected_table)]"
           End If
       End If
   End If
   cases[]:=executeQuery(SQL_Query) {recovering of the cases in database}
   cases[]:=fsimilirity_cosine(cases[]) {cosine similarity-based algorithm}
   cases[]:=fcorrelation(cases[]) {eliminating correlated cases}
End
Algorithm_Reuse(cases[], new_case) {Reuse Algorithm}
Begin
  blenew_case=false
  If description_new_case<>descripcion_previous_case then
     blnew_case=true  {If user or Ip_adress are different of previous case}
  End If
  If blnew_case then   {If is true then training neural network}
    Input:=Retrieve_Input(cases[]) {Retrieval of input}
    Output:=Retrieve_Output(cases[]) {Retrieval of ouput}
    {Training of the neural network}
    error_training:=Training_Neural_Network(Input, Output)
    if (error_training)=low then
       {Classification by mixture of neural network}
       assessment:=Classification_Neural_Network(new_case)
    Else
       Exception(Error_Code, Description) {Imposible to classify new case}
    End If
  Else
   {Classification by mixture of neural network}
    assessment:=Classification_Neural_Network(new_case)
  End If
End
Algorithm_Revise(new_case, assessment)  {Revise Algorithm}
Begin
   Boolean decision
   If new_case=complete and assessment=optimal then {Evaluation by an expert}
      decision:=true
   End If
End
Algorithm_Retain(new_case, solution) {Retain Algorithm}
Begin
   executeUpdate(new_case, solution) {Update case memory with new case}
End
```

FIGURE 5. Algorithm of the Cycle CBR for classifying SQL query.

As shown in Figure 5, an essential element in the classification algorithm executed by the Anomaly agent is the mixture of neural networks that is implemented in the reuse stage of the CBR cycle to predict attacks. The mixture uses two neural networks, both of which are multilayer perceptrons, but each of the networks uses a different type of activation function. The general idea of the mixture is that each of these networks obtains an individual solution for the problem by following a particular strategy. The solutions provided are then combined to find the optimal classification. The following paragraphs provide a detailed explanation of the internal structure of the mixture of neural networks, as well as the way it is used to classify SQL queries as *attack* or *not attack*. Figure 6 illustrates the structure for the mixture of the neural networks.



Iteraciones: 3
Error Total: 1.005481958389282
Error Medio: 0.335160652796427

Estimaciones

Salida Sigmoidal	Estimación Sigmoidal	Salida hiperbolica	Estimación hiperbolica	Mixtura
-1.000000	0.000017	-1.000000	-0.732754	-0.197056
0.000000	0.000110	0.000000	0.009199	0.001194
1.000000	0.994647	1.000000	0.505709	0.971458

FIGURE 6. Snapshot of the mixture of the neural networks.

As shown in Figure 6, the new case is presented to both neural networks at the same time, after which each of the neural networks will give its own decision regarding the classification. The neural network based on a sigmoidal function gives two results (illegal or legal) and the neural network that uses a hyperbolic tangential function produces three results (illegal, legal or suspicious). The learning algorithm for the neural networks and an explanation for the difference in each type of network are detailed in [19]. Because the mixture is composed of two multilayer perceptrons that use different activation functions, the learning algorithm has been particularized by taking both possible activation functions into account [19].

- The Sigmoidal activation function has its range of possible values at the interval $[0, 1]$. It is used to detect if the request is classified as an attack or not attack. The value 0 represents an illegal request (*attack*) and 1 a legal request (*non attack*). The Sigmoidal activation function is the most commonly used activation function for classifications between two groups. This function has the drawback of only placing classifications in two groups, that is:

$$f(x) = \frac{1}{1 + e^{-ax}} \tag{1}$$

  Where the value a=1 is used in the equation
- The hyperbolic tangential function has its range of possible values in the interval $[-1, 1]$. It is used to detect if the request is an *attack*, *not attack* or *suspicious*. The hyperbolic tangential function allows more possible cases than the sigmoidal function. The value 0 represents an illegal request, value 1 represents a legal request, and value -1 are suspicious requests. The hyperbolic tangential function is suitable for classifying the request into three groups. The hyperbolic tangential activation function is given by

$$f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2}$$

If only one network with a sigmoidal activation function is used, then the result provided by the network would tend to be *attack* or *not attack*, and no *suspicious* results would be detected. On the other hand, if only one network with a hyperbolic tangent activation is used, then a potential problem could exist in which the majority of the results would be identified as *suspicious* although they were clearly *attack* or *not attack*. The mixture provides a more efficient configuration of the networks since the global result is determined by merging two filters. This way, if the two networks classify the user request as an attack, so too will the mixture; and if both agree that it is not an attack, so will the mixture as well. If there is no concurrence, the system uses the result of the network with the least error in the training process or classifies it as a suspicious. In the reuse phase the two networks are trained by a back-propagation algorithm for the same set of training patterns (these neural networks are named Multilayer Perceptron), using a Sigmoidal activation function (which will take values in $[0, 1]$, where $0 =$ Illegal and $1 =$ legal) for a Multilayer Perceptron and a hyperbolic tangent activation function for the other Multilayer Perceptron (which take values in $[-1, 1]$, where $-1 =$ Suspect, $0 =$ illegal and $1 =$ legal).

The response of both networks is combined obtaining the mixture of networks. The networks that are combined should have the same number of neurons in output and input layers. In addition, it is necessary that the training set used in both networks be the same (same output and input data, and number of patterns), since the intention is to assess which of the two networks learns best from the training set. However, the number of neurons in the hidden layer can be variable. The steps required to mix the outputs of the neural networks are:

– Determine the best network for the output neuron (the network that provides a minimum average absolute error for the output neuron which is denoted by, where r indicates the network considered).
– Consider outputs of each network according to the good results obtained during the learning process.

To formalize the mix of networks, a new counter r is added to indicate the number of the network being considered.

$$E.A.M.S.^r = \frac{1}{2q} \sum_{p=1}^{q} |Y^{pr} - d^{pr}| \,; r = 1, 2 \tag{3}$$

Where Y^{pr}: is the output obtained by training pattern p of the network r, d^{pr}: is the desired output according to training pattern p of the network r, and q the number of training patterns. The networks are sorted from best to worst performance, which is valued through the rate of learning, that is, of least to greatest $E.A.M.S.^r$. Once sorted, the output obtained from merging both networks is calculated, resulting in a weighted function of the output from each network. The fact that the network has provided a better learning rate must be taken into account and positively weighted. The output obtained from the mixture of neural networks with output k for the Z networks is denoted as y^2, as is presented in (4), where $a = 1$ if the networks are previously sorted from best to worst.

