Mathematical model for a temporal-bounded classifier in security environments

JUAN F. DE PAZ, Departamento Informática y Automática, Universidad de Salamanca, Salamanca, Spain.
E-mail: fcofds@usal.es

MARTÍ NAVARRO, Departamento de Sistemas Informaticos y Computacion, Universidad Politécnica de Valencia, Valencia, Spain.
E-mail: mnavarro@dsic.upv.es

CRISTIAN I. PINZÓN, Facultad de Sistemas Computacionales, Centro Regional de Veraguas, Universidad Tecnológica de Panamá, Santiago de Veraguas, Panamá.
E-mail: cristian_ivanp@usal.es

VICENTE JULIÁN, Departamento de Sistemas Informaticos y Computacion, Universidad Politécnica de Valencia, Valencia, Spain.
E-mail: vinglada@dsic.upv.es

DANTE I. TAPIA, Departamento Informática y Automática, Universidad de Salamanca, Salamanca, Spain.
E-mail: dantetapia@usal.es

JAVIER BAJO, Departamento Informática y Automática, Universidad de Salamanca, Salamanca, Spain.
E-mail: jbajoipe@usal.es

Abstract

Security is a major concern when web applications are implemented. This has led to the proposal of a variety of specifications and approaches to provide the necessary security for these environments. SQL injection attacks on web applications have become one of the most important information security concerns over the past few years. The purpose of this article is to present an adaptive and intelligent mechanism that can handle SQL injection attacks taking into account a controlled time response. Our approach is based on a soft real-time classifier agent that incorporates a mixture of experts based on soft computing to choose a specific classification technique depending on the attack and the time available to solve the classification. A case study to evaluate the effectiveness of the approach and the preliminary results obtained with an initial prototype are also presented.

Keywords: Clustering, SOM, hierarchical clustering, PAM, dendrogram.

© The Author 2011. Published by Oxford University Press. All rights reserved.
For Permissions, please email: journals.permissions@oup.com
doi:10.1093/jigpal/jzr015
2 Mathematical model for a temporal-bounded classifier

1 Introduction

In recent years, Internet attacks have increased due to the large number of information systems connected to the Internet. One of the most serious security threats around Web application and databases has been the SQL injection attack [21]. This attack takes place at the database layer when a user request that has been sent through an HTTP request is executed without prior validation. Confidentiality, integrity and availability are the main objectives of any information security model [25]. Various approaches have attempted to deal with the problem of SQL injections [11, 17, 20, 26, 29]. However, the biggest inconvenience of these solutions is their inability to adapt to the rapid changes in attack patterns, which renders them a bit inefficient in the long term. None of the approaches consider the limitations or restrictions in response time remains a critical aspect in the majority of Internet security systems. With systems requiring a response to be given before a specific deadline, as determined by the system needs, it is essential that the execution time for each of the tasks carried out by the system is predictable and capable of guaranteeing correct execution within the time needed for the given response.

This study presents a new agent model with a novel perspective for analysing and classifying SQL injections in real time. The agent’s internal structure is composed of different techniques of soft computing integrated with a mixture of an artificial neural network (ANN) and a support vector machine (SVM), used as a classification mechanism. By using this mixture, it is possible to exploit the advantages of both strategies in order to classify the SQL queries in a more reliable way. The aim of the reasoning model is to obtain a soft computing system to facilitate real-time decision making in a robust manner and low solution cost [5, 8, 12]. The internal structure of the agent is based on the case-based reasoning (CBR) model, with the main difference being that the different CBR phases are time-bounded, thus enabling its use in real time. CBR can be very suitable for application in agent reasoning, where similar problems should have similar solutions. However, few of the existing approaches cope with the problem of applying CBR as deliberative engine for agents in MAS with soft real-time constraints. Additionally, the adaptation phase in the CBR system integrated in the agent proposes a new analysis classification model that is carried out by a mixture of experts. The concept of a mixture of experts was first proposed by [1]. Depending on the time available for performing classification, a set of experts is selected to perform the different analyses. The experts are selected with a multiple method model [23]. Finally, the different selected experts generate the predictions and the outputs are fused to generate a new unique result [18, 27].

The article is structured as follows: Section 2 presents the real-time agent (RTA) and CBR. Section 3 describes the SQL attack problem. Section 4 shows a general view of the temporal-bounded CBR used as deliberative mechanism in the classifier agent. Section 5 describes a set of tests to evaluate our proposal.

