AIIDA-SQL: An Adaptive Intelligent Intrusion Detector Agent for Detecting SQL Injection Attacks

Cristian Pinzón
Faculty of Computer Systems Engineering
Technological University of Panama
Panama City, Panama
c.pinzon.t@acm.org

Juan F. De Paz, Javier Bajo
Departamento de Informática y Automática
Universidad de Salamanca
Salamanca, Spain
fcofds@usal.es, jbajope@usal.es

Álvaro Herrero
Department of Civil Engineering
University of Burgos
Burgos, Spain
ahcosio@ubu.es

Emilio Corchado
Departamento de Informática y Automática
Universidad de Salamanca
Salamanca, Spain
escorchado@usal.es

Abstract— **SQL Injection attacks on web applications have become one of the most important information security concerns over the past few years. This paper presents a hybrid approach based on the Adaptive Intelligent Intrusion Detector Agent (AIIDA-SQL) for the detection of those attacks. The AIIDA-SQL agent incorporates a Case-Based Reasoning (CBR) engine which is equipped with learning and adaptation capabilities for the classification of SQL queries and detection of malicious user requests. To carry out the tasks of attack classification and detection, the agent incorporates advanced algorithms in the reasoning cycle stages. Concretely, an innovative classification model based on a mixture of an Artificial Neuronal Network together with a Support Vector Machine is applied in the reuse stage of the CBR cycle. This strategy enables to classify the received SQL queries in a reliable way. Finally, a projection neural technique is incorporated, which notably eases the revision stage carried out by human experts in the case of suspicious queries. The experimental results obtained on a real-traffic case study show that AIIDA-SQL performs remarkably well in practice.**

Agent, Case-Based Reasoning, Support Vector Machine, Artificial Neural Network, SQL Injection, Intrusion Detection

I. 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 applications and databases has been the SQL Injection attack [1]. This attack can be used to gain confidential information, to bypass authentication mechanisms, to modify the database, and to execute arbitrary code [2]. A large number of solutions has been proposed so far [3], [4], [5], [6], [7], [8], [9], but they can not defend against sophisticated attack techniques such as those that use alternate encodings and database commands to dynamically construct SQL strings. These approaches lack the learning and adaptation capabilities for dealing with attacks and their possible variations in the future. In addition, a limitation of the majority of these solutions is that they are based on centralized mechanisms with little capacity to work in distributed and dynamic environments.

This paper introduces the hybrid intelligent agent named AIIDA-SQL (Adaptive Intelligent Intrusion Detector Agent), designed according to the strategy of an Intrusion Detection System (IDS) and aimed at detecting SQL injection attacks. AIIDA-SQL is characterized by the integration of a Case-Based Reasoning (CBR) engine which provides it with a great level of adaptation and learning capability, since CBR systems make use of past experiences to solve new problems [10]. This is very effective for blocking SQL injection attacks as the ID mechanism uses a strategy based on anomaly detection [11].

In addition to the CBR motor in the AIIDA-SQL agent's internal structure, an integrated mixture of an Artificial Neural Network (ANN) and a Support Vector Machine (SVM) is 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. Finally, to assist the expert in the making of decisions regarding those queries classified as suspicious, a visualization mechanism is proposed which combines clustering techniques and unsupervised neural models to reduce the dimensionality of the data.

AIIDA-SQL is the principal component of a distributed hierarchical multi-agent system (MAS) aimed at detecting different attacks in dynamic and distributed environments.

The rest of this paper is structured as follows: section 2 describes the problem that has prompted most of this research work. Section 3 explains the internal structure of the AIIDA-SQL agent used as a classifier and visualizer agent. Finally, the conclusions and experimental results of this work are presented in section 4.