$$y^2 = \frac{1}{\sum_{r=1}^{2} e^{-|a-r|}} \sum_{r=1}^{2} e^{-|a-r|} y^r \quad (4)$$

Compared to traditional techniques, neural networks are a good alternative for classification problems as evidenced by the empirical results presented in the results section. The reason for choosing a mixture of networks in classifying SQL injection attacks is that the use of neural networks alone is limited when it comes to making decisions. As a clear example of this type of problem, suppose that we approach the problem using a single neural network, for example one based on neurons with a sigmoidal activation function. In this case, the network could not conclude anything automatically and the intervention by a human expert would be required. However, if instead of considering just one network, we use a mixture of networks, the system is capable of solving this type of situation. Table 7 presents possible situations for the output of the neural network with a sigmoidal function and a value of 0.5. This network has the least error training of the two systems, and different cases of output are evaluated by the network with tangential hyperbolic activation function.

Table 7. Possible situations where the mixture of neural networks would not generate a solution

Sigmoidal Network Output	Hyperbolic Network Output	Mixtured Output
0.5	0.5	$(1/(1+Exp[1]))(0.5+0.5*Exp[1]) = 0.5$
0.5	-0.5	$(1/(1+Exp[1]))(0.5-0.5*Exp[1]) = -0.231059$
0.5	0.25	$(1/(1+Exp[1]))(0.5+0.25*Exp[1]) = 0.317235$
0.5	-0.25	$(1/(1+Exp[1]))(0.5-0.25*Exp[1]) = -0.0482939$
0.5	0.75	$(1/(1+Exp[1]))(0.5+0.75*Exp[1]) = 0.682765$
0.5	-0.75	$(1/(1+Exp[1]))(0.5-0.75*Exp[1]) = -0.413823$

As shown in Table 7, if the mixture is used, the only situation requiring intervention by a human expert is the one in which both networks give an output value of 0.5. The likelihood of this happening is $\frac{1}{11}$; applying the Laplace rule which tells us that the probability of the occurrence of an event is the quotient of favorable cases and possible cases. When working with one digit decimal numbers, there is one case in which the mixture cannot decide: when the value from among the 11 possible interval values $[0,1]$ is equal to 0.5. The probability that a network with hyperbolic activation function will take on a value of 0.5 is $\frac{1}{21}$ (there continues to be one case with one digit decimal numbers in which the mixture cannot decide from among the 21 possible values interval $[-1,1]$). So, the likelihood of needing the intervention by a human expert by using a mixture of the two networks, as compared to both networks individually would be: $\frac{1}{11} * \frac{1}{21} * \frac{1}{231} = 0.0043$

4.2. **SQLCBR Agent with Classification Capabilities.** This is a SQLCBR agent specially designed to predict the behavior of suspicious users. When a user makes requests that are seen as malicious or suspicious, it is very important for the system to have a mechanism to predict when and what type of request the user is going to make next. Such predictions allow the system to anticipate the actions of potentially dangerous users and block the attacks before they occur. The use of CBR mechanisms is especially suitable for this type of problem since it is possible to use past experiences to predict future behavior.

This paper proposes a strategy of a CBR cycle based on the use of Time Series at the reuse stage. This is a novel system for predicting behaviors that will improve the results that until now were only obtained with other existing methods. The reason is that each user identified as a hacker usually follows unique patterns of action, especially if he or she has already been successful with previous attacks. In other words, there is a high component of similarity with past experiences. Among the techniques that are currently used to predict the behavior that a hacker can engage in when executing SQL injection attacks are the Linear Regression Method [35], Decision Trees [38] and the Time Series [35]. Table 8 shows the efficiency of these methods based on 100 cases of SQL injection attacks. As seen in Table 8, the prediction based on Time Series and the prediction based on Decision Trees provide excellent results, whereas the results of the Linear Regression method show a much lower degree of success.

TABLE 8. Effectiveness of the prediction techniques

Method	Efficient(%)
Linear Regression	50
Tree Decision	95
Temporal Series	97

The CBR-BDI mechanism based on a time series analysis is responsible for predicting behaviour according to the requests made by users. When a user makes a new request, it is matched against well-known patterns of attack. If there is a match, the request is automatically identified as an attack. Additionally, the request is identified as an attack by the Forecaster agent in cases where there are several unsuccessful attempts at certain requests, which is known as a brute force attack. In order to identify the rest of the SQL attacks, the system uses CBR, which must have a memory of cases dating back at least 4 weeks, and store the following variables: User ID, Host ID & IP Address, Request, Completion (results of user requests, which returns a zero if successful, otherwise it returns the error code describing the reason for the failure), Valid Time (which has two values, startup time and time of completion). The structure of a case for the behavior prediction problem of users is shown in Table 9.

TABLE 9. Structure of a case for user behavior prediction problem

Problem Description	Solution: Prediction
User's log-queries	Next request to be made by the user
User profile	Estimated time to resolve the request
Current query (Case newly classified)	

As formerly indicated, in order to carry out a suitable prediction it is necessary to have previously stored the variables that make up the problem description during a period of at least 4 weeks. With these data stored, a multivariate time series is created and certain trends are eliminated (overall direction of the variable in the observation period), using the moving averages method. This method involves the replacement of one series by another series formed with the means of several values from the original series. The moving averages method of size n is used when each of the means of the new series created is equal to the means of n elements adjacent to the series. So, for example: $m(X_t) = \frac{X_{t-1}+X_t+X_{t+1}}{3}$, is the moving average for three points. If n is odd then only the first series of means is calculated. $m(X_t) = \frac{X_{t-1}+X_t+X_{t+1}}{3}$ is the moving average for three points. If n is even, then the second series of means is collected. In other words, $m(X_t) = \frac{(X_{t-2}/2)+X_{t-1}+X_t+X_{t+1}+(X_{t+2}/2)}{4}$,is the moving average for four points.