2 RTA and CBR

In environments where temporal constraints must be taken into account it is necessary to use specific agent models. In this type of environments, the validity of the solution is determined not only by its correct execution, but also by its ability to be carried out within the allotted time frame [19]. For this reason, we employ the notion of RTAs, which must give support to analyse and fulfil tasks taking into consideration their temporal conditions. Therefore, RTAs are specifically designed to be used in real-time environments [17]. The main difference in the architecture of a RTA with respect to other agent models is with the deliberation process. This process, which may use artificial intelligence (AI) techniques as problem-solving methods to compute more intelligent actions, must be temporal
constrained. In a typical AI technique, it is difficult to know the time required, because it can either be unbounded or have a high variability. If the agent has to operate in a real-time environment, the agent complexity required to achieve any or all of these features is greatly increased. Thus, a RTA requires an efficient integration of high-level, deliberative processes within reactive processes. When using AI methods, it is necessary to provide techniques that allow their response times to be bounded. These techniques are mainly based on well-known real-time artificial intelligence system (RTAIS) techniques [10]. Some examples of RTAs are as follows: the ARTIS agent specifically designed to develop real-time systems [3, 15], the ObjectAgent Architecture developed by Princeton Satellites in 2001 [15] and time-aware agents proposed by Prouskas et al. in 2002 [3].

If a RTA employs a CBR as its reasoning mechanism, the CBR phases must be temporal bounded to ensure that solutions are produced on time. In accordance with this, the real-time classifier agent used here incorporates a CBR specially designed to work in real-time environments, called temporal-bounded CBR (TB-CBR) [22]. Figure 1 shows the reasoning cycle for a TB-CBR system. The TB-CBR cycle starts at the learning stage, where it checks to see if there are previous cases waiting to be revised and possibly stored in the case-base. In our model, the plans provided at the end of the deliberative stage will be stored in a solution list while feedback about their utility is received. When each new TB-CBR cycle begins, this list is accessed. If there is enough time, the learning stage is implemented for those cases whose solution feedback has been recently received. If the list is empty, this process is omitted.

The next stage to be implemented is the deliberative stage. The retrieval algorithm is used to search the case-base and retrieve a case that is similar to the current case (i.e. one that characterizes the problem to be solved). Each time a similar case is found, it is sent to the reuse phase where it
Mathematical model for a temporal-bounded classifier

is transformed into a suitable plan for the current problem by using a reuse algorithm. Therefore, at the end of each iteration of the deliberative stage, the TB-CBR method is able to provide a plan for the problem at hand, although this plan can be improved in subsequent iterations if the deliberative stage has enough time to perform them.

To guarantee a correct temporal control of the execution, the temporal cost of executing the cognitive task must not exceed the execution time of the learning and deliberative stages (\(t_{\text{cognitiveTask}} \leq t_{\text{learning}} + t_{\text{deliberative}}\)). Moreover, the execution time of the deliberative and learning stages is related to the time necessary to execute the phases launched in the stage and the number of times that these phases are executed. (as shown in Equation 1):

\[
\begin{align*}
    t_{\text{learning}} &\geq (t_{\text{revise}} + t_{\text{retain}}) \cdot n \\
    t_{\text{deliberative}} &\geq (t_{\text{retrieve}} + t_{\text{reuse}}) \cdot m
\end{align*}
\]

where \(t_x\) is the execution time of the phase \(x\) (where \(x\) can be revise, retain retrieve or reuse) and \(n\) and \(m\) are the number of iterations of the learning and deliberative stages, respectively.

This algorithm can be launched when the RTA considers it appropriate and the RTA has enough time to execute it. The RTA indicates to the TB-CBR the maximum time \(t_{\text{max}}\) where \(t_{\text{max}} \geq t_{\text{cognitiveTask}}\) that is available to complete its execution cycle. This time must be divided between the two stages to guarantee their execution. The designer can assign more time to the learning stage if it desires a RTA with a greater capacity to learn.

3 SQL attack

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. A SQL injection attack can cause serious damage to an organization, including financial loss, breach of trust with clients, among others. There have been many proposed solutions for SQL injection attacks, including some AI techniques. One of the approaches is web application vulnerability and error scanner (WAVES) [20], a solution is based on a black-box technique that identifies vulnerable points, and then builds attacks that target those points based on a list of patterns and attack techniques. Valeur [29] presents an IDS approach that uses a machine learning technique based on a dataset of legal transactions. These are used during the training phase prior to monitoring and classifying malicious accesses. Generally, IDS systems depend on the quality of the training set; a poor training set would result in a large number of false positives and negatives. Skaruz [26] proposes the use of a recurrent neural network (RNN). The detection problem becomes a time serial prediction problem. The main problem with this approach is the large number of false positives and false negatives. Other strategies based on string analysis techniques and the generation of dynamic models have been proposed as solutions to SQL injection attacks. Halfond and Orso [17] propose analysis and monitoring for neutralizing SQL injection attacks (AMNESIA). Kosuga et al. proposes syntactic and semantic analysis for automated testing against SQL injection (SANIA) [11]. With only slight variations of accuracy in the models, these strategies have as drawback their meaningful rate of false positives and negatives.