978-1-4244-7365-6/10/$26.00

II. SQL INJECTION ATTACKS

An SQL injection attack takes place when a hacker changes the semantic or syntactic logic of an SQL text string by inserting SQL keywords or special symbols within the original SQL command which is executed at the database layer of an application [1]. Different attack techniques exist which include the use of SQL tautologies, logic errors / illegal queries, union query and piggy-backed queries. Other more advanced techniques use injection based on interference and alternative codification [1]. The cause of the SQL injection attacks is relatively simple: an inadequate input validation on the user interface. As a result of this attack, a hacker can be responsible for unauthorized data handling, retrieval of confidential information, and in the worst possible case, taking over control of the application server [1].

Different strategies have been presented as a solution to the problem of SQL injection attacks [1], with special attention given to strategies based on IDSs [2], [3], [4], [5], [6], [7], [8], [9]. One approach based on anomaly detection was proposed by [2], applying a clustering strategy to group similar queries and isolate queries which are considered malicious. The main disadvantage of this approach is in its high computational overhead which would affect a real-time detection. Kemalis and Tzouramanis propose SQL-IDS (SQL Injection Detection System) [4] that uses security specifications to capture the syntactic structure of the SQL queries generated by the applications. The main limitation of this approach is the computational cost while comparing the new query with the predefined structure at runtime.

In [5] two types of SQL injection attacks are raised: tautology attacks and those based on the UNION operator. Through the syntactic analysis of SQL query strings, the data of the HTTP requests are extracted to later be used in the training phase and to determine the threshold to use in the evaluation phase. Bertino, Kamra and Early [5] propose an anomaly detection mechanism applying data mining techniques. The main problem of this approach is to find an adequate threshold to maintain a low rate of both false positives and false negatives. Another anomaly-based approach is proposed by Robertson, Vigna, Kruegel and Kemmerer [7]. The approach uses generalisation techniques to convert suspicious requests within abnormal signatures. These signatures are later used to group malicious requests which present similar characteristics. Another of the used techniques is characterization, that deduces the type of attack associated with a malicious request. A low computational overhead is generated. However, it is susceptible to generating false positives. The algorithm ID3, presented by Garcia, Monroy and Quintana [8], proposes the detection of attacks targeted at web applications. The algorithm ID3 is used to detect and filter malicious SQL strings, presenting a significant percentage of incorrect classifications. Valeur, Mutz, and Vigna [9] propose the use of anomaly detection through the generation of a series of models beginning with a set of recovered queries. At execution time, they monitor the applications in order to identify requests which are not associated with the aforementioned models.

III. AN ADAPTIVE INTELLIGENT AGENT FOR DETECTING SQL INJECTION ATTACKS

Agents are characterized by their autonomy, which gives them the ability to work independently and in real-time environments [12]. The AIIDA-SQL agent presented in this study interacts with other agents within an architecture still under development. These agents carry out tasks related to message capture and syntactic analysis, administration, and user interaction. As opposed to the tasks performed by these agents, the AIIDA-SQL agent executes classification SQL queries that we will subsequently define in greater detail.

CBR is a paradigm based on the idea that similar problems have similar solutions. Thus, a new problem is resolved by consulting the case memory to find a similar case which has been resolved in the past. When working with this type of systems, 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 that depicts the initial problem; a solution that describes the sequence of actions performed in order to solve the problem; and the final state, which describes the state that was achieved once the solution was applied.

As previously mentioned, the AIIDA-SQL agent is an specialization of a CBR agent which is the key component of a multi-agent architecture and is geared towards classifying SQL queries for the detection of SQL injection attacks. Below, a new classification mechanism incorporated in the internal structure of the AIIDA-SQL agent is explained in detail.

A. AIIDA-SQL Agent

In this section the AIIDA-SQL agent is presented, with special attention paid to its internal structure and the classification mechanism of SQL attacks. This mechanism combines the advantages of CBR systems, such as learning and adaptation, with the predictive capabilities of a combination of ANNs and SVMs. The use of this mixture of techniques is based on the possibility of using two classifiers together to detect suspicious queries in the most reliable way.