When a user executes a request, the multivariate time series informs us about what type of action to take, what it expects the following requests will be, and what the approximate time of the next action taken will be. This will allow us to take the necessary measures and anticipate movements made by the user. The basic idea in obtaining the time series is to take into account both the exact moment at which requests are made by users, as well as earlier requests made. Each new request that is executed has a direct correlation with the previous request, as well as with the response given by the system. Thus, the request made at the exact time $t+1$, depends on the request made at the exact time t and of the success or failure of the request made in time t. The time series takes the patterns stored in the database into account and applies the moving averages method. The time series labels the axis of the ordinate with the number of requests, the request related to time t, and the success or failure obtained with such request. To predict the next request, the series with the actions carried out by the user until that instant of time are recovered. Then a search is executed on the stored data in order to obtain the sequence of actions most likely to be carried out by the user. In this way, the time series can be represented as a function: Request, Completion (results of user requests, which return a zero if successful, otherwise an error code describing the failure is returned.) and Valid Time (which has two values, startup time and time of completion).

5. Case Study. A Medical Database.

A case study was proposed to test the effectiveness of a SCMAS prototype. The prototype was evaluated by a previously developed multi-agent system installed in a geriatric facility/care home [14]. The implemented multi-agent system has improved the security of the patients, facilitated care-giver activity and guaranteed an adequate level of efficiency. The system was developed in a distributed environment containing devices such as PDA, notebook computers and wireless internet access. A back-end database stores and supplies information. The database manager is MySQL. The participants, including nurses, doctors, patients, social workers and other employees can be seen in Figure 7. The medical staff in charge of patient care was comprised of 2 doctors, 10 nurses and 1 social worker. Thirty patients were being observed and attended to by the multi-agent system. Each nurse was equipped with a PDA, so a total of 10 PDAs were executing queries on the database during the work day. With these data, we prepared an attack scenario. The equipment necessary for performing the test included 2 workstations and 3 PDAs. The test was carried out over a period of 30 uninterrupted working days.

During the execution of the multi-agent system, several types of SQL queries were carried out on the database. The queries were related to patient treatments, scheduling the work day for the nurses, etc. Most of the queries were executed from PDAs. The PDAs are used by doctors and nurses to accomplish their tasks. To facilitate the evaluation of the prototype, we focused on the nurse role. A main volume of queries was generated each time a plan was assigned to a nurse. The plans changed for different reasons during their execution and these changes increased the number of queries on the database. When nurses start and finish a task, they send a response through a SQL query. The nurses have direct access to the database system through the application interface on their PDAs. The strategy followed in the case study was based on the execution of queries crafted from 2 attack PDAs. These PDAs were installed with a user interface similar to the nurses' PDAs, but the two attack PDAs are capable of executing tainted queries. When a query is executed from the attack PDA, it carries out a type of SQL injection that has to be captured, analyzed and classified as legal, illegal or suspicious according to our

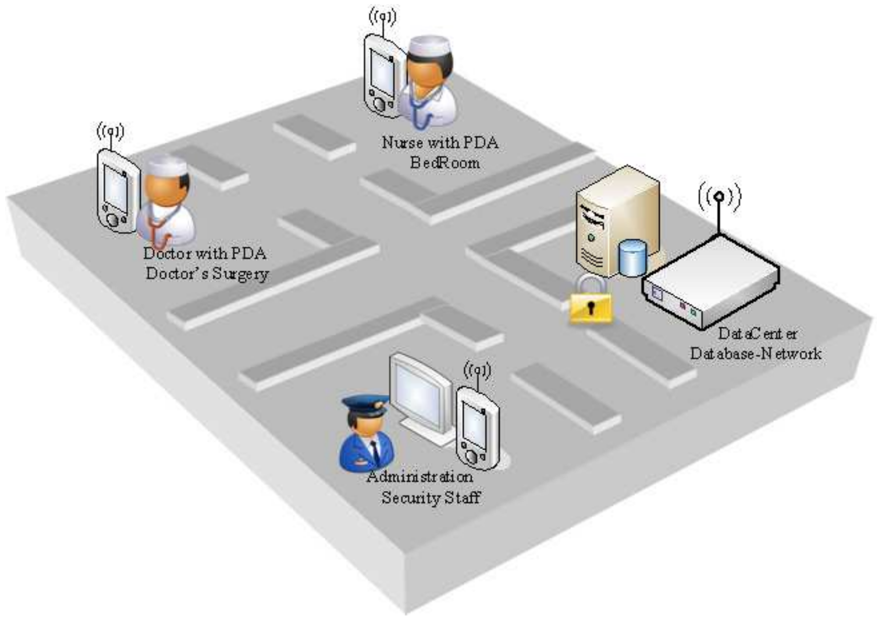


Figure 7. Abstract scenario of the real environment (Geriatric Care Home).

parameters. The FingerPrint agents and Anomaly SQLCBR agents were distributed in the 2 workstations. Because the test was carried out on a real medical database, a special mechanism was built to guarantee the integrity of the database. All the queries executed both by the nurses' PDAs and the attack PDAs were examined and classified. The test was conducted with a total of 12 PDAs, 10 PDAs assigned to the active nurses and 2 PDAs to execute attacks, and a total of 10, 200 queries were sent to the medical database. Each nurse's PDA executed approximately 30 daily queries, and during the 30 days of the test, 9, 000 legal queries were carried out. Each of the two attack PDAs executed 20 illegal queries daily. These PDAs sent 40 attacks during a work day. Throughout the 30 day test period, a total of 1, 200 attacks targeted the medical database. The volume of queries during the test period allowed us to build a case memory to validate the proposed strategy.

The following example explains the classification process for the SQL queries in SCMAS. Let us assume that a nurse uses her personal identification number and password to log into the system from a PDA, whereby the following SQL string is used to place the request:

SELECT IdNurse, FirstName, SecondName, BirthDate, Age, Sex, SInsurance, salary, BAccount FROM TBNurses WHERE strIdNurse='PA0012' AND strPassword= 'ONeil15'

The string is attacked and the resulting SQL code is:

SELECT strIdNurse, FirstName, SecondName, BirthDate, Age, Sex, SInsurance, salary, BAccount from TBNurses where strIdNurse=" OR 'abcd'='abcd' –' AND StrPassword="

Let us assume that the first filter applied by the FingerPrint agent did not detect a malicious code and the query continues to the Anomaly agent for its classification in the next layer. Applying a syntactic analysis on the text string for the SQL query, the following values, as listed in the table 10, would be generated.

Following the procedure as explained in section 4.1, the correlation data are eliminated and the initial data from the neural network are normalized within the range $[0, 1]$. Table 11 presents the results of the normalization of the values for the new case.

