An Attack Detection Mechanism Based on a Distributed Hierarchical Multi-agent Architecture for Protecting Databases

Cristian Pinzón[1], Yanira de Paz[2], Rosa Cano[3], and Manuel P. Rubio[4]

[1] Universidad Tecnológica de Panamá, Av. Manuel Espinosa Batista, Panama
cristian.pinzon@utp.ac.pa
[2] Universidad Europea de Madrid, Tajo s/n 28670, Villaviciosa de Odón, Spain
yanirarosario.depaz@uem.es
[3] Instituto Tecnológico de Colima, Av. Tecnológico s/n, 28976, Mexico
rdegca@gmail.com
[4] Escuela Politécnica Superior de Zamora, Av. Cardenal Cisneros 34, 49022, Zamora, Spain
mprc@usal.es

Abstract. This paper presents an innovative approach to detect and classify SQL injection attacks. The existing approaches are centralized while this proposal is based on a distributed hierarchical architecture to provide a robust and dynamic strategy. The strategy for the classification and detection of SQL injection attacks uses a combination based on detection by anomalies and misuses. The detection by anomaly uses a case-based reasoning mechanism incorporating a mixture of neural networks. The approach has been tested and the results are presented in this paper.

Keywords: SQL injection, Security database, IDS, Multi-agent, case-based reasoning.

1 Introduction

A potential security problem on the database is a SQL injection attack. This attack seriously affects the database and it takes place when an original query is modified and is executed on the database by a hacker. The SQL injection attack has been addressed by the majority of the proposal from a centralized perspective [1] [2]. The main drawback of these approaches is that they solve the SQL injection attacks partially. Other solutions more sophisticated apply intrusion detection techniques [3] [4], but they have as drawback their large rate of cases poorly classified.

The proposal presented in this work tackles the SQL injection attack problem through a distributed hierarchical multi-agent architecture. Within the architecture are implemented strategies based on misuse and anomaly detection [5]. The key component of the architecture is a type of BDI (Belief, Desire and Intention) deliberative agent [6] which incorporates a based-case reasoning (CBR) mechanism [7]. The idea of a CBR mechanism is to exploit the experience gained from similar problems in the past and to adapt then successful solution to the current problem. This CBR-BDI type of agent [8] has been specially adapted to resolve the SQL injection attack problem. This agents use the CBR concept to gain autonomy and improve their problem-solving capabilities. In addition, it integrates a novel strategy of classification that lies in a mixture of neural networks which allows carrying out short term attack predictions. This work presents an entirely new approach in order to face the problem of SQL injection attack and describing an architecture that is unique in its conception.

The rest of the paper is structured as follows: section 2 presents the problem that has prompted most of this research work. Section 3 focuses on the details of the multiagent architecture, section 4 explains in detail the classification model integrated within the classifier agent. Finally, section 5 describes how the classifier agent has been tested inside a multi-agent system and presents the results obtained.

2 SQL Injection Attacks

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 [1] [9]. The results of this attack can produce unauthorized handling of data, retrieval of confidential information, and in the worst possible case, taking over control of application server. The main problem for the detecting of SQL injection attack is the large number of variants. The detection of some SQL injection results trivial whereas that the detection of other result extremely complex due to large number of possible strategies.

Nowadays, this type of attack has been handled from distinct perspectives. The string analysis [10] has been the support of many others approaches such as [1] and [11], which carried out an analysis more complete applying a treatment dynamic and hybrid over the SQL string. In other cases, artificial intelligence techniques have been applied to face the SQL injection attack, such as [12] with WAVES (Web Application Vulnerability and Error Scanner). This proposal uses a black-box technique which includes a machine learning approach. Valeur [3] presented 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. Rietta [4] proposed an IDS at the application layer using an anomaly detection model. Finally, Skaruz [13] proposed the use of a recurrent neural network (RNN). The detection problem becomes a time serial prediction problem. Usually, many approaches present a large number of false positive and false negative. The proposals based on intrusion detection depend on database, which requires a continue updating in order to detect new attacks.

Our approach takes advantage of the multi-agent system to reanalyze the problem in a distributed mode. Moreover, intrusion detection technique based on misuse and anomaly has been incorporated at strategic level into the architecture. The detection by anomaly is built by means of a case-based reasoning (CBR) mechanism [7], whose characteristics do it especially suitable to tackle classification problems and this is reinforced with the predictive capacity of a mixture of neural network [14]. The capture of SQL queries is carried out through of distributed agents and the detection can be executed in a distributed mode. Moreover, the architecture presents a high scalability, flexibility and learning capacity that allows it a greater adaptation for distributed environments and new strategic of attacks.