4 Classifier mechanism

In this section, we present the CBR mechanism that allows to classify queries in SQL attack. The current classifiers can be grouped as: probabilistic models Naive Bayes [9], Fuzzy logic: K-NN
Mathematical model for a temporal-bounded classifier

(K-nearest neighbours), NN (K-nearest neighbours) [24], sequential minimal optimization (SMO) [28], decision trees and rules classification and regression trees (CART) [4] C4.5, C5.0/See5 [2] RIPPER [14]. It is necessary to study the execution time and the success rate for each of the techniques to use them in real time.

The fields defining a case are as follows: IdCase, Session, User, IP_Address, Query_SQL, Affected_table, Affected_field, Command_type, Word_GroupBy, Word_Having, Word_OrderBy, Numer_And, Numer_Or, Number_literals, Number_LOL, Length_SQL_String, Start_Time_Execution, End_Time_Execution, and Query_Category. Additionally to the information stored in the memory of cases, the system maintains a memory of models that contains information for each of the query types for each of the models. Moreover, the success rate and the execution time are stored, obtaining the following tuple:

\[ <(id, \{m_i\}, rate, time)> \]

where \(m_i\) is a classification model, id is the classification identifier, rate is the success rate and time is the time used for the classification.

The classifier mechanism presented in this study combines a SVM and a MLP in the reuse phase of a CBR system to classify SQL queries in real time. Next the different stages applied in the reasoning cycle are shown.

4.1 Retrieve

In this phase, the CBR retrieves the models used in previous experiences to classify similar queries. The models are generated offline using the retrieved cases for the type of query, thus it is not necessary to rebuild them in each iteration. In this way, the time used for the retrieval can be temporal bounded, since it is independent of the size of the memory of cases. Besides, the number of combinations is small and requires a low computational load.

The models for the SVM and MLP are recovered together with the classification time and success rate for each model. The recovery of these memory models allows the improvement of the system’s performance.

4.2 Reuse

In this phase, the time available for classifying the query is taken into account. The algorithm is selected in such a way that the success rate is maximized. For each of the classifiers its associated model is selected together with the success rate and the execution time. The retrieved cases are sorted regarding the success rate and the case with higher rate and execution time lower than the available time is selected. The SQL injection in our proposal can be analysed by two different techniques and their combination. Execution time in both cases is known, since previous stored models are used. The first is known as the light technique SVM and is usually a detection algorithm with a low temporal cost, but of low quality as well. Using the heavy technique, Multiplayer Perceptron, the result of the analysis is much more exact, but it requires a much higher amount of execution time. The inputs of the MLP are as follows: Query_SQL, Affected_table, Affected_field, Command_type, Word_GroupBy, Word_Having, Word_OrderBy, Numer_And, Numer_Or, Number_literals, Number_LOL and Length_SQL_String. The number of neurons in the hidden layer is \(2^n + 1\), where \(n\) is the number of neurons in the input layer. Finally, there is one neuron in the output layer. The activation function selected for the different layers has been the sigmoid. Taking into account the
activation function $f_j$, the calculation of output values are given by the following expression.

\[ y_j^p = f_j \left( \sum_{i=1}^{N} w_{ji} x_i^p(t) + \theta_j \right) \]  

(3)

where $w_{ji}$ represents the weight that joins $j$-th neuron in the hidden layer with $i$-th neuron in the input layer, $t$ is the time instant and $p$ the pattern in question. $x_i^p(t)$ is the $i$-th input value in the pattern $p$, $N$ the number of neurons in the input layers and $\theta_j$ is the bias. The calculation of the output in a neuron in the output layer is calculated in a similar way, taking the hidden layer as a input layer. As the neurons exiting from the hidden layer of the neural network contain sigmoidal neurons with values between $[0, 1]$, the incoming variables are redefined so that their range falls between $[0.2$ and $0.8]$.

The light algorithm SVM represents an extension of non-linear models [7]. SVM also allows the separation of element classes which are not linearly separable. To do so, the space of initial coordinates is mapped in a high dimensionality space through the use of functions. Due to the fact that the dimensionality of the new space can be very high, it is not feasible to calculate hyperplanes that allow the production of linear separability. For this reason, a series of non-linear functions called kernels is used.