Table 10. Effectiveness of the prediction techniques

Fields	Values	IdField
Affected_table	1	c1
Affected_field	9	c2
Command_type	0	c3
Word_GroupBy	0	c4
Word_Having	0	c5
Word_OrderBy	0	c6
Numer_And	1	c7
Numer_Or	1	c8
Number_literals	4	c9
Number_LOL	1	c10
Length_SQL_String	165	c11
Query_Category	2	c12

Table 11. Description of the new normalized case

c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12
0	0	0	0	0	0	0	1	$0,4$	$0,5$	$0,036$	0

To complete this process, the Anomaly agent executes a CBR cycle in order to classify the new SQL string.

- A cosine similarity algorithm is applied in order to recover the cases that have a 90% similarity rate to the new case. Table 12 lists the similar cases (SQL strings) that were recovered from the memory of cases.

Table 12. Structure of a case for user behavior prediction problem

SELECT strIdNurse, FirstName, SecondName, BirthDate, Age, Sex, SInsurance, salary, BAccount FROM TBPatients where strIdNurse=" OR 1=1 --' AND StrPassword="
SELECT strIdNurse, FirstName, SecondName, BirthDate, Age, Sex, SInsurance, salary, BAccount FROM TBPatients where strIdNurse='mysql.user' --' AND StrPassword="
SELECT strIdNurse, FirstName, SecondName, BirthDate, Age, Sex, SInsurance, salary, BAccount FROM TBPatients where strIdNurse=" OR 2=2 --' AND StrPassword="
SELECT strIdNurse, FirstName, SecondName, BirthDate, Age, Sex, SInsurance, salary, BAccount FROM TBPatients where strIdNurse=" OR 1=2 --' AND StrPassword="
SELECT strIdNurse, FirstName, SecondName, BirthDate, Age, Sex, SInsurance, salary, BAccount FROM TBPatients where strIdNurse=" OR '111'='111' '-- AND StrPassword="

The description of the cases that have been recovered and normalized with the corresponding similarity measure is shown in Table 13. The results fall within the range $[0, 1]$ where the larger the value, the more similar the recovered case is to the new case.

- In the second phase of the CBR cycle both of the combined networks are trained with cases that were recovered from the first phase of the CBR cycle. Table 14 shows the results obtained during the training period for the 5 most similar cases, and the estimate for the new case that has been classified.
- The solution is evaluated in the third phase of the CBR cycle. The exit value for the new case (0.278) indicates that the SQL string from the user request is classified as

Table 13. Description of cases recovered and normalized with the similarity measure corresponding to the new case

c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	Measure of Similarity
0	0	0	0	0	0	0	1	0, 4	0, 5	0, 036	0	0, 858
0	0	0	0	0	0	0, 5	0	0	0	0, 072	0	0, 370
0	0	0	0	0	0	0, 5	1	0, 4	0, 5	0, 036	0	0, 930
0	0	0	0	0	0	0, 5	1	0, 4	0, 5	0, 036	0	0, 930
0	0	0	0	0	0	0, 5	1	0, 4	0, 5	0, 133	0	0, 933

Table 14. Classification obtained by the mixture of neural networks shown in Table 12, and the estimate for the new case

Case	Result
1	0, 231
2	0, 298
3	0, 211
4	0, 223
5	0, 214
New Case	0, 278

an attack and does not require evaluation by an expert since it falls outside of the given interval.

- Finally, an expert assigns an efficiency rate of 98% in the learning phase and the case is stored in the memory of cases to be used in future situations.

Figure 8 provides a graphical representation of the CBR phase, showing the entry and exit data and the tasks that were completed in each phase.

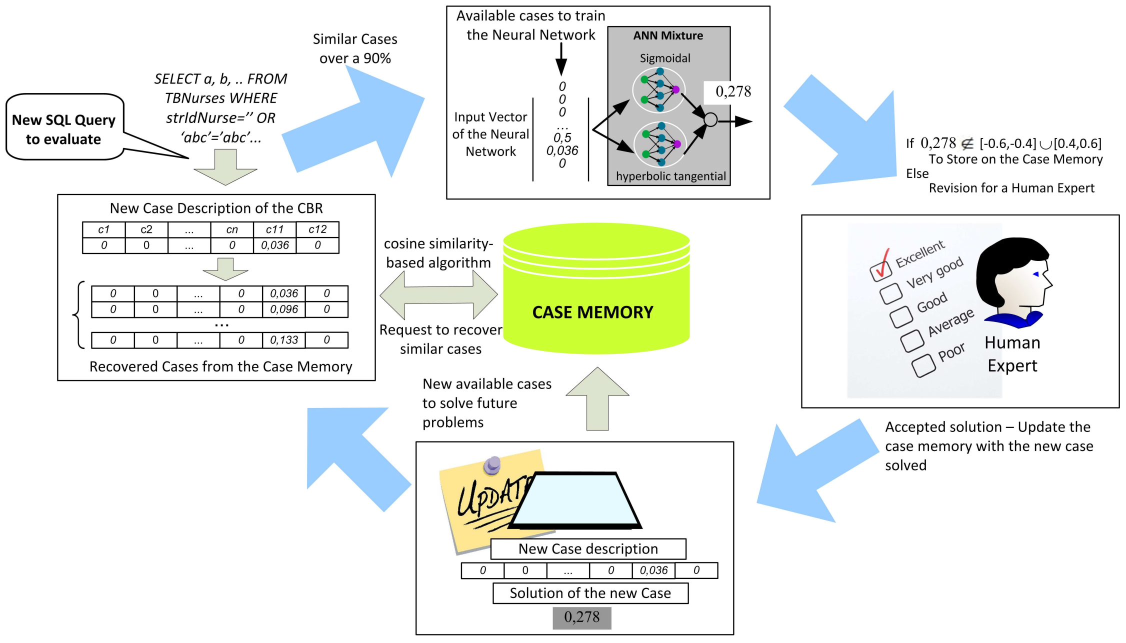

New SQL Query to evaluate
SELECT a, b, .. FROM TBNurses WHERE strIdNurse='' OR 'abc'='abc'...
Similar Cases over a 90%
Available cases to train the Neural Network
ANN Mixture
Sigmoidal
hyperbolic tangential
0,278
Input Vector of the Neural Network
0 0 0 ... 0,5 0,036 0
If 0,278 ∉ [-0.6,-0.4] ∪ [0.4,0.6]
To Store on the Case Memory
Else
Revision for a Human Expert
New Case Description of the CBR
c1 c2 ... cn c11 c12
0 0 ... 0 0,036 0
0 0 ... 0 0,036 0
0 0 ... 0 0,096 0
...
