SiC: An Agent based Architecture for Preventing and Detecting Attacks to Ubiquitous Databases

Cristian Pinzón, Yanira De Paz, Javier Bajo, Ajith Abraham and Juan M. Corchado

Abstract One of the main attacks to ubiquitous databases is the SQL injection attack, which causes severe damages both in the commercial aspect, as in the user's confidence. This Chapter proposes the SiC architecture as a solution to the SQL injection attack problem. This is a hierarchical distributed multiagent architecture, which involves an entirely new approach with respect to existing architectures for the prevention and detection of SQL injections. SiC incorporates a kind of intelligent agent, which integrates a case-based reasoning system. This agent, which is the core of the architecture, allows the application of detection techniques based on anomalies as well as detection techniques based on patterns, providing a great degree of autonomy, flexibility, robustness and dynamic scalability. The characteristics of the multiagent system allow an architecture to detect attacks from different types of devices, regardless of the physical location. The architecture has been tested on a medical database, guaranteeing safe access from various devices such as PDAs and notebook computers.

Key words: SQL injection, Security database, Intrusion Detection System, Multiagent, Case based Reasoning

Cristian Pinzón
University of Salamanca, Plaza de la Merced s/n, 37008, Salamanca, Spain, e-mail: cristian_ivanp@usal.es

Yanira De Paz
University of Salamanca, Plaza de la Merced s/n, 37008, Salamanca, Spain e-mail: yanira@usal.es

Javier Bajo
University of Salamanca, Plaza de la Merced s/n, 37008, Salamanca, Spain e-mail: jbajope@usal.es

Ajith Abraham
Norwegian University of Science and Technology e-mail: ajith.abraham@ieee.org

Juan M. Corchado
University of Salamanca, Plaza de la Merced s/n, 37008, Salamanca, Spain, e-mail: corchado@usal.es

1 Introduction

New technologies have provided ubiquitous working environments without time and location constraint. Nowadays, users handle several mobile devices such as a notebook computers, PDAs or intelligent phones. These devices manage information in immediate way, independently of the physical location and time instant. These new computing environments are supported by the growth of the network computing, especially wireless networks [26]. Furthermore, it is necessary taking into account a distributed database in a strategic mode. Databases provide information to user applications on the different devices.

Information systems are based on a back-end database system. The database is a critical piece both in daily operations as on decision making [10]. Information systems have great impact in each aspect of the daily life (e.g. bank accounts registers, medical registers, retirements, payrolls, phone registers, tax registers, vehicle registers, supermarket purchases, school registers). Any meaningful data of the daily life is stored on a database system [32]. Due to this situation, very often the databases are a target of great number of attacks. The current solutions have been unable to provide enough confidentiality and integrity of the stored data. Firewalls, Intrusion Detection System (IDS), antivirus software and other security measures are limited and cannot protect of new treats and zero day attacks.

The effort carried out in order to detect and stop the attacks targeted to the information systems, seems to be not very sound. However, the cyber attack problem acquires more importance if recent technologies are taken into account. Nowadays the users use mobile devices with wireless access. These devices have a great capability to access data in a ubiquitous way. As a consequence of these new computing environments, the information should be distributed to fulfil the requests of different users independently of the location, platform or physical devices. The distribution of information supports ubiquitous databases, where the data are partitioned according to the autonomy degree and efficiency required. The distribution of information and new technologies makes it possible for an increase of complex attacks directed to databases. One of the weak points on new working environments is caused by the data transference through insecure communication channels, such as local networks and Internet. This weakness is exploited by a malicious user who scans the traffic for eavesdropping and can steal, change or delete sensitive information.

The SQL injection represents a potential attack for the database systems. The SQL injection attack is at the top of the list of latest threats in recent years. The damages caused by a SQL injection attack involves financial losses, reliance of the consumers, providers and trading partners. In addition, such attacks disrupt the development of outside and inside activities of the organization [13]. The architecture presented in this Chapter, named SiC (Agent based architecture for preventing and detecting attacks to ubiquitous database), is targeted for solving the problem of the SQL injection attacks on databases. This proposal is oriented for ubiquitous environments but it is not only limited for this scenario. SiC proposes a novel strategy to block SQL injection attack through a distributed approach based on the capacities of the agents and multiagent systems [49]. The philosophy of multiagent systems

allows dealing with the SQL injection attacks from a perspective of the communication elements, ubiquity and autonomous computation and from a view-point of a global coordinated system. Every component in SiC interacts to achieve a global common goal. SiC presents a hierarchical organization structured by levels or layers of agents. The agents of each level have assigned specific tasks which can be executed independently to their physical location. The complexity of the agents is incremented with the advance by the hierarchy pyramid. This hierarchical structure allows a distribution of roles and tasks for the detection and prevention of SQL injection attacks. Additionally, it has a great capacity for errors recovery.

The use of agents with advanced capabilities to reason and predict situations is the main feature of this architecture. SiC makes use of CBR-BDI agents [15], which are characterized by the integration of a CBR mechanism (Case-Based Reasoning) [1]. This mechanism provides the agents a greater level of adaptation and learning capacity. CBR systems make use of past experiences to solve new problems [22]. Thus, it is possible to generate new solutions from the results obtained in problems with similar characteristics taken place in the past. CBR systems are characterized by executing a reasoning cycle to solve each new problem. This reasoning cycle is able to make a feedback from each new experience and modifying the case memory and the reasoning capacity according to new changes. The latter is very suitable to block SQL injection attacks by anomaly detection [35], [29]. A CBR system learns and predicts behaviours or events that disclose a particular signature of a SQL injection attack. CBR-BDI agents have a predictive capacity by means of a mixture of neural network within the adaptation stage of the CBR cycle.

Agents can be characterized through their capacities in areas such as autonomy, reasoning, reactivity, social abilities, proactivity, and mobility, among others. These capacities provide great advantage for offering solutions at highly dynamic and distributed environment. Many activities in areas as networks security, e-commerce, telecommunications, among others are carried out by multiagent systems implemented successfully [19], [7], [2]. The capacity of the agents to be executed by mobile devices makes them particularly suitable to detect SQL injection attacks on ubiquitous databases. The agents integrated in SiC are based on the BDI deliberative model (Believe, Desire, Intention) [48], [21]. The internal structure of these agents and the capacities are based on mental aptitude using beliefs, desires and intentions [12].

In summary, a distributed hierarchical multiagent architecture is presented as a solution to SQL injection attacks. The main feature of SiC is the use of CBR-BDI agents with detection and prediction capabilities to classify and block this type of threats. CBR systems are especially suitable to solve classification problems, similar to the SQL injection attacks. CBR-BDI agents incorporate a mixture of neural networks in the adaptation phase of the CBR cycle to predict new attacks. Finally, the architecture handles misuse detection, which accomplishes all the strategy of detection and prevention proposed.

The preliminary results obtained after the implementation of the initial prototype show the effectiveness of the strategy to minimize attacks; the highest performance through the distribution of the workload among the available nodes into the architec-

ture; the scalability, offering an easy way to incorporate new nodes in monitored environments; a great learning and adaptation capacity, which is provided by the CBR mechanism and the mixture of neural networks; and the flexibility to be adapted to many sensitive scenario to SQL injection attacks. The ultimate goal of this work is the presentation of an effective and novel solution, designed for working in new environments, mainly in those where the mobility of the information is essential.

The remainder of this Chapter is structured as follows: Section 2 presents the problem that has prompted most of this research work; Section 3 describes the SiC architecture, different agents incorporated to the architecture and the communication among them; Section 4 explains in detail the most important agent of the SiC architecture, the CBR-BDI classifier agent. Section 5 describes a study case using a medical database and presents the results and discussion. Finally, in Section 6, conclusions are presented.