Let us consider a set of patterns $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ where $x_i$ is a vector of the dimension $n$. The idea is to convert the elements $x_i$ in a space of high dimensionality through the application of a function, in such a way that the set of original patterns is converted into the following set $\Phi(T) = [(\Phi(x_1), y_1), (\Phi(x_2), y_2), \ldots, (\Phi(x_m), y_m)]$ that, depending on the selected function $\Phi(x)$, could be linearly separable. To carry out the classification, this equation sign is studied [22]:

\[ \text{class}(x_k) = \text{sign} \left( \sum_{i=1}^{m} \lambda_i y_i \Phi(x_i) \Phi(x_k) + b \right) \]  

(4)

where $\lambda_i$ is a Lagrange multiplier, $y_i$ output value for the pattern $x_i$, $b$ constant, these parameters are described in [16]. The selected kernel function in this problem was polynomial. The values used for the estimation are dominated by decision values and are related to the distance from the points to the hyperplane. The distances from the point to the hyperplane are calculated using 4.3.

\[ d(x; w, b) = \frac{|w \cdot \Phi(x) + b|}{\|w\|} \]  

(5)

If there is enough time for both techniques to be carried out, the mixture is performed defining a pondered average for both techniques related to the success rate. To obtain the pondered average, the values are transformed into the $[0, 1]$ range, with 0 as legal and 1 a attack. Before this transformation, the extreme atypical values are deleted, conserving the values in the following interval. $Q_i$ is the quartile $i$.

\[ [Q_1 - 3(Q_3 - Q_1), Q_1 + 3(Q_3 - Q_1)] \]  

(6)

4.3 Revise and retain

The revise phase can be manual or automatic depending on the output values. The automatic review is given for non-suspicious cases during the estimation obtained for the reuse phase. For cases
detected as suspicious, with output values determined experimentally in the interval [0.35, 0.6], a review by a human expert is performed.

The learning phase updates the information of the new classified case and reconstructs the classifiers offline to leave the system available for new classifications. The ANN classifier is reconstructed only when an erroneous classification is produced. In the case of a reference to inspection of suspicious queries, information and classifiers are updated when the expert updates the information.

5 Results and conclusions

This article has presented a novel proposal for detecting SQL injections in soft real-time environments, where the possible violation of temporal constraints results in a degraded quality of the attack analysis but the system can continue to operate. The article proposes a new vision in which each attack mechanism is individually analysed. It also makes it possible to obtain better classification results with regard to both the effectiveness of the classification process and the response time, since all classification mechanism tasks are temporally bounded. In order to validate the initial prototype, we proposes a benchmark case study that contains 705 SQL queries (437 legal queries and 268 attacks). The entries were automated by using the SQLMap 0.6.3 tool, with which an initial case base was established for training the SQL-TB-CBR Classifier. Prior to initiating the tests, the attack classification mechanisms were analysed for each use of a light or heavy technique and other classifiers. To analyse the successful rates, a test of the classification of queries was conducted, taking into account the following classifiers: Bayesian Network, Naive Bayes, AdaBoost M1, Bagging, DecisionStump, J48, JRIP, LMT, Logistic, LogitBoost, MultiBoosting AdaBoost, OneR, SMO and Mixture. The different classifiers were applied to 705 previously classified queries. Figure 2 shows the number of errors obtained for each of the soft computing techniques and the proposed model based on a mixture taking into account the suspicious queries (legal queries with values up to 0.5 plus illegal queries with values lower than 0.5), false negative lower than 0.35, false positive up to 0.6, suspicious with values in the range [0.35, 0.6].

The analysis demonstrated that the use of mixture proposed in this study provided a better classification, but with a greater temporal cost. The average execution time for the queries,
and the worst time used for the light and heavy techniques were, respectively, 0.013/0.051 and 0.28/1.07 ms.

For the second test, a set of 50 queries were selected and then classified according to different pre-determined deadlines. The number of executions and errors obtained for each of the classifiers are shown in Figure 3. The x-axis represents the average time between queries and the deadline, while the y-axis represents the number of queries executed. As can be seen, the number of executions for the mixture increased as the execution time between queries increased.

The proposed SQL-TB-CBR agent is capable of detecting SQL injections with low error rates compared with other existing techniques, as shown in Figure 3. Moreover, it is possible to provide a soft real-time classifier mechanism, with a high level of confidence to identify legal queries and attacks. The combination of different AI paradigms allows the development of a hybrid intelligent system with characteristics such as the capacity for learning and reasoning, flexibility and robustness which make the detection of SQL injection attacks possible.

**Funding**


**References**


Mathematical model for a temporal-bounded classifier


Mathematical model for a temporal-bounded classifier


Received 18 October 2010