0 0 ... 0 0,133 0
Recovered Cases from the Case Memory
cosine similarity-based algorithm
Request to recover similar cases
CASE MEMORY
Excellent
Very good
Good
Average
Poor
Human Expert
New available cases to solve future problems
Accepted solution – Update the case memory with the new case solved
UPDATE
New Case description
0 0 ... 0 0,036 0
Solution of the new Case
0,278

Figure 8. Representation of the CBR cycle within the Anomaly agent

The next section presents the results obtained after executing the test.

6. **Results and Conclusion.**

The problem of SQL injection attacks on databases supposes a serious threat against information systems. This paper has presented a novel solution based on a new hierarchical multi-agent architecture for detecting SQL injection attacks. This solution combines the advantages of multi-agent systems, such as autonomy and distributed problem solving, with the adaptation and learning capabilities of CBR systems. Because current approaches are based on centralized strategies [26, 29, 30], the architecture proposed in this study offers a novel perspective in the detection and prediction of SQL injection attacks. The SCMAS architecture provides a hierarchical, distributed structure, which allows a more efficient balance and distribution of the tasks involved in the problem of detecting, classifying, blocking and predicting malicious SQL injection attacks on databases. In addition, this paper has presented two interesting mechanisms for improving the classifications of SQL attacks and the prediction of attacks from malicious users. Both mechanisms were implemented through SQLCBR agents, a special type of CBR-BDI agent [16] which demonstrates a great capacity for learning and adaptation. The SQLCBR Anomaly agent is a classifier agent that, based on the philosophy of the case-based reasoning mechanisms [17], proposes a new strategy that uses past experiences to classify SQL injection attacks. This strategy differs in its conception from other current strategies and, moreover, incorporates the prediction capabilities that characterize neural networks. The Anomaly agent integrates an innovative model into its CBR cycle consisting of a mixture of neural networks that provides a significant reduction in the error rate during the classification of attacks, and improves the efficiency of the current methods. The second type of SQLCBR agent is the Forecaster agent, which incorporates a mechanism based on time series analysis into the reuse stage of its CBR cycle in order to predict the behaviour of the malicious users, thus allowing preventive actions to minimize possible attacks on the database.

To check the validity of the proposed model, we elaborated a series of tests which were executed on a memory of cases, specifically developed for these tests, which generated attack consults. The results obtained are promising, improving in many cases those obtained with other current techniques, which allows us to conclude that SCMAS can be considered a good alternative for the detection and prediction of SQL injection attacks. The tests were conducted in the following way: first, we evaluated the efficiency of the classification and prediction methods proposed in this research and compared the results obtained to alternative techniques. Then, we evaluated the global efficiency of the architecture by comparing different meaningful parameters before and after the implementation of the system in the test environment. The following paragraphs describe the experiments and discuss the conclusions obtained.

The classification system integrated within the Anomaly agent provided the results shown in Table 15, which are promising: it is possible to observe different techniques for predicting attacks at the database layer and the errors associated with misclassifications. All the techniques presented in Table 15 have been applied under similar conditions to the same set of cases, taking the same problem into account in order to obtain a new case common to all the methods. Note that the technique proposed in this article provides the best results, with an error in only 0.537% of the cases.

As shown in Table 15, the Bayesian method is the most accurate statistical method since it is based on the likelihood of the events observed. It has the disadvantage of determining the initial parameters of the algorithm, although it is the fastest of the statistical methods. After taking the errors obtained with the different methods into account, the regression models follow both the neural networks and the Bayesian methods. Because of the non linear behaviour of the hackers, linear regression offers the worst results, followed by the polynomial and exponential regression methods. This can be explained by looking at

Table 15. Results after testing different classification techniques

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

hacker behaviour: as hackers break security measures, the time they have for obtaining information via their attacks decreases exponentially. The empirical results show that the best methods are those that involve the use of neural networks, and if we consider a mixture of two neural networks, the predictions are notably improved. These methods are more accurate than statistical methods for detecting attacks to databases because the behaviour of the hacker is not linear, but dynamic and chaotic.

The advantage of using a mixture of neural networks not only improves performance provided by other classification techniques, but also improves performance that only neural networks can provide. The mixture has the advantage of reducing the number of cases in which the classifier agent cannot make decisions, thus requiring human intervention in only a few cases. We were able to check the decision of the mixture of networks with that of the human expert for those cases in which a single network did not decide, and found that both the mixture of networks and the human expert were in agreement in 99% of the cases. Figure 9 shows the effectiveness in the classification for two individual networks with a distinct activation function, and the effectiveness of the mixture of networks.

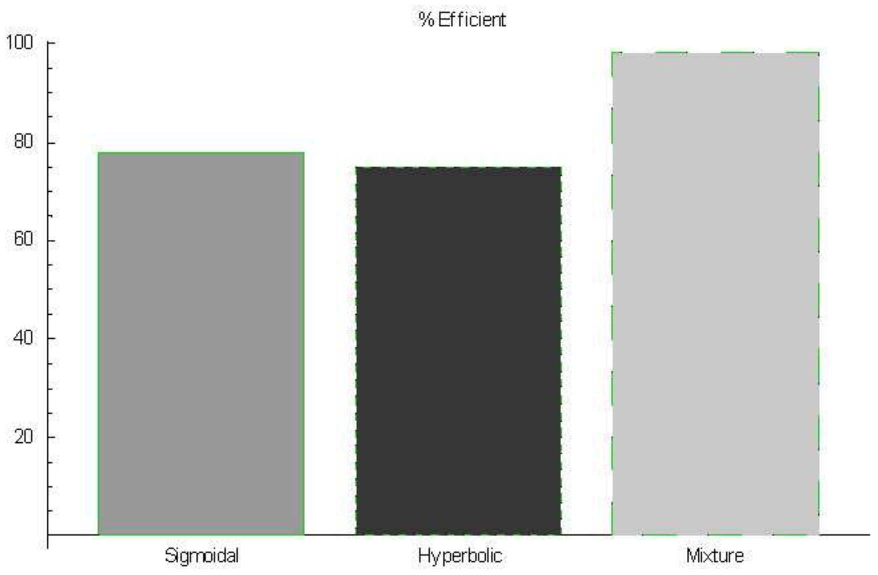


Figure 9. Effectiveness in the individual classification of networks and the mixture of networks.