2 Database Threat and Security Revision

Data is usually stored in a ubiquitous database in order for the applications to have access to them from any location and any time. A ubiquitous system is necessarily distributed [52], claiming that data have to be present everywhere for the authorized user. This feature is achieved when the databases are partitioned, so that data are distributed in several local databases strategically located on different geographic nodes.

A ubiquitous database allows any user to access its data through custom applications. The source of these data do not need to be known by the user. The development of a ubiquitous model has two determining factors; the rising tide of Internet and the World Wide Web. These factors had been made into in a means for the global spreading and the data interchange. In this sense, databases have played a crucial role for the storage of a huge volume of data. On the other hand, the access via wireless has enabled a great interconnection among devices and unrestricted data accesses.

As a result of the decentralization of the information, new issues about the privacy and the information security have been addressed. In recent years, large companies have opted by transferring the management control through service outsourcing of a specialized supplier for specific tasks. One of the most notable outsourcing services is database outsourcing where organizations outsource the data storage and management to third-party service supplier [51]. This management model has generated discussions about the issue of sharing sensitive data that might endanger private information of clients and the organization itself. In the same vein, the knowledge extracting through rules of data mining have caused hard criticism. The tools used to discover unknown patterns can extract unauthorized information that place in risk the privacy of individuals and the confidentiality of their data [45]. Regarding the ongoing threats targeted against the information system and databases are the viruses and worms. The worms are considered a particularly dangerous threat because of

its evolution towards complex techniques to avoid the security mechanisms. They can carry an explosive charge to be executed according to fixed conditions by the hacker. New sophisticated variants of worms are expected to become more prevalent in short term such as SQL injecting attacks through the application layer [6]. Because of the increase of incidents, the information security is considered a critical issue within the strategic policies of organizations. In the commercial sector and the research centers, more resources and human capital are devoted to research new security solutions that can face new attacks and to protect corporate databases.

Security measures to protect information systems and databases of outsider attacks 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 [36]. In the particular case of the database security, it is necessary to have a closer look from a outlook of mechanisms such as access control policies, authentication and identification mechanisms. In a multilevel secure database management system (MLS/DBMS), authorized users at different security levels access share a database at different security levels without violating security. The security policy of MSL/DBMS includes a policy for mandatory access control (MAC) and discretionary access control (DAC). Mandatory security controls restrict access to data depending on the sensitivity levels of the data and the authorization level of the user. Discretionary Security measures are usually in the form of rules, which specify the type of access that users or groups of users may have to different kinds of data [10]. Additionally to these mechanisms, other approaches have arisen such as the Hippocratic Databases inspired in the Hippocratic Oath [3] and the use of cryptography techniques to protect the confidentiality of data [33].

The current databases security measures seem insufficient and less if it is examined from the perspective of the threats targeted to the new working environments. Nowadays the attacks are addressed to the application layer and the database systems causing that the protection mechanisms cannot detect them. A decade or more ago, databases were usually kept physically secure in a central data center and accessed mostly by applications into the corporate borders. However, now applications and databases may be distributed in business units to meet local needs. Even more critical is the fact that these applications and databases are increasingly available to suppliers, customers and business partners in order to carry out business over the Web [5]. Organizations are hit hard when a malicious user bypass 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) forms the backbone of many Database Management Systems, especially relational databases. It allows carry out information handling and databases management, but it also facilitates building a type of attack which results extremely lethal. The SQL injection is not a new attack, but it has not been removed of the threat list for databases.

A SQL injection attack brings harm to the organizations such as financial losses; affects customer confidence, suppliers and business partners and disrupts the outside and inside activities 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 [4], [31], [24]. The response capacity after carrying out a SQL injection attack depends on the type of technique used and the caused damage grade. This response can take hours, days and even weeks. Web applications are the main target of this type of attack. In the case of these applications, the static chain is concatenated with user inputs. If the user inputs are tainted, an injection attack is carried out when the query is executed on database. However, even though the most common attack method being through request via HTTP (HyperText Transport Protocol) protocol, other methods are vulnerable to a SQL injection attack. Applications on wireless mobile devices execute SQL queries on the database. These queries are transmitted through insecure transmission channel allowing that it can be monitored, captured and changed by a hacker. Finally, a recent vulnerability has arisen in the pervasive computing applications by the use devices or sensors vulnerable [42], [40], [41]. This new technology has presented security hole and therefore it can be exploited by a SQL injection attack causing great damage.

The cause of the SQL injection attacks is relatively simple. This attack is caused by inadequate input validation on user interface. As a result of this attack a hacker can carry out an unauthorized data handling, retrieval of confidential information, and in the worst possible case, to take over control of the application server. The main features to give a detailed description of a SQL injection attack are the attack mechanism used and the attack intention [25].

The most commons Database Management Systems such as Microsoft SQL Server, Oracle, MySQL, Informix, Sybase have been target of SQL injection attacks during recent years [32]. The problem of the SQL injection attack increases with the use of technologies designed to offer new working environments, especially in sectors such as e-commerce, healthcare system, industry, e-government, among other. The benefits offered by the new devices such as the full interconnection and corporate database access from any location, can give space to SQL injection attacks. The new working environments require information at any location and time for all the authorized users. This fact forces decentralization of data and a strategic location of databases into the working environments. In the case of the SQL injection attacks, this setting is special to exploit new vulnerabilities.

SQL injection attacks have led up to a significant number of research works, both in the sector commercial as at academic research centers. Unfortunately the advances in the detection and prevention measures have not achieved the required level to overcome this type of attack. The current security products found at the market are vital for the defence of information security; nevertheless the results against the SQL injection attacks are poor enough. The low efficacy is due to that security measures are not intended for a specific type of attack, but on the contrary, they are diversified to many threats. These security products are not intended for SQL injection attacks exclusively.

Regarding the proposed academic approaches as a solution to the SQL injection attacks, a wide revision is carried out. Some artificial intelligence techniques have

been proposed as solution to the SQL injection attack. Between the approaches revised is WAVES (Web Application Vulnerability and Error Scaner) [27]. 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 can not check all the vulnerable points like the traditional penetration testing. The strategy used by the intrusion detection systems have been implemented in the SQL injection attacks. Valeur [46] presents an IDS approach which 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 depends on the quality of the training set; a poor training set would result in a large number of false positives and negatives. Rietta [43], proposed an IDS system at the application layer using an anomaly detection model, which assumes certain behavior of the traffic generated by the SQL queries; that is, 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 the database, which requires a continue updating in order to detect new attacks. Finally, Skaruz [44] proposed the use of a recurrent neural network (RNN). The detection problem becomes a time series prediction problem. This approach leads to a large number of false alarms.

Other strategy based on techniques of string analysis and the generations of dynamic models has been proposed as solution to the SQL injection attacks. The Java String Analysis (JSA) library [16] 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 a specific string can have at a point in the application. This technique is not targeted to SQL injection attacks, but it is important because other approach use the library to generate middle forms of models. JDBC Checker [23] is a technique for statically checking the type correctness on SQL queries dynamically generated. 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 and Su [47] proposed 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 [24] propose AMNESIA (Analysis and Monitoring for Neutralizing SQL Injection Attacks). This approach uses a static analysis to build the 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 accuracy static analysis. With only slight variations

of accuracy, it generates a large number of false positive and negatives. SQLGuard [14] is an approach that checks queries at runtime to analyze if these queries conform to a model of expected queries. In this approaches, the model is expressed as a grammar that only accepts legal queries. The model is deduced at runtime by examining the structure of the query before and after the addition of user input. The approach uses a secret key to delimit user input during parsing by the runtime checker. The security of this approach depends on an attacker not being able to find the key. Additionally, it requires that the programmer rewrites the code to use a special middle library. Kosuga et al. [28] proposed SANIA (Syntactic and Semantic Analysis for Automated Testing against SQL Injection), which captures queries between Web application and database. It automatically generates crafted attacks according the syntax and semantic of vulnerable points. SANIA uses a syntactic analysis tree of the query to evaluate the security of the points. SANIA presents a drawback; it has a significant rate of false positive.