3 Detection SQL Injection Based on Multi-agent Architecture

The agents are characterized through their capacities such as autonomy, reactivity, pro-activity, social abilities, reasoning, learning and mobility [6]. One of the main features of agents is their ability to carry out cooperative and collaborative work, when they are grouped into multi-agent systems to solve problems in a distributed way [15], [16] [17] These features make to the agents suitable to face the SQL injection attack problem. A distributed hierarchical multi-agent presents a great capacity for the distribution of task and responsibilities, error recovering, adaptation to new changes and high level of learning. These factors are keys to achieve a robust and efficient solution. One main innovation of the architecture is the use of a CBR-BDI agent [8], which presents a great capacity of learning and adaptation. The agents BDI have a deliberative structure based on the BDI model [6]. Moreover, a BDI agent integrates a case-based reasoning mechanism [7] that allow it solve problems through the use de past experiences. As the core of the strategy for the classification of SQL queries is based on an anomaly detection technique, it seems appropriate to use a CBR mechanism [7] that leverage past experience to detect anomaly. This CBR mechanism additionally incorporates a mixture of neural networks [14] in its reuse phase. Using a mixture of neural networks improves the performance provided by other classification techniques such as Back-Propagation Neural Networks, Bayesian Forecasting Method, Exponential Regression, Polynomial Regression, Linear Regression, but also improves performance provided by the neural networks working individually. The number of cases where the classifier mechanism can not provide a decision is small and in few cases would be needed the intervention of a human expert.

An additional advantage provided by the architecture is the ability for executing agents on mobile devices. It is common to find SQL queries that can be originated from different mobile devices including assistant personals (PDA), mobile phones, notebook computers and workstations. Specialized agents (misuse and anomaly) can be organized in a distributed way to take advantage of available resources and improve the performance of the classification process, regardless of the nature of the physical devices. Finally, another advantage to use this type of agents is the ability to inform to security staff about events that are happening regardless of their physical location, sending alerts on mobile devices. All these advantages are achieved through an organizational design based on a hierarchical multi-agent architecture. These agents are distributed so when a classification starts, each type of agent knows its concrete tasks; the data required to carry out its job and where to send its results. The interaction and communication between the agents is crucial to achieve the goal of classification of the new SQL query:

Next, is described each type of agent within of the architecture:

- Sensor agents: Located in each of the devices accessing the database. They have 3 specific functions: a) The capture of datagrams launched by the devices. b) Order TCP fragments to extract the request's SQL string. c) Syntactic analysis of the request's SQL string. The duties of the agent Sensor end when the results (the SQL string transformed by the analysis, the result of the analysis of the SQL string and the user data) are sent to the next agent at the hierarchy of the classification process.

- FingerPrint agents: The numbers of agents FingerPrint depend on the workload at a given time. An agent FingerPrint receives the information of a Sensor agent and executes a pattern matching known attacks stored at a previously built database. The FingerPrint agent finishes its task when it sends its results to the Anomaly agent. The results of the FingerPrint agent consist of the SQL string transformed by the analysis, the result of the analysis of the SQL string, the user data and the results achieved by pattern matching.
- 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: These agents are based on the CBR-BDI model. They are the key component within the classification process. Their strategy is based on a case-based reasoning mechanism that incorporates a mixture of neural networks. These agents retrieval those similar past cases to the new case of classification, training the neural networks with the recovered cases and generating 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.
- Loguser agent: This agent records the actions of the user and searching for the user profile (the historical profile and the user statistics) when it is requested by the Anomaly agent.
- Manager agent: It is the agent responsible for decision-making, evaluation and coordination of the overall operation of architecture. It evaluates the final decisions for classifications, manages alerts of attacks and coordinates the actions necessary when an attack is detected. It selects an Anomaly agent by means of a voting method.

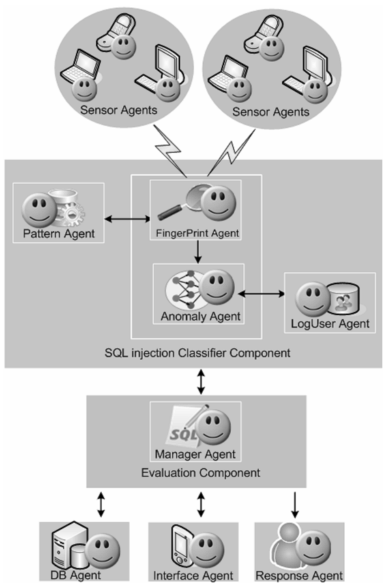

Fig. 1. Description of the hierarchical multi-agent architecture

- 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 a ubiquitous communication to attend the alerts immediately.
- DB agent: It is in charge of executing the query in the database when the classification of the SQL query is legal, that is, the SQL query is not malicious.
- Response agent: This agent provides an answer to the user once obtained a solution of the classification. If the query has been classified as legal, the result of the query is sent to the user interface. Otherwise, if the query has been classified as illegal, it is sent to the user interface a warning message.