In terms of CBR, a case is composed of elements of the analysed SQL query described as follows:

- **Problem**: describes the initial information available for generating a plan. The problem description consists of: case identification, user session and SQL query elements.
- **Solution**: states the action carried out in order to solve the problem (in this case, the applied prediction models).
- Final State: describes the state achieved after applying the solution.

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, the information related to the used prediction models is stored as well.

In Fig. 1, the different stages applied in the reasoning cycle can be seen.

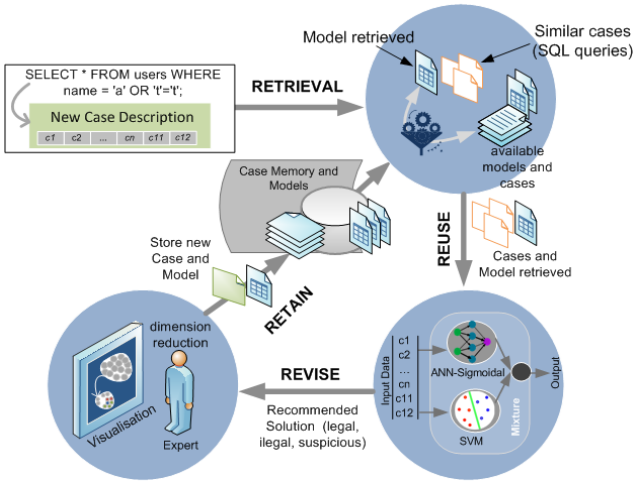

Figure 1. CBR cycle and classification mechanism of the AIIDA-SQL agent.

In the retrieval stage, there is a selection of queries sorted by their type and the memory classification models. In the reuse phase, as seen in Fig. 1, a Multilayer Perceptron (MLP) and an SVM are simultaneously applied to carry out the prediction of the new query. Subsequently, a new inspection is performed which can be done automatically or by a human expert. In the case of the query resulting as suspicious, further visual inspection will be carried out by a human expert. At this stage, the most similar cases are selected by means of a Growing Cell Structures (GCS) network [13], and visualized by the unsupervised neuronal model named Cooperative Maximum Likelihood Hebbian Learning (CMLHL). As a result, the human expert will graphically see the relationship between the suspicious query and the recovered most similar queries. During learning, memory information regarding the cases and models will be updated. Below, the different stages of the CBR reasoning cycle associated with the system are described in more detail.

1) Retrieve

The retrieval phase is broken down into two phases; case retrieval and model retrieval. Case retrieval is performed by using the *Query_Category* attribute which retrieves queries from the case memory (C_r) which were used for a similar query in accordance with attributes of the new case c_n. Subsequently, the models for the MLP (mlp_r) and SVM (svm_r) associated with the recovered cases are retrieved. The recovery of these memory models allows the improvement of the system's performance so that the time necessary for the creation of models will be considerably reduced, mainly in the case of the ANN training.

2) Reuse

The reuse phase inputs are the information of the retrieved cases C_r and the recovered models mlp_r and svm_r. The combination of MLPs and SVMs is fundamental in the reduction of the false negative rate. The inputs of the MLP are: *Query_SQL, Affected_table, Affected_field, Command_type, Word_GroupBy, Word_Having, Word_OrderBy, Numer_And, Numer_Or, Number_literals, Number_LOL*, and *Length_SQL_String*. The number of neurons in the hidden layer is $2n+1$, where n is the number of neurons in the input layer. Finally, there is one neuron in the output layer. The 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.

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

The outputs correspond to x^r. 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, 0.8]. This transformation is necessary because the network does not deal with values that fall outside of this range. The outgoing values are similarly limited to the range of [0.2, 0.8] with the value of 0.2 corresponding to a non-attack and the value of 0.8 corresponding to an attack. The network training is carried out through the error Backpropagation algorithm [14].