Other main query development paradigms proposed as solution to SQL injection attacks are discussed below. SQLrand [11] provides a framework that allows developers to create SQL queries 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 would not have been constructed by the randomized key-words, so the tainted SQL strings would have syntax error. SQLrand depend on secret key to modify keywords, its security relies on hackers not being able to discover this key. Additionally it requires the application developer to rewrite code. SQL DOM [34] and Safe Query Objects [17] use encapsulation of database queries to avoid SQL injection attacks. These techniques changing the process to build SQL string to one systematically way that uses a type-checked API. API is able to systematically apply coding best practices such as input filtering and close-fitting type checking of user input. Although effective, these techniques have the drawback that both require developers to learn and use a new programming paradigm or query development process.

A great interest has existed to overcome the SQL injection attacks through new solutions. However, the approach addressed for this type of attack has been limited to centralized models with little flexibility, scalability and a low efficacy. If it is considered the use of new technologies such as mobile technology, many of these solutions are not easy to implement or they require changes to adapt to this environments. In this sense a solution has been proposed to work at scenarios where the protecting of the database and the information is carried out into a ubiquitous environment. The proposal is based on a distributed hierarchical multiagent architecture, using autonomous agents organized by levels. It is an innovative solution to stop the SQL injection attacks. The special design allows incorporating two main techniques used in the IDS Systems, such as anomaly detection and misuse detection. Both techniques are integrated inside of SiC architecture.

With a well structured architecture, each component knows its roles and has the necessary resource to do its job. SiC as solution to SQL injection attack is effective, presents a great performance and provides flexibility, adaptability and scalability for

new computation environments. Detailed architecture of SiC is presented in the following Section, describing the role of each components, type of agents, interaction, communication and tasks.

3 An Architecture based on Multiagent System

The agents handle capacities such as autonomy, social abilities, reasoning, learning, mobility, among others [49]. One of the main features of agents is their ability to carry out cooperative and collaborative work, when they are grouped into multiagent systems to solve problems in form distributed [18]. These features make the agents suitable to deal with the SQL injection attacks. A distributed hierarchical multiagent architecture presents a great capacity for the distribution of task and responsibilities, such as failure recovering, adaptation to new changes and high level of learning. These factors are important to achieve a robust and efficient solution. One of the main novelties of the architecture is the use of CBR-BDI agent [30], which presents a great capacity of learning and adaptation. The agents BDI have a deliberative structure based on the BDI model [49]. Moreover, a BDI agent integrates a case-based reasoning mechanism [1] that allows to solve problems through the use of past experiences. As the core of the strategy for the classification of SQL queries is founded in detection by anomaly, it seems appropriate to use a CBR mechanism [1] that leverages past experience to detect anomaly. This CBR mechanism additionally incorporates a mixture of neural networks [38] in its reuse phase. This mixture of neural networks provides a capacity for the prediction of SQL injection attack.

The SiC presents an additional advantage through the use of wireless mobiles device, which can execute mobile agents. These devices have experimented a great growth in recent years, and it is common to find SQL queries that can be originated from different mobile devices including personal assistants (PDA), mobile phones, computer notebooks and workstations. The agents based on misuse detection and anomaly detection can be organized in a distributed way to leverage of available resources and improve the performance of the classification process, regardless of the nature of the physic device. The approach is based on an organizational design that is obtained through a multi-hierarchical architecture. The agents are distributed so that at the time of initiating a classification task, each type of agent knows its responsibilities; the data it needs to do its job and where to send the results. The interaction and communication between the agents is crucial to achieve the goal of classification and detection of SQL injection attacks.

Function of each type of agent within the architecture is described below:

- Sensor agents: They are incorporated at each device with access to the database. Their functions consist of capturing datagrams, ordering of TCP fragments for extracting the SQL query string, and syntactic analysis. The tasks of the Sensor agents end when the results (the SQL string transformed by the analysis, the result of the analysis of the SQL string and the user data) are sent to the next agent at the hierarchy of the classification process.

- FingerPrint agents: The numbers of FingerPrint agents depend on the workload at a given time. A FingerPrint agent receives the information of a Sensor agent and executes a searching process and matching with well known patterns stored in a previously built database. The FingerPrint agents works in coordination with the Pattern agents to search and save SQL string patterns in the database. The FingerPrint agent finishes its task when it sends its results together with the results of the Sensor agent to the Anomaly agent. The results of the FingerPrint agent consist of the SQL string transformed by the analysis, the result of the analysis of the SQL string, the user data and the search results.
- Pattern Agent: It is the responsible to save the new SQL string patterns in the database and search for patterns when the FingerPrint agent requests it.
- Anomaly agents: They are the core of the classification process. Their strategy is founded in a case-based reasoning mechanism that incorporates a mixture of neural networks. These agents retrieve those similar past cases with respect to the new cases, and then train the neural networks with the recovered cases and generate the final classification. The numbers of Anomaly agents depend on the workload at a given time. The result of the classification is sent to the Manager agent for the evaluation. This agent work in coordination with the LogUser agent.
- LogUser agent: This agent records the actions of the user and it searches for the user profile (the historical profile and the user statistics) when it is requested by the Classifier agent.
- Manager agent: This agent allows an expert to evaluate the classification process and situations that have not been solved in the classification process such as a suspicious classification. Moreover, it allows adjustment of the configuration of the architecture, carries a record and control of the active agents in each level and coordinates the distribution of the workload among the agents. Finally, it coordinates the alerts with the Interface agent and required actions to take over an attack when it has been detected. Avoiding the risk to compromise the architecture to a fault, an anomaly agent can be promoted to be Manager agent. This agent is selected by means of a voting method [50] between the Anomaly agents.
- Interface agent: This agent allows the interaction of the user of the security system with the architecture. The interface agent communicates the details of an attack to the security personnel when an attack is detected. It has the ability to work on mobile devices. This capacity allows ubiquitous communication to attend the alerts immediately.
- DB agent: It is in charge of executing the query in the database. When the query has been classified as legal, then it executes on the database and the results are send to the user owner of the request.
- Response agent: This agent delivers a response to the user once a classification solution is obtained. If the query has been classified as legal, the results of the query are sent to the user interface. Otherwise, if the query has been classified as illegal, a warning message is sent to the user interface.

Figure 1 presents the abstract multiagent architecture showing different types of agents in charge of the classification of SQL queries.

Fig. 1 Multiagent architecture for the classification of SQL queries

3.1 Communication Among Agents

In distributed environments, it is essential to provide necessary mechanisms for the coordination and cooperation among the agents so they can efficiently develop their tasks. The SiC architecture incorporates agents to work on mobile device such as PDAs, Smart phones, computer notebooks and also on workstations. The communication between the devices is carried out via wireless and LAN. The wireless mobile devices allow taking advantage of the portability.

The communication among the type of agents is carried out using a standard recommend by FIPA (Foundation for Intelligent Physical Agents) [8]. The standard is named CAL (Communicative Act Library), which includes a set of performative to build the message format. The platform to build SiC architecture has been JADE (Java Agent Development Framework) [9], which is an implementation extended of the FIPA standard and as such, it platform provides a set of libraries for the development of the agents. The communication of the agents by remote device is through an extension HTTP of JADE. In the case of the mobile agents, it uses Jade-Leap [9], that is another available extension of the framework. The Content specification language used is FIPA-SL [20], which allows defining the messages semantic ac-

cording to the type of contents of the message defined for SiC. Figure 2 shows an example of the messages communicated between two agents of the architecture.

```
:( Performative inform
 :sender (agent-identifier :  name SensorO1Agent
 :receiver(set (Agent-identifier : name FingerPrintO1Agent
 :Content
       String_Analizer(ParserSQL, ParserAnalysis, UserData)
 :Language FIPA-SL
```

Fig. 2 Example of a format of message communicated among the agents.