In figure 1 is presented the hierarchical multi-agent architecture showing different types of agents in charge of the classification of SQL queries.

4 Classifier Model of SQL Injection Attacks

The Anomaly agent type has been specially adapted to resolve the SQL injection attack problem. This agent is based on CBR-BDI model [8], which incorporates a case-based reasoning system that allows the detection and blocking of SQL injection attacks reusing previous experiences. This mechanism uses a prediction model based on neural networks, configured for short-term predictions of intrusions. To carry out this short-term prediction, the CBR mechanism uses a memory of cases which identifies past experiences with the corresponding indicators that characterize each of the attacks. 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: It describes the initial information available for generating a classification. As can see in Table 1, the problem description consists of a case identification, user session and SQL query elements.
- Solution: Describes the action carried out to solve the problem description. As shown in Table 1, contains the case identification and the applied solution.
- Final State: Describes the achieved state after that the solution has been applied. It takes three possible values: attack, not attack o suspect. The Manager agent allows an expert to evaluate the classification.

The integration CBR-BDI [8] allows a BDI agent to use case-based reasoning to resolve the problem of classifying of a SQL query and blocking SQL injection attack. Regarding each state of a CBR system equivalent to a belief, the intention will be the plan that contains an ordered set of actions that the CBR-BDI agent should make to achieve the goals and each desire corresponds to one or more of the achieved final states in the past. The result of classifying a SQL query as attack, not attack or suspect is the desire that the agent seeks to achieve.

Table 1. Problem definition and solution for a case of SQL query classification

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

The proposed mechanism is responsible for classifying SQL database requests made by users. When a user makes a new request, it is checked for matching well-known patterns of attack by a FingerPrint agent (misuse detection). These patterns are stored at a database that handles a significant number of signature not allowed on user level such as symbol combination, binary and hexadecimal encoding and reserved statement of language (`union, execute, drop, 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 on the SQL query. This process can be understood easily through the following example: It is captured a SQL query with the following syntax: `Select field1, field2, field3 from table1 where field1 = input1 and field2=input2.` If we assume that 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 table1 where field1 ='' or 9876 = 9876 -- 'and field2=''`

A syntactical analysis of the SQL string would generate the result presented in the Table 2 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$^{(c9)}$, Length_SQL_String$^{(c10)}$, Cost_Time_CPU$^{(c11)}$. The fields Command_type and Query_Category have been encoding with the following nomenclature Command_Type: 0=select, 1=insert, 2=update, 3=delete; Query_Category: -1= suspect, 0=illegal, 1=legal.

Table 2. SQL String transformed through a syntactical analysis technique

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

The first phase of the CBR cycle consists of a retrieval past experience from the memory of cases, specifically those with a similar problem description to the current case. In order to carry out this process, a cosine similarity-based algorithm is applied, allowing the retrieval of those cases which are at least 90% similar to the current case. 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 retrieved 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 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). Figure 2 explains the four steps of CBR cycle, which incorporates a mixture of neural networks through an algorithm. This strategy of classification is carried out within the Anomaly CBR-BDI agent. This Anomaly CBR-BDI agent is located on a strategic level into the architecture.

```
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
```

Fig. 2. CBR Cycle Algorithm for classifying SQL query

Additionally, we mean 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 attack or not attack, 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 suspect 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, the mixture will as well be. If there is not concurrence, the system uses the result of the network with the least error in the training process or classifies the user request as suspect. In the reuse phase the two networks are trained by a back-propagation algorithm for the same set of training patterns (in particular, 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 y^2; where the superscript indicates the number of mixed networks

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

$$Error = \frac{1}{P} \sum_{i=1}^{P} \left| \frac{Forecast_P - T\arg et_P}{T\arg et_P} \right| \tag{2}$$

The number of neurons in the output layer for both Multilayer Perceptrons 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 with formula (2), where P is the total number of training patterns.

5 Results and Conclusions

SQL injection attacks on databases suppose a serious threat against information systems. This paper has presented a distributed hierarchical multi-agent architecture incorporating a novel type of agent based on the CBR-BDI model [8] specially designed for detecting and blocking SQL injection attacks. This CBR-BDI agent handles a great adaptation and learning capacities using a CBR mechanism. In addition, it incorporates the prediction capabilities that characterize neural networks. As a result, an innovative and robust solution has been presented allowing a significant reduction of the error rate during the classification to attacks and a different way to tackle SQL injection attacks using a distributed and hierarchical approach. To check the validity of the proposed model many tests were done. These tests were executed on a memory of cases, specifically developed to generate malicious queries. In Table 3 it is possible to observe 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 into account the same problem common to all the methods. Note that the technique proposed in this paper provides the best results, with an error in only 0.5% of the cases.