At the same time as the estimation through neuronal networks is performed, it is also carried out by the SVM model, a supervised learning technique applied to the classification and regression of elements. The algorithm represents an extension of nonlinear models [15]. Additionally, SVM 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 $T = \{(x_1, y_1),(x_2, y_2),...,(x_m, y_m)\}$ 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 $\Phi(T) = \{(\Phi(x_1), y_1),(\Phi(x_2), y_2),...,(\Phi(x_m), y_m)\}$ that, depending on the selected function $\Phi(x)$, could be linearly separable. To carry out the classification, this equation sign is studied [16]:

$$class(x_k) = sign\left(\sum_{i=1}^{m} \lambda_i y_i \Phi(x_i)\Phi(x_k) + b\right) \quad (2)$$

The selected kernel function in this problem was polynomial. The values used for the estimation are dominated by decision values and are related to the distance from the points to the hyperplane.

Once the output values for the ANN and the SVM are obtained, the mixture is performed by means of a weighted average of the error rate of each one of the techniques. A weighted average is applied to take into account the error rate of each technique. If different classifications are obtained from each technique, the query would then be classified as suspicious, and subsequent revision would be launched. Before carrying out the average, the values are normalized to the interval [0, 1], as SVM provides positive and negative values and those of greater magnitude.

3) Revise

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 experimentally determined in the interval [0.35, 0.6], a review by a human expert is visually performed. As CBR learns, the interval values are automatically adjusted to the smallest of the false negatives. The greater limit is constantly maintained throughout the iterations. The review consists of recovering queries that are the most similar to the current one together with their previous classifications. This combines a clustering technique for the selection of similar requests with a neuronal model for the reduction of dimensionality, which permits visualisation in 2D or 3D as shown in Figs. 3 and 4.

The selection of similar cases is carried out through the use of a neural GCS network. The different cases are distributed in meshes and the mesh containing the new case is selected. To visualize the similar cases (those in the selected mesh), the dimensionality of data is reduced by means of the CMLHL neural model [17] which performs Exploratory Projection Pursuit by unsupervised learning. Considering an N-dimensional input vector (x), and an M-dimensional output vector (y), with W_{ij} being the weight (linking input j to output i), then CMLHL can be expressed as:

Feed-forward step:

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

Lateral activation passing:

$$y_i(t+1) = \left[y_i(t) + \tau(b - Ay) \right]^+ \tag{4}$$

Feedback step:

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

Weight update:

$$\Delta W_{ij} = \eta . y_i . sign(e_j) | e_j |^{p-1} \tag{6}$$

Where: η is the learning rate, τ is the "strength" of the lateral connections, b the bias parameter, p a parameter related to the energy function [15], [16] and A is a symmetric matrix used to modify the response to the data [15]. The effect of this matrix is based on the relation between the distances separating the output neurons.

Finally, the information is represented and the associated queries are recovered with the retrieved mesh, as can be seen in Figs. 3 and 4.

4) Retain

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

IV. Experimental Results and Conclusions

To empirically test the proposed approach with a real-life dataset, legal queries were sent from a designed user interfaces of a made-to-measure web application. SQLMAP 0.5 [18] was used to automatically dispatch malicious queries. This tool is able to fingerprint an extensive DBMS back-end and retrieve remote DBMS databases.

To compare the success rates, a test on the classification of queries was conducted by applying the following classifiers: Bayesian Network, Naive Bayes, AdaBoost M1, Bagging, DecisionStump, J48, JRIP, LMT, Logistic, LogitBoost, MultiBoosting AdaBoost, OneR, SMO, and Stacking. The different classifiers were applied to 705 previously classified queries (437 legal queries and 268 attacks). The consecutive process to carry out the output test was the following: selecting one of the cases, extracting it from the set, conducting the model starting from the remaining cases and classifying the extracted case. This process is repeated for each one of the cases and techniques in order to analyze each query without it being used to build the model. The performance of the different classifiers can be seen in Fig. 2.