Figure 2 presents a message format transmitted by an agent to other. Once captured the SQL query by a Sensor agent, this sends a message inform type to a FingerPrint agent to carry out a detection based on pattern matching. The message includes data of the captured SQL string such as transformed SQL string, data of the SQL string analysis and user data owner of the query.

The types of messages used in the multi-agent architecture are: request, agree, cancel, inform, query-if, subscribe, propose, reject-proposal, accept-proposal, failure and not-understood. The protocols used for the communication and negotiation are defined by FIPA: FIPA-request protocol, FIPA-query protocol and FIPA-ContractNet protocol. The agents need to interact and negotiate continuously to fulfill the assigned task. Figure 3 present two examples of the communication between two agents through a protocol diagram.

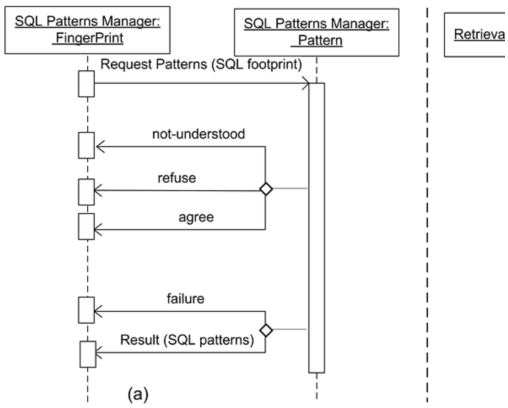

Fig. 3 Communications pattern during the interchange de message between the agents .

Figure 3(a) shows the communication between the FingerPrint agent and the Pattern agent to request stored SQL patterns when is applied for misuse detection. Figure 3(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.

The security is a primordial element in the agents communication. This resource has been provided by a secured channel through the protocol HTTPS (Hy-pertext Transfer Protocol Secure) [39]. HTTPS is an Internet Protocol that provides a SSL layer of security. This protocol uses SSL and HTTP to protect the communication channel between the client and the server on a network. When HTTP is used to access the data on the Internet, HTTPS provides strong authentication. For the internal communication between the agents, the solution applied was by means of JADE-S [9]. JADE-S is a plug-gin that supports user authentication and agents, encryption and signature of message, but JADE-S is limited for working with mobile agents.

4 Classifier Model of SQL Injection Attacks

The classifier CBR-BDI agent [37]incorporates a Case-Based Reasoning system that allows the prevention and detection of a SQL injection attack. The prevention and detection is supported by a prediction model based on neural networks, 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 Chapter presents a novel classification system that combines the advantages of the CBR systems, such as learning and adaptation, with the predictive capabilities of a mixture of neural networks. These features make the architecture appropriate for using it in dynamic environments. For working with CBR mechanism, the key concept is that of “case”. A case is defined as a previous experience and is composed of three elements: a description of the problem; a solution; and the final state. To introduce a CBR motor into a BDI agent, we represent CBR system cases using BDI and implement a CBR cycle. This CBR cycle consists of four steps: retrieve, reuse, revise and retain.

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

- Problem Description: Describes the initial information available for generating a classification. As evident in Table 1, the problem description consists of a case identification, user session and SQL query elements.
- Solution: Describes the action carried out in order to solve the problem description. As evident in Table 1, it contains the case identification and the applied solution.
- Final State: Describes the achieved state after the solution has been applied. It takes three possible values: attack, not attack and suspicious. The multi-agent architecture incorporates the Manager agent, which allows an expert to evaluate the classification.

The proposed mechanism is responsible to classify SQL database queries made by users. When a user makes a new request, it is checked by a pattern matching. These patterns are stored in a database that handles a significant number of signature that are not allowed on user level such as symbol combination, binary and hexadecimal encoding and reserved statement of language (union, execute, drop,

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

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

revoke, concat, length, asc, chr among others). If the FingerPrint agent detects some known signature, it is automatically identified as an attack. In order to identify the rest of the SQL attacks, the Anomaly agent uses a CBR mechanism, which must have a memory of cases dating back at least 4 weeks, with the structure described in Table 1. The problem description of a case is obtained by means of a string analysis technique over the SQL query. This process can be understood easily through the following example: It receives a query with the following syntax: Select field1, field2, field3 from table1 where field1 = input1 and field2=input2.

If the fields input1 and input2 are used to bypass the authentication mechanism with the following input data: Input1=‘ or 9876= 9876 – and Input2= (blank). The result of these input data would alter the SQL string as follows: Select field1, field2, field3 from Table 1 where field1 =” or 9876 = 9876 - - ‘and field2= ”

The analysis of the SQL string would generate the result presented in Table 2 with the following fields: Affected_table$^{(c1)}$, Affected_field$^{(c2)}$, Command_type$^{(c3)}$, Word_GroupBy$^{(c4)}$, Word_Having$^{(c5)}$, Word_OrderBy$^{(c6)}$, Number_And$^{(c7)}$, Number_Or$^{(c8)}$, Number_literals$^{(c9)}$, Length_SQL_String$^{(c10)}$, Number_LOL$^{(c11)}$, Cost-Time_CPU$^{(c12)}$, Query_Category$^{(13)}$. The fields Command_type and Query_Category has been encoded with the following nomenclature Command_Type: 0=select, 1=insert, 2=update, 3=delete; Query_Category: -1=suspicious, 0=illegal, 1=legal.

The first phase of the CBR cycle consists of recovering past experience from the memory of cases, specifically those with a problem description similar to the current 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 request.

Table 2 SQL String transformed through the string analysis

c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13
1	3	0	0	0	0	1	1	2	81	1	2.91	0

The cases recovered are used to train the mixture of neural networks implemented in the reuse phase. The neural network with the sigmoidal function is trained with the recovered cases that were an attack or not, whereas the neural network with hyperbolic function is trained with all the recovered cases (including the suspects). A preliminary analysis of correlations is required to determine the number of neurons of the input layer of the neuronal networks. Additionally, it is necessary to normalize the data (i.e., all data must be values in the interval [0,1]). The data used to train the mixture of networks must not be correlated. With the cases stored after deleting correlated cases, the inputs for training the mixture of networks are normalized. It is considered to be two neural networks. The results 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).

Figure 4 shows the four steps of the CBR cycle including the mixture of the neural networks through an algorithm. This strategy of classification is carried out inside an Anomaly CBR-BDI agent. This Anomaly CBR-BDI agent is located on a strategic level of the architecture.

4.1 Neural Network Learning Algorithm

As is indicated earlier, an essential element is the mixture of neural networks that is used in the reuse stage of the CBR cycle by the Anomaly CBR-BDI agent to predict attacks. This section describes in detail the operation of the mixture of neural networks. The mixture uses two neural networks, and both of them are multilayer perceptrons, but use different types of activation functions. Each of these networks obtains an individual solution for the problem. Then, the solutions provided are combined to find the optimal classification. Figure 5 illustrates a GUI of the neural networks architecture.