The prediction mechanism integrated within the Forecaster agent also provided good results, as can be seen in Table 16. Looking at the behaviour of the time series analysis, we have observed that when the number of training patterns for the neural network increases, prediction error decreases. It is important to note that the number of training patterns is the result of applying filters such as the similarity-based algorithm and the correlation function. These filters meaningfully reduce the quantity of cases and allow improved performance during the training stage. The graph in Figure 10 indicates the success of the predictions with regards to the number of training patterns presented in Table 16.

Table 16. Successful (%) depending on number of training patterns

Number of patterns of training	Successful (%)
1000	99.5
900	99.1
700	98.5
500	98.6
300	96.8
100	89

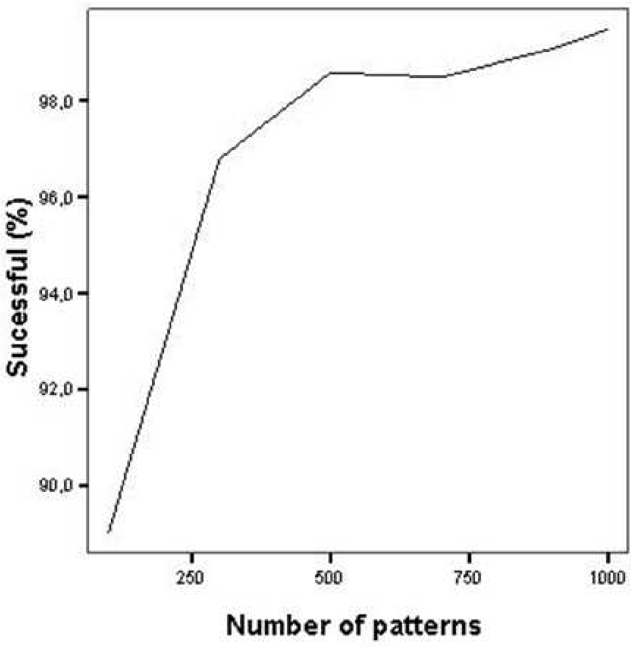


Figure 10. Sucessful (%) vs. Number of patterns

Figure 10 shows the percentage of predictions with regards to the number of patterns in the training phase. It is clear that with a large number of training patterns the percentage of successful predictions improves. Since we are working with CBR systems, which depend on large amounts of data stored in the memory of cases for each user, the percentage of successful predictions increases, as demonstrated in Figure 10. CBR systems need initial information (past experiences) to generalize efficient results. In addition, the time series analyses also need the data that is passed on in order to allow reliable predictions. Figure 11 shows that a period of 4 weeks implies an acceptable threshold to carry out predictions with a high rate of success.

In order to analyze the impact of multi-agent architecture proposed in the context of this investigation, we have evaluated the percentage of attacks detected before and after implementing the system, as shown in Figure 12. Figure 12 shows a progressive increase in the percentage of attacks detected successfully over time. As expected, the system does not initially provide a high detection rate, since it works with synthetic data. But

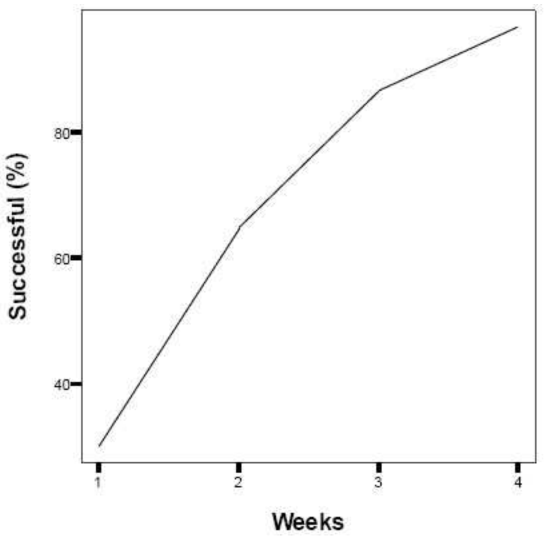


Figure 11. Success prediction according to the number of weeks from the database

as time passes, the system learns and adapts to the environment in which it is located. Thus, with four weeks of stored data, it is possible to obtain an attack detection rate above 87%.

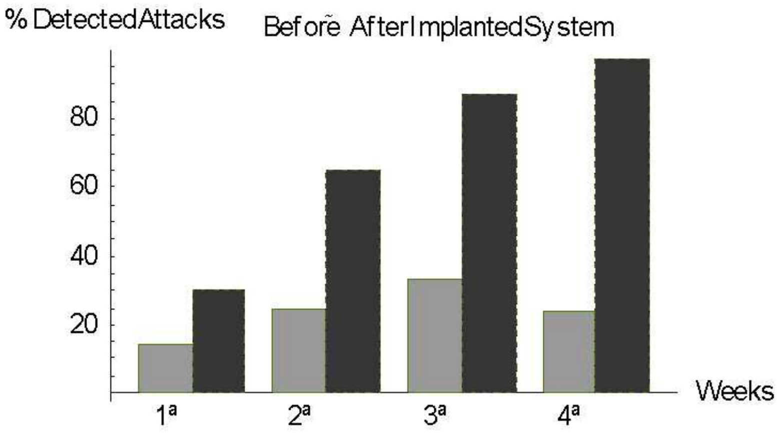


Figure 12. Progressive increase in the detection of attacks based on time.

Another meaningful indicator for evaluating the classification system of SQL injection attacks is the percentage of false positives identified by the system. This is an important parameter, because classifying queries as attacks when they are not so can cause serious delays in the system and discomfort to users. The percentages of false positives caused before and after implementing the system are shown in Figure 13. As shown, the SCMAS architecture has adapted itself gradually over time. The results obtained during the first three weeks exceeded the percentage of error obtained without the implementation of the system. However, after the fourth week the SCMAS system learns and reduces the rate of false positives.