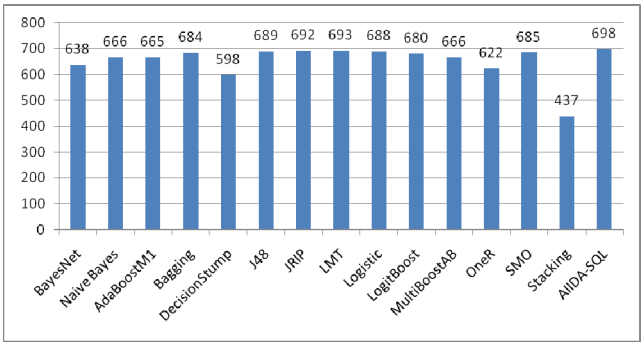

Figure 2. Total number of hits for the different classifiers.

As can be seen in Fig. 2, the highest-performance system is AIIDA-SQL, whith a success rate of 698/705.

As an example, an illegal query has been selected from the existing ones:

Select TBorder_customer.id_order, brands, code, name, price from TBorder_brands, TBorder_customer, TBproduct where TBorder_customer.id_order = TBorder_brands.id_order and TBproduct.code = TBorder_brands.code and TBorder_customer.id_order = 1 AND ORD(MID((SELECT 7 FROM information_schema.TABLES LIMIT 0, 1), 1, 1)) > 63 order by brands

The query is a clear case of SQL injection so it should be classified as an attack. The system classifies this query as illegal since an output value of 0.84 is given by the classifiers mixture. Fig. 3 shows the provided visualization of queries if the manual revise phase would have been carried out. The most similar queries are recovered and depicted in different colours: legal queries are shown in green, attacks in red and the analysed query in blue. Non-recovered queries are shown in grey. The above detailed illegal query is closer to the group of queries in red, so the human expert would clearly consider this query as an illegal one.

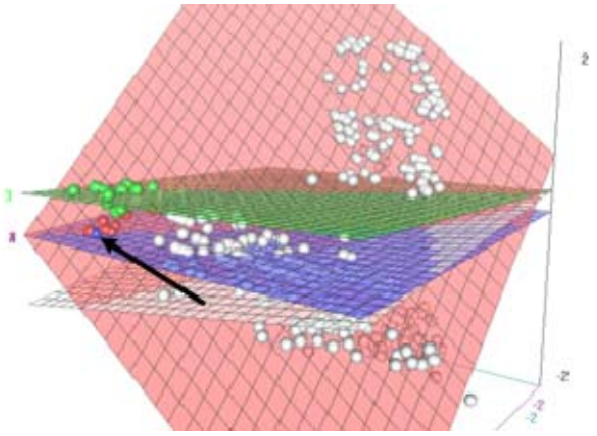

Figure 3. Visualization of SQL queries in the revise stage.

The number of queries detected as suspicious in the test of Fig. 1 was limited to seven, being one of them the following query:

Update table TBcustomer set name = 'Jose', lastname = 'Perez', city = 'Murcia', province = 'Murcia', ZC = 37008 where id = test' or id = 'test2'

This query is not a clear attack because it is usual to have a literal in the right side of the *or* operator, and it can be legal or suspicious.

The value obtained by the ANN for this query was 0.24. However, SVM deemed that the output value was 0.48. The mixture gave an output value of 0.36, which is in the range of suspicious queries. If the ANN would have been applied alone it would have considered this query as a valid one. However, the SVM would have considered it as an attack. The mixture deemed suspicious and a review was carried out manually.

During the visual review, similar queries were recovered and the data dimensionality was reduced to perform 3D visualization. The obtained result to be shown to the human expert is depicted in Fig. 4. At first glance, it can be determined that the query is a legal one viewing the position with respect to the similar queries.

Two examples of the similar queries recovered for the manual review are shown below.

Update table TBorder_customer set state = 'P' AND ORD(MID((SELECT 4 FROM information_schema.TABLES LIMIT 0, 1), 2, 1)) > 1 AND '1'='1' where date between '2008-05-05 00:00:00' and '2008-05-05 23:23:23'

Update table TBcustomer set name = 'Jose', lastname = 'Perez', city = 'Murcia', province= 'Murcia', ZC = 37008 where id = 'test'

The first one is a clear attack, while the second is not.