The new case is presented to both neural networks and then each neural network provides its independent opinions about the classification. The neural network based on a sigmoidal function gives two results (illegal or legal) and the neural network through a hyperbolic tangential function produce three results (illegal, legal or suspicious). In the following paragraphs, we describe the learning algorithm for the neural networks, explaining the differences for each type of network. The advantages of the classification method provided by each of the individual networks are discussed. Finally, the mixture is presented and formalized. The equations are presented in the order they should be executed.

```
Begin
  cases:=Retrieve(new_case)   {case retrieval function}
  assessment:=Reuse(cases[], new_case) {result of the classifi
  decision:=Revise(new_case, assessment) {revision of the clas
  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
     SQL_Query+="user=[user]and IP_Adress=[IP_adress] and Comm
     [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_typ
         Affected_table=[Content(Affected_table)]"
         Else
           If [Ip_adress] then {If only one Ip Adress is recog
               SQL_Query+="IP_adress=[IP_adress] and Command_t
               [Command_type] and Affected_table=[Content(Affe
           Else  {If user and Ip Adress are not included in th
               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 i
   cases[]:=fsimilirity_cosine(cases[]) {cosine similarity-bas
   cases[]:=fcorrelation(cases[]) {eliminating correlated case
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 p
  End If
  If blnew_case then   {If is true then training neural networ
    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 classi
    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 {Evaluatic
      decision:=true
   End If
End
Algorithm_Retain(new_case, solution) {Retain Algorithm}
Begin
   executeUpdate(new_case, solution) {Update case memory with
End
```

Fig. 4 Algorithm of the Cycle CBR for classifying SQL query

1. To present the input vector to the input layer.

$$X^p = (x_1^p, ..., x_i^p, ..., x_N^p)^T \tag{1}$$

2. To calculate the value of the levels of excitation for the neurons from the hidden layer

$$net_j^p = \sum_{i=1}^{N} W_{ji}^p(t) X_i^p(t) + \theta_j^p \tag{2}$$

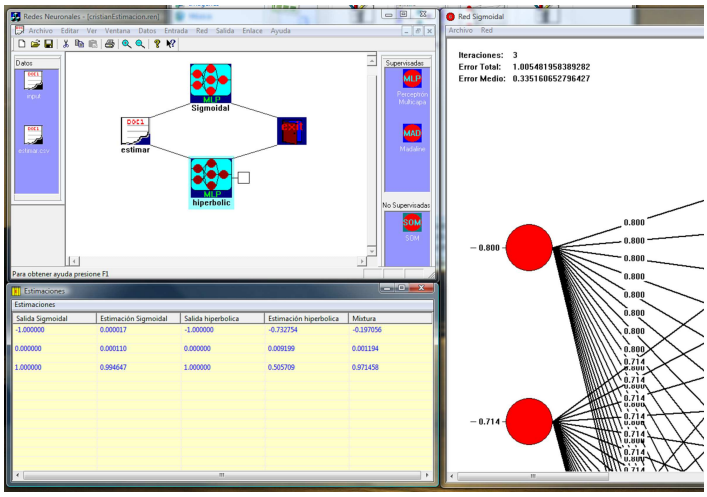

Fig. 5 Capture of the mixture of the neural networks

Where W_{ji}^{P} is the weight that connects the neuron "i" from the input layer with the neuron "j" from the hidden layer according to a "p" pattern (figure 5). θ_j^P is the threshold or bias associated to the neuron "j" from hidden layer according to a "p" pattern.

3. To calculate the outputs of the neurons from the hidden layer.

$$Y_j^p = f_j(net_j^p), \tag{3}$$

Where f_j is the deactivation function of the neurons "j" from the hidden layer.

4. To calculate the value of the levels of excitation for the neurons at the output layer.

$$net_k^p = \sum_{j=1}^{H} W_{kj}^p(t) y_j^p(t) + \theta_k^p \tag{4}$$

where W_{kj}^P is the weight that connects the neuron "k" from the output layer with the neuron "j" from the hidden layer according to a "p" pattern (figure 5). θ_k^P, it is the threshold or bias associated to the neuron "k" from output layer according to a "p" pattern.

5. To calculate the output of the neural network.

$$Y_k^p = f_k(net_k^p), \tag{5}$$

where f_k is the activation function of the neuron "k" from the output layer.

6. To calculate the sensitivity of the neurons from the output layer based on error showed at the output with the targeted output vector is defined as

$$d^p = (d_1^p, ..., d_k^p, d_M^p)^T \tag{6}$$

$$\delta_k^p = -\frac{\partial E^p}{\partial (net_k^p)} = (d_k^p - y_k^p) = \frac{\partial f_k(net_k^p)}{\partial net_k^p} \tag{7}$$

7. To calculate the sensibility of the neurons from the hidden layer is given by

$$\delta_j^p = f_j'(net_j^p) \sum_{k=1}^{M} \delta_k^p w_{kj}^p \tag{8}$$

8. To update the weights and the bias of the connections that connect the neurons from the hidden layer with the output layer

$$\Delta w_{kj}^p(t+1) = \eta \delta_k^p y_j^p + \mu \Delta w_{kj}^p(t) \tag{9}$$

$$\Delta \theta_k^p(t+1) = \eta \delta_k^p + \mu \Delta \theta_k^p(t) \tag{10}$$

η: Learning rate controls the size of the change of the weights in each iteration. μ: Momentum term allows to filter the oscillations in the surface of the error caused by the learning rate and considerably accelerates the convergence of the weights.

9. To upgrade the weights and the thresholds of the connections between the neurons of the hidden layer with the input layer

$$\Delta w_{ji}^p(t+1) = \eta \delta_j^p x_i^p + \mu \Delta w_{ji}^p(t) \tag{11}$$

$$\Delta \theta_j^p(t+1) = \eta \delta_j^p + \mu \Delta \theta_j^p(t) \tag{12}$$

10. To calculate the error

$$E^p = \frac{1}{2} \sum_{k=1}^{M} (d_k^p - y_k^p)^2, \tag{13}$$

where d_k^P is the desired output of the neuron "k" from the output layer according to a "p" pattern. Since this term reflects the adaptation capacity of the neural network, it is necessary to keep it in mind to determine if the neural network learns in a satisfactory way or not. As previously explained the mixture is composed of two multilayer perceptrons, one of then uses sigmoidal function and the other tangential function. In this sense, the algorithm has to been adapted by considering the activation functions, the sigmoidal or the tangential function.

- The Sigmoidal activation function has its range within the interval [0,1]; It is used to detect if the request is an attack or not. The value 0 represents an illegal re-quest and 1 a legal request. The Sigmoidal activation function is the activation function most used for classifications between two groups. This function has drawbacks that it works well for binary classifications.

That is:

$$f(x) = \frac{1}{1+e^{-ax}} \tag{14}$$

$a = 1$. For the weights used to connect a hidden layer with an output layer; the updating formula of the weights in series is given by "p" pattern as follows.

$$\Delta w_{kj}^{p}(t+1) = \eta \delta_k^P y_j^p + \mu \Delta w_{kj}^P(t) = \eta (d_k^p - y_k^p)(1-y_k^p) y_k^p y_j^p + \mu \Delta w_{kj}^p(t) \tag{15}$$

For the bias associated to the neurons from the output layer, given a "p" pattern, the updating formula of the weights in series is given by

$$\Delta \theta_k^p(t+1) = \eta \delta_k^p + \mu \Delta \theta_k^p(t) = \eta (d_k^p - y_k^p)(1-y_k^p) y_k^p + \mu \Delta \theta_k^p(t) \tag{16}$$

For the weights used to connect the input layer with the hidden layer; the updating of the weights in series is give a "p" pattern is given by

$$\Delta w_{ji}^p(t+1) = \eta (1-y_j^p) y_j^p (\sum_{k=1}^{M} \delta_k^p w_{kj}) x_i^p + \mu \Delta w_{ji}^p(t) = \tag{17}$$

$$= \eta (1-y_j^p) y_j^p (\sum_{k=1}^{M} (d_k^p - y_k^P)(1-y_k^p) y_k^p w_{kj}) x_i^p + \mu \Delta w_{ji}^p(t)$$

For the bias associated to the neurons from the hidden layer, given "p pattern; the update is given by

$$\theta_j^p(t+1) = \theta_j^p(t) + \eta (1-y_j^p) y_j^p (\sum_{k=1}^{M} \delta_k^p w_{kj}) + \mu (\theta_j^p(t) - \theta_j^p(t-1)) = \tag{18}$$

$$\theta_j^p(t) + \eta (1-y_j^p) y_j^p (\sum_{k=1}^{M} (d_k^p - y_k^p)(1-y_k^p) y_k^p w_{kj}) + \mu (\theta_j^P(t) - \theta_j^p(t-1))$$