As mentioned in subsection 4.1, user requests can be classified as attack, not attack or suspicious. The classification of suspected attacks reduces the success rate of the system and requires intervention by a human expert. That is why we have evaluated the percentage of suspicious requests before and after implementing the system. Figure 14 shows the percentage of suspicious requests identified by SCMAS compared to those

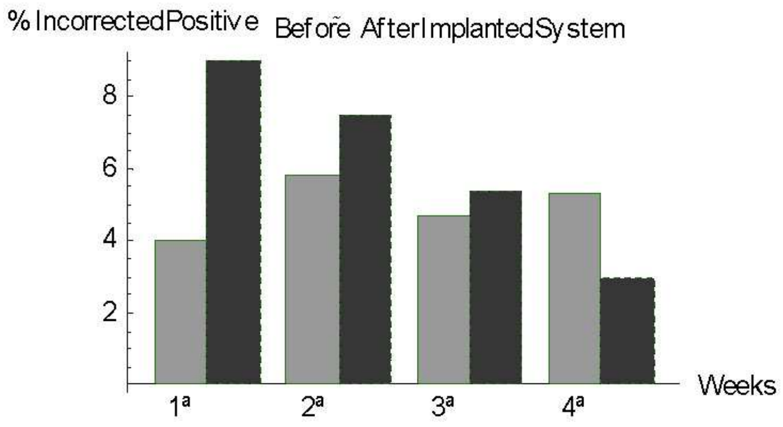


Figure 13. Percentage of false positives caused before and after the introduction of SCMAS.

identified by a human expert. As shown in Figure 14, the SCMAS system provides rates of 36.8% initially, but this error rate decreases quickly, falling to 4.3% in the fourth week of operation.

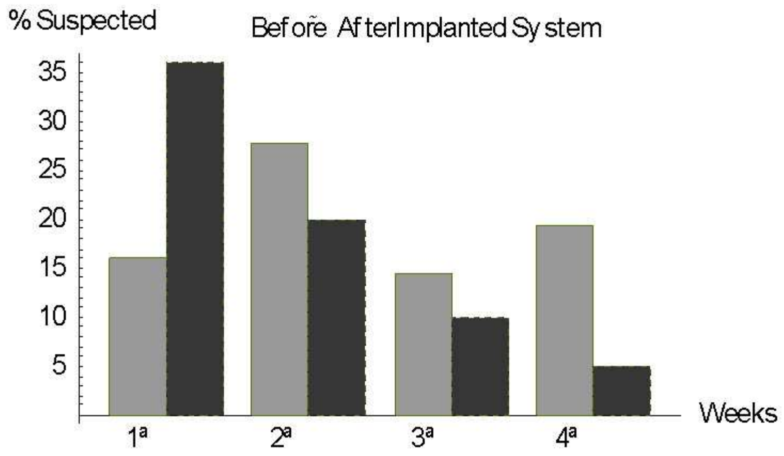


Figure 14. Percentage of suspicious requests before and after implemented SCMAS system.

The architecture presented in this paper provides a novel strategy for detecting SQL injection attacks. The results are promising and allow us to conclude that the SCMAS architecture considerably improves results provided by current technologies. SQLCBR agents are well-suited to the prediction and classification tasks, and improve current techniques. However, there is still much work to do, especially checking the validity of our architecture in heterogeneous real environments. Moreover, some alternatives to optimize the effectiveness in the retrieval and reuse stages will be considered [33, 45]. These are our next challenges.

Acknowledgment. This work has been supported by the Spanish Ministry of Science and Technology project TIN2006-14630-C03-03, the JCyL SA071A08 project and The Professional Excellence Program 2006-2010 IFARHU-SENACYT-Panama.The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] A. Aamodt and E. Plaza, Case-based reasoning: foundational issues, methodological variations, and system approaches, *AI Communications*, vol.7, no.1, pp.39-59, 1994.

[2] A. Abraham, R. Jain, J. Thomas and S. Y. Han, D-SCIDS: Distributed soft computing intrusion detection system, *Journal of Network and Computer Applications*, vol.30, no.1, pp.81-98, 2007.

[3] J. Bajo, V. Julin, J. M. Corchado, C. Carrascosa, Y. De Paz, V. Botti and J. F. De Paz, An execution time planner for the ARTIS agent architecture, *Engineering Applications of Artificial Intelligence*, vol.21, no.5, pp.769-784, 2008.

[4] J. Bajo, J. F. De Paz, D. Tapia and J. M. Corchado, Distributed Prediction of Carbon Dioxide Exchange Using CBR-BDI Agents, *International Journal of Computer Science*, pp.16-25, 2007.

[5] J. Bajo, A. De Luis, A. Gonzalez, A. Saavedra and J. M. Corchado, A Shopping Mall Multiagent System: Ambient Intelligence in Practice. *In: 2nd International Workshop on Ubiquitous Computing & Ambient Intelligence*, pp.115-125, 2006.

[6] F. Bergenti and A. Poggi, LEAP: A FIPA Platform for Handheld and Mobile Devices, *In: Intelligent Agents VIII*, Springer-Verlag, London, UK, pp.436-446, 2002.

[7] E. Bertino and R. Sandhu, Database Security-Concepts, Approaches, and Challenges. *In: IEEE Transactions on Dependable and Secure Computing*, IEEE Computer Society, Los Alamitos, CA, USA, pp.2-19, 2005.

[8] S. W. Boyd and A. D. Keromytis, SQLrand: Preventing SQL Injection Attacks. *In: Applied Cryptography and Network Security*, pp.292-302, 2004.

[9] M. E. Bratman, D. J. Israel and M. E. Pollack, Plans and Resource-Bounded Practical Reasoning. *In: Philosophy and AI: Essays at the Interface*, The MIT Press, pp.349-355, 1988.

[10] B.C. Chaib-draa and F. Dignum, Trends in agent communication language, *Computational Intelligence*, vol.18, pp.89-101, 2002.

[11] Z. Chen, H. Wang, A. Abraham, C. Grosan, B. Y. Y Chen, and L. Wang, Improving Neural Network Classification Using Further Division of Recognition Space, *International Journal of Innovative Computing, Information and Control*, vol.4, 2008.

[12] A. S. Christensen, A. Mller and M. I. Schwartzbach, Precise Analysis of String Expressions, *In: 10th International Static Analysis Symposium*, Springer-Verlag, pp.1-18, 2003.

[13] W. R. Cook and S. Rai, Safe query objects: statically typed objects as remotely executable queries, *In: 27th international conference on Software engineering*, ACM, New York, USA, pp.97-106, 2005.

[14] J. M. Corchado, M. Glez-Bedia, Y. De Paz, J. Bajo and J. F. De Paz, Replanning Mechanism for Deliberative Agents in Dynamic Changing Environments. *Computational Intelligence*, vol.24, pp.77-107, 2008.