As shown in Table 3, the Bayesian is the most accurate statistical method since it is based on the likelihood of the events observed. But it needs determining the initial

Table 3. Results obtained after testing different classification techniques

Forecasting Techniques	Successful (%)	Approximated Time (secs)
Anomaly Agent (CBR-BDI)	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

parameters of the algorithm. Considering the errors obtained with the different methods, the Anomaly Agent and Bayesian methods provide the better results. Because of 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 behaviour: 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 it is considered a mixture of two neural networks, the predictions capabilities are remarkably improved. These methods are more accurate than statistical methods for detecting attacks to databases as the behaviour of the hacker is not linear, dynamic and chaotic.

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

References

1. Halfond, W., Orso, A.: AMNESIA: Analysis and Monitoring for Neutralizing SQL-injection Attacks. In: 20th IEEE/ACM international Conference on Automated software engineering, pp. 174–183. ACM, New York (2005)
2. Kosuga, Y., Kono, K., Hanaoka, M., Hishiyama, M., Takahama, Y.: Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection. In: 23rd Annual Computer Security Applications Conference, pp. 107–117. IEEE Computer Society, Los Alamitos (2007)
3. Valeur, F., Mutz, D., Vigna, G.: A Learning-Based Approach to the Detection of SQL Attacks. In: Conference on Detection of Intrusions and Malware and Vulnerability Assessment, Vienna, pp. 123–140 (2005)
4. Rietta, F.: Application layer intrusion detection for SQL injection. In: 44th annual Southeast regional conference, pp. 531–536. ACM, New York (2006)
5. Abraham, A., Jain, R., Thomas, J., Han, S.Y.: D-SCIDS: distributed soft computing intrusion detection system. Journal of Network and Computer Applications 30, 81–98 (2007)
6. Woolridge, M., Wooldridge, M.J.: Introduction to Multiagent Systems. John Wiley & Sons, Inc., New York (2002)
7. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Commun. 7, 39–59 (1994)

8. Laza, R., Pavon, R., Corchado, J.M.: A Reasoning Model for CBR_BDI Agents Using an Adaptable Fuzzy Inference System. In: Conejo, R., Urretavizcaya, M., Pérez-de-la-Cruz, J.-L. (eds.) CAEPIA/TTIA 2003. LNCS, vol. 3040, pp. 96–106. Springer, Heidelberg (2004)
9. Anley, C.: Advanced SQL Injection. In: SQL Server Applications (2002), `http://www.nextgenss.com/papers/advancedsqlinjection.pdf`
10. Christensen, A.S., Moller, A., Schwartzbach, M.I.: Precise Analysis of String Expressions. In: Cousot, R. (ed.) SAS 2003. LNCS, vol. 2694, pp. 1–18. Springer, Heidelberg (2003)
11. Su, Z., Wassermann, G.: The essence of command injection attacks in web applications. In: 33rd Annual Symposium on Principles of Programming Languages, pp. 372–382. ACM Press, New York (2006)
12. Huang, Y., Huang, S., Lin, T., Tsai, C.: Web application security assessment by fault injection and behavior monitoring. In: 12th international conference on World Wide Web, pp. 148–159. ACM, New York (2003)
13. Skaruz, J., Seredynski, F.: Recurrent neural networks towards detection of SQL attacks. In: 21th International Parallel and Distributed Processing Symposium, pp. 1–8. IEEE International, Los Alamitos (2007)
14. Ramasubramanian, P., Kannan, A.: Quickprop Neural Network Ensemble Forecasting a Database Intrusion Prediction System. In: 7th International Conference Artificial on Intelligence and Soft Computing, Neural Information Processing, vol. 5, pp. 847–852 (2004)
15. Corchado, J.M., Bajo, J., Abraham, A.: GerAmi: Improving Healthcare Delivery in Geriatric Residences. Intelligent Systems 23, 19–25 (2008)
16. Corchado, J.M., Bajo, J., de Paz, Y., Tapia, D.: Intelligent Environment for Monitoring Alzheimer Patients, Agent Technology for Health Care. Decision Support Systems 44(2), 382–396 (2008)
17. Corchado, J.M., Gonzalez-Bedia, M., De Paz, Y., Bajo, J., De Paz, J.F.: Replanning mechanism for deliberative agents in dynamic changing environments. Computational Intelligence 24(2), 77–107 (2008)