As a conclusion, it can be said that the combination of different Artificial Intelligence 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. The proposed AIIDA-SQL agent is capable of detecting these abnormal situations with low error rates compared with other existing techniques, as shown in Fig. 2. There are not strong differences in the performance of the best techniques, but in the case of computer security only one false negative is very important as it could extremely damage the protected system. Some other techniques could be applied in the mixture to improve the performance. The presented approach also provides a decision mechanism which eases the review of suspicious queries through the selection of similar queries and their visualization by means of an unsupervised neural model. For future work, we will thoroughly study the query clustering process in order to take into account the characteristics of queries with greatest influence in SQL injections.

ACKNOWLEDGMENT

This development has been partially supported by the Spanish Ministry of Science and Technology project OVAMAH: TIN 2009-13839-C03-03, Junta of Castilla and León (JCyL) project: [BU006A08], the projects of the Spanish Ministry of Education and Innovation [CIT-020000-2008-2] and [CIT-020000-2009-12] CIT-020000-2009-12 (funded by the European Regional Development Fund), project of the Spanish Ministry of Science and Innovation TIN2010-21272-C02-01 (funded by the European Regional Development Fund)and Grupo Antolin Ingenieria, S.A., within the framework of project MAGNO2008 - 1028.- CENIT also funded by the same Government Ministry, and the Professional Excellence Program 2006-2010 IFARHU-SENACYT-Panama.

REFERENCES

[1] W. G. J. Halfond, *et al.*, "A Classification of SQL-Injection Attacks and Countermeasures," in *Proceedings of the IEEE International Symposium on Secure Software Engineering*, Arlington, VA, USA, 2006.

[2] A. Asmawi, Sidek Zailani Mohamed Razak Shukor Abd, "System architecture for SQL injection and insider misuse detection system for DBMS," in *International Symposium on Information Technology (ITSim'2008)*, 2008, pp. 1 -6.

[3] C. Bockermann, *et al.*, "Learning SQL for Database Intrusion Detection Using Context-Sensitive Modelling (Extended Abstract)," in *6th International Conference on Detection of Intrusions and*

Malware, and Vulnerability Assessment (DIMVA '09), Berlin, Heidelberg, 2009, pp. 196--205.

[4] K. Kemalis and T. Tzouramanis, "SQL-IDS: a specification-based approach for SQL-injection detection," in *Proceedings of the 2008 ACM symposium on Applied computing (SAC'2008)*, New York, NY, USA, 2008, pp. 2153--2158.

[5] M. Kiani, *et al.*, "Evaluation of Anomaly Based Character Distribution Models in the Detection of SQL Injection Attacks," in *Third International Conference on Availability, Reliability and Security (ARES'2008)*, Washington, DC, USA, 2008, pp. 47--55.

[6] E. Bertino, *et al.*, "Profiling Database Applications to Detect SQL Injection Attacks," in *Proceedings of the Performance, Computing, and Communications Conference (IPCCC'2007)*, 2007, pp. 449-458.

[7] W. Robertson, *et al.*, "Using Generalization and Characterization Techniques in the Anomaly-Based Detection of Web Attacks," in *13th Annual Network and Distributed System Security Symposium (NDSS'2006)*, 2006.

[8] V. H. García, *et al.*, "Web Attack Detection Using ID3," in *International Federation for Information Processing* 2006, pp. 323-332.

[9] F. Valeur, *et al.*, "A Learning-Based Approach to the Detection of SQL Attacks," in *Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, Vienna, Austria, 2005, pp. 123--140.

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

[11] S. Mukkamala, *et al.*, "Intrusion detection using an ensemble of intelligent paradigms," *Journal of Network and Computer Applications,* vol. 28, pp. 167--182, 2005.