[15] J. M. Corchado, J. Bajo, Y. De Paz and D. Tapia, Intelligent environment for monitoring Alzheimer patients, agent technology for health care, *Decision Support Systems* 44(2), pp.382-396, 2007.

[16] J. M. Corchado, J. Pavón, E. S. Corchado and L. F. Castillo, Development of CBR-BDI Agents. *In: Advances in Case-Based Reasoning*, Springer Berlin / Heidelberg, 2004.

[17] J. M. Corchado and R. Laza, Constructing deliberative agents with case-based reasoning technology, *International Journal of Intelligent Systems*, vol.18, pp.1227-1241, 2003.

[18] A. Damba and S. Watanabe, Hierarchical Control in a Multiagent System, *International Journal of Innovative Computing Information and Control*, vol.4, no.12, pp.3091-3100, 2008.

[19] Y. De Paz, Mixture of Weibull distributions by means of Artificial Neural Networks with censored data, PhD thesis, Salamanca University, 2008.

[20] FIPA, Foundation for Intelligent Physical Agents, available: http://www.fipa.org, [Accessed 15 August 2007], 2007.

[21] D. Florescu, A. Levy, and A. Mendelzon, Database techniques for the World-Wide Web: a survey. *ACM SIGMOD Record*, vol.27, no.3, pp.59-74, 1998.

[22] V. H. Garcia, R. Monroy, and M. Quintana, Web Attack Detection Using ID3. *In: IFIP International Federation for Information Processing*, Springer, vol.218, pp.323-332, 2006.

[23] M. P.Georgeff and A.L. Lansky, Reactive Reasoning and Planning, *In: Advancement of Artificial Intelligence*, American Association of Artificial Intelligence, pp.677-682, 1987.

[24] C. Gould, Z. Su and P. Devanbu, JDBC Checker: A Static Analysis Tool for SQL/JDBC Applications. *In: 26th International Conference on Software Engineering*, IEEE Computer Society, Washington, DC, USA, pp.697-698, 2004.

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

[26] W. G. Halfond and A. Orso, AMNESIA: Analysis and Monitoring for NEutralizing SQL-injection attacks. *In: 20th IEEE/ACM international Conference on Automated software engineering*, ACM, New York, USA, pp.174-183, 2005.

[27] Z. Hayat, J. Reeve and C. Boutle, Ubiquitous security for ubiquitous computing. Information Security Technical Report, vol.12, no.3, pp.172-178, 2007.

[28] D. R. Hernandez, Introduction to the Bayesian Analysis, OceanDocs, 2007

[29] Y. W. Huang, S. K. Huang, T. P. Lin and C. H. Tsai, Web application security assessment by fault injection and behavior monitoring, *In: 12th international conference on World Wide Web*, ACM, New York, USA, pp.148-159, 2003.

[30] Y. Kosuga, K. Kono, M. Hanaoka, M. Hishiyama and Y. Takahama, Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection, *In: 23rd Annual Computer Security Applications Conference*, IEEE Computer Society, pp.107-117, 2007.

[31] Y. Li, S. Shiu and S. Pal, Combining feature reduction and case selection in building CBR classifiers, *IEEE Transactions on Knowledge and Data Engineering*, vol.18, no.3, pp.415-429, 2006.

[32] Y. Li, S. K. Shiu, S. Pal and J. K. Liu, A rough set-based CBR approach for feature and document reduction in text categorization, *In: Third International Conference on Machine Learning and Cybernetics*, pp.2438-2443, 2004.

[33] C. Liu, New Evolutionary Algorithm for Multi-objective Constrained Optimization. *ICIC Express Letters*, vol.2, no.4, pp.339-344, 2008.

[34] W. L. Low, J. Lee and P. Teoh, DIDAFIT: Detecting Intrusions in Databases Through Fingerprinting Transactions, *In: Databases and Information Systems Integration*, pp.121-128, 2002.

[35] Q. Martín, M. Cabero and Y. De Paz, Statistical treatment of data with SPSS. Resolved and commented practices, Thomson, 2007.

[36] R. A. McClure and I. H. Krger, SQL DOM: compile time checking of dynamic SQL statements, *In: 27th international conference on Software engineering*, ACM, New York, USA, pp.88-96, 2005.

[37] S. Mukkamala, A. H. Sung, A. Abraham, Intrusion detection using an ensemble of intelligent paradigms *Journal of Network and Computer Applications*, vol.28, no.2, pp.167-182, 2005.

[38] J. Quinlan, R. L. Rivest, Inferring decision trees using the minimum description length principle, *Information and Computation*, vol.80, no.3, pp.227-248, 1989.

[39] E. Rescorla and A. Schiffman, The Secure HyperText Transfer Protocol, RFC Editor, United States, http://www.rfc-editor.org/rfc/rfc2660.txt, 1999

[40] M. R. Rieback, P. N. Simpson, B. Crispo and A. S. Tanenbaum, RFID malware: Design principles and examples, *Pervasive and Mobile Computing*, vol.2, no.4, pp.405-426, 2006

[41] F. Rietta, Application layer intrusion detection for SQL injection. *In: 44th annual Southeast regional conference*, ACM, New York, USA, pp.531-536, 2006.

[42] J. Skaruz and F. Seredynski, Recurrent neural networks towards detection of SQL attacks, *In: IPDPS'07: Parallel and Distributed Processing Symposium*, IEEE International, pp.1-8, 2007.

[43] T. Uno, H. Katagiri and K. Kato, An Evolutionary Multi-Agent Based Search Method for Stackelberg Solutions of Bilevel Facility Location Problems *International Journal of Innovative Computing, Information and Control*, vol.4, no.5, pp.1033-1043, 2008.

[44] F. Valeur, D. Mutz and G. Vigna, A Learning-Based Approach to the Detection of SQL Attacks. *In: Intrusion and Malware Detection and Vulnerability Assessment*, Springer Berlin/Heidelberg, pp.123-140, 2005.

[45] Y. Wang, Fuzzy Clustering Analysis by Using Genetic Algorithm. *ICIC Express Letters*, vol.2, no.4, pp.331-337, 2008.

[46] G. Wassermann and Z. Su, An Analysis Framework for Security in Web Applications, *In: FSE Workshop on Specification and Verification of Component-Based Systems*, pp.70-78, 2004.

[47] M. Woolridge and M.J.Wooldridge, Introduction to Multiagent Systems, John Wiley & Sons, Inc., New York, USA, 2002.