- The hyperbolic tangential function has its range 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 illegal request, value 1 represent legal request and value -1 of those suspicious requests. Hyperbolic tangential function is suitable for classifying in three groups. Hyperbolic tangential activation function is given by

$$f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{19}$$

For the weights used to connect a hidden layer with an output layer, the updating formula of the weights in series is given by a "p" pattern is

$$\Delta w_{kj}^p(t+1) = \eta \delta_k^p y_j^p + \mu \Delta w_{kj}^p(t) = \eta (d_k^p - y_k^p)(1-y_k^p) y_k^p y_j^p + \mu \Delta w_{kj}^p(t) \tag{20}$$

For the bias associated to the neurons from the output layer, given a "p" pattern; the updating formula of the weights in series is defined as

$$\Delta \theta_k^p(t+1) = \eta \delta_k^p + \mu \Delta \theta_k^p(t) = \eta (d_k^p - y_k^p)(1-(y_k^p)^2) + \mu \Delta \theta_k^p(t) \tag{21}$$

For the weights used to connect the input layer with the hidden layer, the updating of the weights in series is given by a "p" pattern is defined as

$$\Delta w_{ji}^p(t+1) = \eta (1-(y_j^p)^2)(\sum_{k=1}^{M} \delta_k^p w_{kj}) x_i^p + \mu \Delta w_{ji}^p(t) \tag{22}$$

$$= \eta (1-(y_j^p)^2)(\sum_{k=1}^{M} (d_k^p - y_k^p)(1-y_k^p) y_k^p w_{kj}) x_i^p + \mu \Delta w_{ji}^p(t)$$

For the bias associated to the neurons from the hidden layers, given a "p" pattern, the updating in series is defined as

$$\theta_j^p(t+1) = \theta_j^p(t) + \eta (1-(y_j^p)^2)(\sum_{k=1}^{M} \delta_k^p w_{kj}) + \mu (\theta_j^p(t) - \theta_j^p(t-1)) = \tag{23}$$

$$\theta_j^p(t) + \eta (1-(y_j^p)^2)(\sum_{k=1}^{M} (d_k^p - y_k^p)(1-(y_k^p)^2) w_{kj}) + \mu (\theta_j^p(t) - \theta_j^p(t-1))$$

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

tivation function for the other Multilayer Perceptron (which take values within [-1,1], where -1 = Suspect, 0 = illegal and 1 = legal). The response of both networks is combined, to obtain the mixture of networks denoted by y^2; where the superscript indicates the number of mixtured networks.

$$y^2 = \frac{1}{\sum_{r=1}^{2} e^{-|1-r|}} \sum_{r=1}^{2} y^r e^{-|1-r|} \tag{24}$$

The number of neurons in the output layer for both networks is 1, and is responsible for deciding whether or not there is an attack. The error of the training phase for each of the neural networks can be quantified using (24), where P is the total number of training patterns.

$$Error = \frac{1}{P} \sum_{i=1}^{P} \left| \frac{Forecast_P - Target_P}{Target_P} \right| \tag{25}$$

5 Experimental Results and discussion

A case study has been proposed to test the effectiveness of a SiC prototype. The prototype has been evaluated by means of a previously developed multiagent system, installed in a geriatric residence [18]. The implemented multiagent system improves the security of the patients, facilitates the carers' activity and guarantees an adequate level of efficiency. The system has been developed in a distributed environment containing devices such as PDA, notebook computers and accessed via wireless. A back-end database stores and supplies information. The database manager is Oracle. The actors in the scenario such as nurses, doctors, patients, worker social and other employees can be seen in Figure 6. The medical staff in charge of patients's care was integrated by 2 doctors, 10 nurses and 1 social worker. The number of patients under the monitoring and attention of the multi-agent system was 30. In the case of the nurses, each nurse was equipped with a PDA, thus a total of 10 PDAs execute queries on the database during the working day. With these data, we prepared the attack scenario. The performance of the test required to incorporate equipments and mobile devices with connection via wireless and LAN. Equipments include 2 workstations and 3 PDAs. The test has been carried out during 30 working days without interruption.

During the execution of the multiagent system, several types of SQL queries were carried on the database. The queries were related to the patients' treatments, the scheduling of the working day of 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 role of the nurse. The main volume of queries were generated each time that a plan was assigned to a nurse. The plans changed due to different reasons during their execution and these changes increased the number of queries on the database. When a nurse starts and

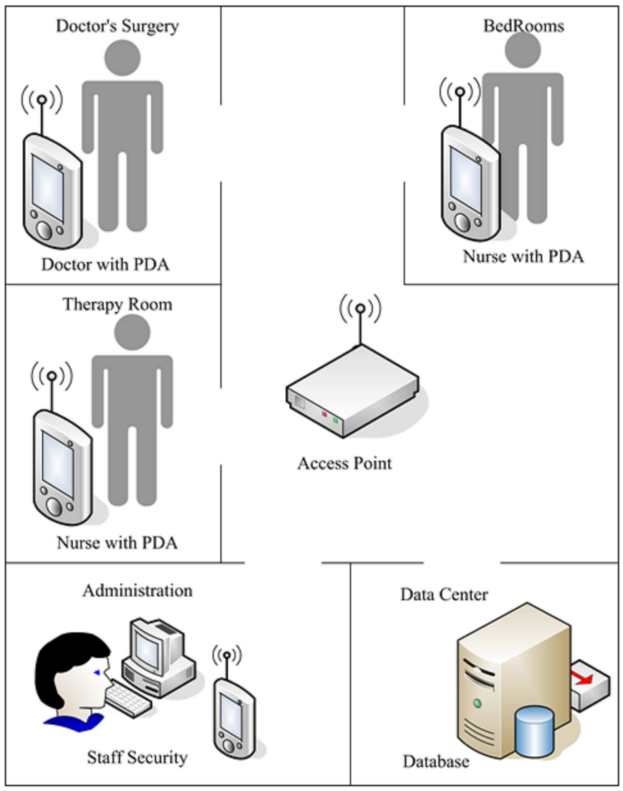

Fig. 6 Abstract scenario of the real environment (Geriatric Residence)

finalizes a task, a response is send through a SQL query. The nurses have direct access to the database system by means of the application interface on the PDAs. The strategy was based on the execution of crafted queries from 2 PDAs. These PDAs were fixed with a similar user interface with the nurses' PDAs, but these have capacity to execute tainted queries. When a query is executed from the PDA of attack, this query carries out a type of SQL injection that has to be captured, analyzed and classified as legal, illegal or suspicious. The FingerPrint agents and Anomaly agents were distributed in the 2 workstations. As the test was carried out on the real medical database, a special mechanism has been built to guarantee the integrity of the database. All the queries executed both by the nurses' PDA and the attack PDAs are examined and classified. The test was conducted with a total of 12 PDAs available, 10 PDAs assigned to the active nurses and 2 PDAs to execute attacks, a total of 10,200 queries were sent to the medical database. Each nurse's PDA executed around 30 daily queries and, during the 30 days of the test, 9,000 legal queries were carried out. In the case of the two attack PDAs, each PDA executed 20 illegal daily queries. These PDAs sent 40 events of attack during a working day. Throughout the 30 days of the test, a total of 1,200 events of attack were targeted to

the medical database. The volume of queries during the test period allows building a case memory to validate the strategy proposed.

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, and which generated attack consults. The results obtained are promising, improving in many cases those obtained with other existing techniques, which let us conclude that SiC can be considered as a serious alternative to detect and predict SQL injection attacks. The classification system integrated within the Anomaly agent provides the results illustrated in Table 3, 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 3 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 of only 0.5% of the cases.