[12] C. Carrascosa, *et al.*, "Hybrid multi-agent architecture as a real-time problem-solving model," *Expert Systems with Applications,* vol. 34, pp. 2--17, 2008.

[13] B. Fritzke, "A Growing Neural Gas Network Learns Topologies," in *Advances in Neural Information Processing Systems 7*, 1995, pp. 625--632.

[14] Y. LeCun, *et al.*, "Efficient BackProp," in *Neural Networks: Tricks of the Trade*, 1998, p. 546.

[15] E. Corchado and C. Fyfe, "Connectionist Techniques for the Identification and Suppression of Interfering Underlying Factors," *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 17, pp. 1447-1466, Dec 2003.

[16] E. Corchado, *et al.*, "Maximum and Minimum Likelihood Hebbian Learning for Exploratory Projection Pursuit," *Data Mining and Knowledge Discovery,* vol. 8, pp. 203-225, May 2004.

[17] Á. Herrero, *et al.*, "DIPKIP: A Connectionist Knowledge Management System to Identify Knowledge Deficits in Practical Cases," *Computational Intelligence,* vol. 26, pp. 26-56, 2010.

[18] B. Damele, "SQLMAP0.5 – Automated SQL Injection Tool," ed, 2007

.

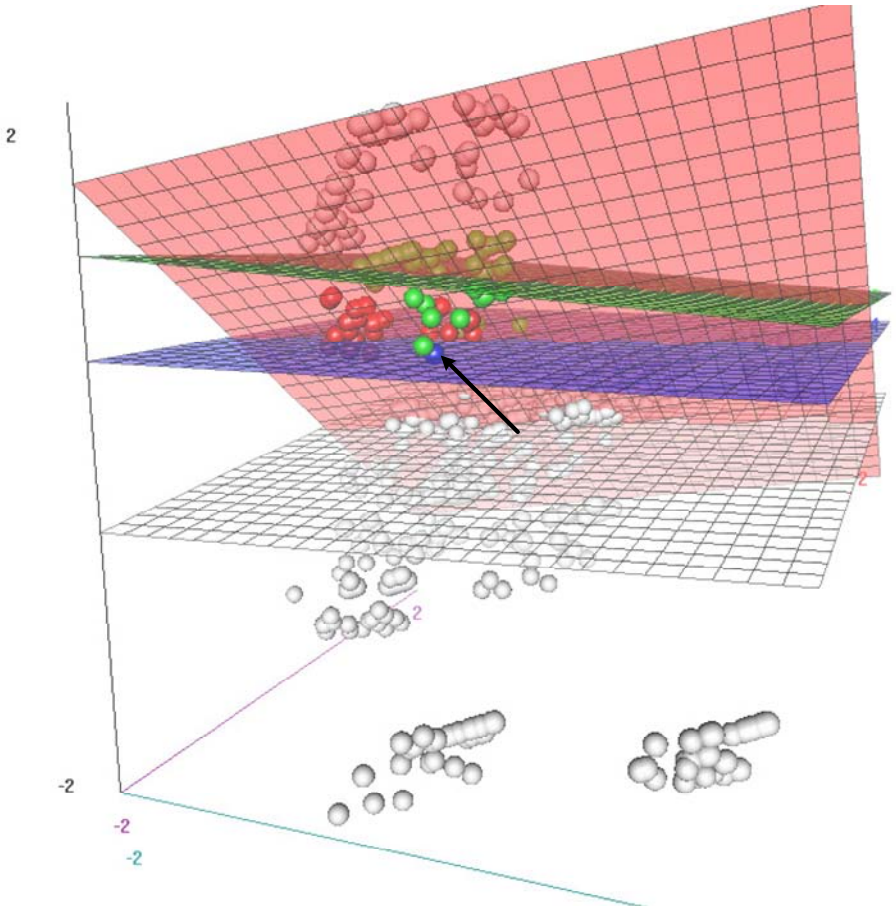

Figure 4. SQL queries recovered in the revise stage.