Table 3 Results obtained after testing different classification techniques

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

As shown in Table 3, the Bayesian method is the most accurate statistical method, since it is based on the likelihood of the events observed. But it has the disadvantage of determining the initial parameters of the algorithm, although it is the fastest of the considered statistical methods. Taking into account, the errors obtained using different methods, after the neural networks and Bayesian methods we found that the regression models could also be used. Due to the non linear behaviour of the hackers, linear regression offers the worst results, followed by the polynomial and exponential regression. This can be explained by looking at hacker behavior: as the hackers break security measures, the time for their attacks to obtain information decreases exponentially.

The empirical results show that the best methods are those that involve the use of neural networks 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 behavior of the hacker is not linear, dynamic and chaotic.

The advantage of using a mixture of neural networks improves performance that provides other classification techniques, but also improves performance that can provide the neural networks on an individual basis. The mixture has the advantage

that the number of cases where the classifier agent could not decide is smaller and in few cases the mediation by an human expert human was required. We could check the decision of the mixture of networks with the verdict of an human expert for those cases which a single network did not decide and both the mixture of networks and the human expert were in agreement on 99% of cases. Figure 7 depicts the effectiveness of the classification schemes both for the networks with distinct activation function work on an individual basis and the effectiveness of the mixture of networks.

Figure 8 shows the success of the predictions for the number of training patterns presented in Table 4. As observed, with a large number the training patterns, percentage of successful prediction could be also improved.

Table 4 Successful (%) depending on the 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

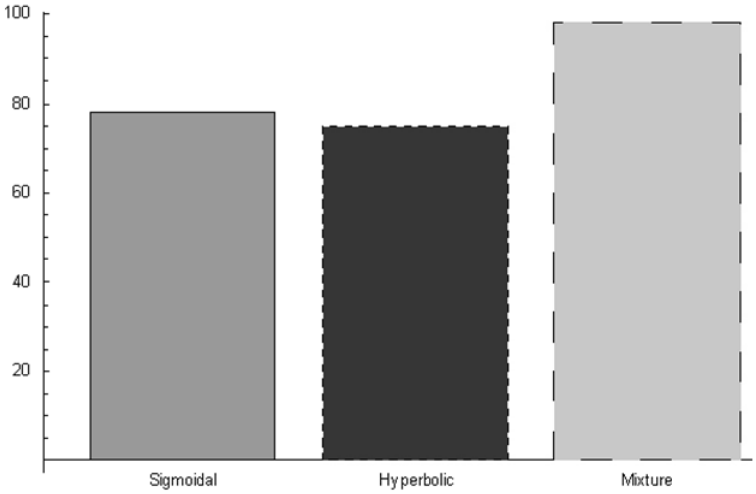

Fig. 7 Effectiveness in the classification of the networks on an individual basis and the mixture of networks

When the number of patterns for training the neural network increases, the prediction error decreases. It is to be noted that the number of training patterns are the

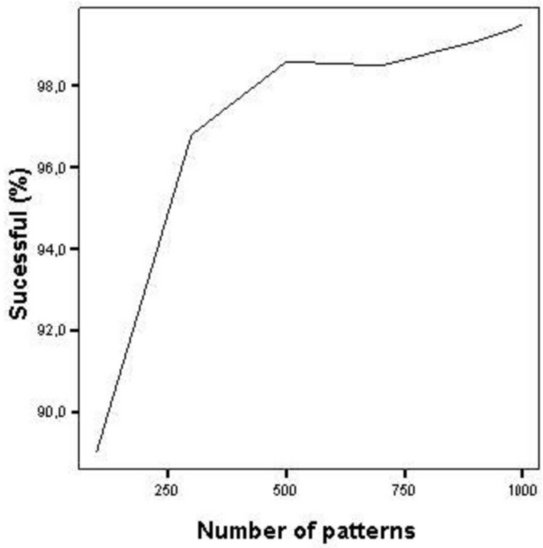

Fig. 8 Successful percentage vs. number of patterns

result after applying filters such as the similarity based algorithm and the correlation function. These filters reduce the quantity of cases meaningfully in order to improve the performance during the training stage.

6 Conclusions

The problem of SQL injection attacks on databases presents a serious threat against information systems. This Chapter has presented a novel solution, consisting of a new hierarchical multiagent architecture for detecting SQL injection attacks, which combines the advantages of multiagent systems, such as autonomy and distributed problem solving, with the adaptation and learning capabilities of the CBR systems. The SiC architecture proposed a new perspective in the strategies for detecting and predicting SQL injection attacks, since the existing approaches are based on centralized strategies. The SiC architecture provides a distributed hierarchical structure, which allows a more efficient balance and distribution of the tasks involved in the problem of detecting and classifying attacks to databases by means of SQL injection. The core of the architecture is a special type of CBR-BDI agent, which assures great capacities for learning and adaptation. This agent is a classifier agent that, supported by the philosophy of the case-based reasoning mechanisms, proposes a new strategy, based on the use of past experiences, to classify SQL injection attacks. This strategy differs in its conception from the existing ones. Moreover, it incorporates the prediction capabilities that characterize neural networks.

The results obtained illustrated a high prediction capacity for the CBR-BDI classifier agents, about a 99.5%, which notably improves the efficiency of the existing solutions. A key factor for the success of the SiC architecture relies on the quality of the cases stored in the memory of cases, requiring a dating back of at least 4 weeks for a correct prediction. The prediction strategy based on a mixture of neural networks presented a clear advantage: the number of problems analyzed where the classifier agent can not provide an automatic decision is notably reduced and, consequently, only in few cases the mediation of an expert human was required. The experimental results indicate that around 99% of cases, both the mixture of networks and the human expert were in agreement.

Finally, the SiC architecture combines techniques based on anomaly detection and misuse detection. This combination achieves a robust solution to block any type of SQL injection attack. The SiC multiagent architecture provides flexibility and scalability to protect ubiquitous databases in new computational environments. The results are promising and it is worthy to conclude that the SiC architecture could improve the results provided by current technologies. The next step is to have a full solution where all the type of agents in the architecture can carry out their tasks in order to improve the effectiveness and the performance in a global way.

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

References

1. Aamodt A. Plaza E (1994) Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Communications. Vol. 7. pp. 39-59.
2. Abraham A, Jain R, Thomas J, Han S Y (2007) D-SCIDS: Distributed soft computing intrusion detection system. Journal of Network and Computer Applications, Elsevier. Vol. 30(1) pp. 81-98
3. Agrawal R, Kiernan J, Srikant R, Xu Y (2002) Hippocratic databases. In: 28th inter-national conference on Very Large Data Bases. Hong Kong. pp.143-154.
4. Anley C (2002) Advanced SQL Injection in SQL Server Applications. NGS Software http://www.nextgenss.com/papers/advanced sql injection.pdf. Accessed 10 April 2007
5. Application Security Inc (2005) Protecting the Crown Jewels. http://www.appsecinc.com/cgi-bin/search.pl?Terms=crown. Accessed 12 April 2008
6. Application Security Inc (2007) Introduction to Database and Application Worms. http://www.appsecinc.com/presentations/Database_Application_Worms.pdf. Accessed 10 April 2008
7. Bajo J, De Luis A, González A, Saavedra A, Corchado J M (2006) A Shopping Mall Multiagent System: Ambient Intelligence in Practice. In: 2nd International Workshop on Ubiquitous Computing & Ambient Intelligence. pp. 115-125.
8. Bellifemine F, Poggi A, Rimassa G (1999) Jade: A FIPA-compliant agent framework. In: Proceedings of PAAM-1999, pp. 97-108.
9. Bergenti F, Poggi A (2001) LEAP: a FIPA Platform for Handheld and Mobile Devices. In: Proceedings of the ATAL 2001 Conference. Seattle, USA.
10. Bertino E, Sandhu R (2005) Database Security-Concepts, Approaches, and Challenges. In: IEEE Computer Society, Los Alamitos, USA. Vol. 2. pp. 2-9.

11. Boyd S W, Keromytis A D (2004) SQLrand: Preventing SQL Injection Attacks. In: Applied Cryptography and Network Security. Vol. 3089 pp. 292-302.
12. Bratman M E (1987) Intention, Plans, and Practical Reason, Harvard University Press, Cambridge, MA.
13. Breach Security, Inc (2007) The Web Hacking Incidents Database. http://www.breach.com/. Accessed 02 April 2008
14. Buehrer G, Weide B W, Sivilotti P A G (2005) Using parse tree validation to prevent SQL injection attacks. In: 5th international workshop on Software engineering and middleware. ACM, New York. pp. 106-113.
15. Carrascosa C, Bajo J, Julian V, Corchado J M, Botti V (2008) Hybrid multi-agent architecture as a real-time problem-solving model. Expert Systems with Applications. Vol. 34(1). pp. 2-17.
16. Christensen A S, Moller A, Schwartzbach M I (2003) Precise Analysis of String Expressions. In: 10th International Static Analysis Symposium. Springer-Verlag. pp. 1-18.
17. Cook R, Rai S (2005) Safe query objects: statically typed objects as remotely executable queries. In: 27th international conference on Software engineering. ACM. St. Louis, USA. pp. 97-106.
18. Corchado J M, Bajo J, Abraham A (2008) GerAmi: Improving Healthcare Delivery in Geriatric Residences. Intelligent Systems, IEEE. vol. 23 pp.19-25
19. Corchado J M, Bajo J, De Paz Y, Tapia D (2008) Intelligent Environment for Monitoring Alzheimer Patients, Agent Technology for Health Care. In: Decision Support Systems. Vol. 34(2) pp.382-396.
20. Foundation for Intelligent Physical Agents. http://www.fipa.org. Accessed 15 August 2007
21. Georgeff M P, Lansky A L (1987) Reactive Reasoning and Planning. In: American Association of Artificial Intelligence. Seattle, USA. pp. 677-682.
22. Glez-Bedia M, Corchado J M (2002) A Planning Strategy Based on Variational Calculus for Deliberative Agents. In: Computing and Information Systems Journal. Vol. 10(1) pp. 2-14.
23. Gould C, Su Z, Devanbu P (2004) 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.
24. Halfond W, Orso A (2005) AMNESIA: Analysis and Monitoring for Neutralizing SQL-injection Attacks. In: 20th IEEE/ACM international Conference on Automated software engineering. ACM, New York. pp. 174-183.
25. Halfond W G, Viegas J, Orso, A. (2006) A Classification of SQL-Injection Attacks and Countermeasures. In: IEEE International Symposium on Secure Software Engineering. Arlington, USA.
26. Hayat Z, Reeve J, Boutle C (2007) Ubiquitous security for ubiquitous computing. In: Elsevier Advanced Technology Publications, Oxford, United Kingdom, 172-178
27. Huang Y, Huang S, Lin T, Tsai C (2003) Web application security assessment by fault injection and behavior monitoring. In: 12th international conference on World Wide Web. ACM, New York, USA. pp. 148-159
28. Kosuga Y, Kono K, Hanaoka M, Hishiyama M, Takahama Y (2007) Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection. In: 23rd Annual Computer Security Applications Conference. IEEE Computer Society. pp. 107-117
29. Kruegel C, Vigna G (2003) Anomaly detection of web-based attacks. In: 10th ACM conference on Computer and communications security, ACM, New York. pp. 251-261.
30. Laza R, Pavón R, Corchado J M (2003) A Reasoning Model for CBR_BDI Agents Using an Adaptable Fuzzy Inference System. In: 10th Conference of the Spanish Association for Artificial Intelligence. Springer. Vol. 3040 pp. 96-106.
31. Litchfield D (2005) Data Mining with SQL Injection and Inference, NGS Software. http://www.ngssoftware.com/research/papers/sqlinference.pdf. Accessed 10 April 2007.
32. Litchfield D, Anley C, Heasman J, Grindlay, B (2005) The Database Hacker's Handbook: Defending Database Servers. John Wiley, New York.
33. Maurer U (2004) The role of cryptography in database security, In: ACM SIGMOD international conference on Management of data. ACM, New York. pp. 5-10.

34. McClure R A, Krger I H (2005) SQL DOM: compile time checking of dynamic SQL statements. In: 27th international conference on Software engineering. ACM, New York. pp. 88-96.
35. Mukkamala S, Sung A H, Abraham A (2005) Intrusion detection using an ensemble of intelligent paradigms. Journal of Network and Computer Applications, Elsevier. Vol. 28(2) pp. 167-182.
36. Pervasive Software Inc (2003) Implementing Security Best Practices for HIPAA with Pervasive.SQL. http://www.msmiami.com/custom/downloads/Pervasive_HIPAASecurity_Paper.pdf. Accessed 10 April 2008
37. Pinzón C, De Paz Y, Cano R (2008) Classification Agent-Based Techniques for Detecting Intrusions in Databases. In: 3rd International Workshop on Hybrid Artificial Intelligence Systems.
38. Ramasubramanian P, Kannan A (2004) Quickprop Neural Network Ensemble Forecasting a Database Intrusion Prediction System. Neural Information Processing. Vol. 5 pp. 847-852
39. Rescorla E, Schiffman A (1999) The Secure HyperText Transfer Protocol RFC Editor, United States. http://www.rfc-editor.org/rfc/rfc2660.txt. Accessed 10 January 2008
40. Rieback M R, Crispo B, Tanenbaum A S (2006) Is Your Cat Infected with a Computer Virus?. In: Fourth Annual IEEE International Conference on Pervasive Computing and Communications. IEEE Computer Society. Washington, DC, USA. pp. 169-179.
41. Rieback M R, Crispo B, Tanenbaum A S (2006) RFID Malware: Truth vs. Myth. IEEE Security and Privacy. Vol. 4(4) pp. 70-72.
42. Rieback M R, Simpson P N, Crispo B, Tanenbaum A S (2006) RFID malware: Design principles and examples. In: Pervasive and Mobile Computing. Vol. 2(4) pp. 405-426.
43. Rietta F (2006) Application layer intrusion detection for SQL injection. In: 44th annual Southeast regional conference. ACM, New York. pp. 531-536.
44. Skaruz J, Seredynski F (2007) Recurrent neural networks towards detection of SQL attacks. In: 21th International Parallel and Distributed Processing Symposium. IEEE International. pp. 1-8.
45. Thuraisingham B (2002) Data mining, national security, privacy and civil liberties. ACM, New York, Vol. 4(2) pp. 1-5.
46. Valeur F, Mutz D, Vigna G (2005) A Learning-Based Approach to the Detection of SQL Attacks. In: Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment. Vienna, Austria. pp. 123-140
47. Wassermann G, Su Z (2004) An Analysis Framework for Security in Web Applications. In: FSE Workshop on Specification and Verification of Component-Based Systemspp. pp.70-78.
48. Wooldridge M, Jennings N R (1995) Intelligent Agents: Theory and Practice. Knowledge Engineering Review. Vol. 10(2) 115-152.
49. Woolridge M, Wooldridge M J (2002) Introduction to Multiagent Systems. John Wiley, New York.
50. Wu J, Wang C, Wang J, Chen S (2006) Dynamic Hierarchical Distributed Intrusion Detection System Based on Multi-Agent System. In: EEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology. IEEE International. pp. 89-93.
51. Xiong L, Chitti S, Liu L (2007) Preserving data privacy in outsourcing data aggregation services. In: ACM Transactions on Internet Technology. New York. Vol. 7(3), pp. 17.
52. Zaidenberg S, Reignier P, Crowley, J L (2007) An Architecture for Ubiquitous Applications. In: 1st International Joint Workshop on Wireless Ubiquitous Computing. Vol. 1 pp. 86-95.