Detection of SQL Injection Using a Genetic Fuzzy Classifier System

Christine Basta, Ahmed elfatatry, Saad Darwish
Information Technology department
Institute of Graduate Studies and Research
Alexandria, Egypt

***Abstract*—SQL Injection (SQLI) is one of the most popular vulnerabilities of web applications. The consequences of SQL injection attack include the possibility of stealing sensitive information or bypassing authentication procedures. SQL injection attacks have different forms and variations. One difficulty in detecting malicious attacks is that such attacks do not have a specific pattern. A new fuzzy rule-based classification system (FBRCS) can tackle the requirements of the current stage of security measures. This paper proposes a genetic fuzzy system for detection of SQLI where not only the accuracy is a priority, but also the learning and the flexibility of the obtained rules. To create the rules having high generalization capabilities, our algorithm builds on initial rules, data-dependent parameters, and an enhancing function that modifies the rule evaluation measures. The enhancing function helps to assess the candidate rules more effectively based on decision subspace. The proposed system has been evaluated using a number of well-known data sets. Results show a significant enhancement in the detection procedure.**

***Keywords*—SQL injection; web security; genetic fuzzy system; fuzzy rule learning**

I. Introduction

Web applications are vulnerable to numerous attacks. SQL injection is a widely common threat, which remains on top of the list of web application attacks as ranked by OWASP (the Open Web Application Security Project) [1]. Various techniques of SQL injection are used by hackers to achieve different purposes: bypassing a login system, modifying a table in a database, shutting down SQL server, getting database information from the returned error message, or executing stored procedures [2].

SQL injection attacks are a type of vulnerability that is ultimately caused by insufficient input validation. Such attacks occur when data provided by the user is not properly validated and included directly in an SQL query. By leveraging these vulnerabilities, an attacker can submit SQL commands directly to the database. Web applications are threatened by this kind of vulnerability that uses user input to form SQL queries to access an underlying database [3]. Generally, SQL injection attacks are classified into seven types: tautologies, illegal/logically incorrect queries, piggy-backed queries, stored queries, inference and alternate encodings [2] [4] [5].

Attackers continuously develop new ways to bypass controls added by developers. In the recent years, hackers started to use different styles to perform SQLI. Hackers developed techniques to bypass web application firewall (WAF bypassing). The security agents started to use buffer overflow methods and applied new bypassing methods like special characters bypassing. The various types of injections at different levels require a solution that can cope with such changes.

A number of approaches address detection of SQLI attacks. Such approaches include static analysis, dynamic analysis, and combined approach. Researchers developed other approaches like mutation based approach, query tokenization and applying regular expressions. These approaches suffer from a number of problems preventing them from being the optimal solutions [6]. Those techniques lack flexibility and scalability; they cannot deal with unknown types or larger ranges of injections [7]. Lack of learning capabilities is a vital problem. Most solutions parse user input and confirm match limited to fixed and very small patterns, which are modeled by reference to existing malicious web code. However, there are new malicious web codes which can deliberately be developed to avoid being matched with the registered patterns [8]. The available parsing techniques can also cause high computational overhead affecting real-time detection [9].

Recently, machine learning techniques are adapted to overcome previously mentioned problems as they can give leverage for the broader range of malicious web code and can be adapted to variations and changes [8]. Machine learning techniques explore the study and construction of algorithms that can learn from and make predictions on data. Such algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions, rather than following strictly static program instructions [10]. Some existing machine learning techniques suffer from high computational overhead; the training of classifiers in those techniques is time-consuming and causes computational overhead. Furthermore, a number of existing solutions lack adaptation capability to detect new attacks [9].

Uncertainty and fuzziness are popular phenomena in applications of machine learning. Different types of uncertainty can be observed: (i) Noise, outliers, and errors affect the input data. A machine learning method has to deal with this type of fuzzy information, showing robustness with respect to such disturbances. (ii) Distribution and fuzziness influence representation of information within a machine learning system. According to these different locations and goals of fuzzy information, a variety of different models exist which allow machine learning to deal with uncertain information as input, output, or internal representation [11]. Fuzzy rule-based systems (FRBSs) are well-known methods within soft computing, based on fuzzy concepts that address complex real-world problems. They are powerful methods to address uncertainty, imprecision, and non-linearity [12].

Fuzzy rule-based classification systems (FRBCSs) are specialized in handling classification tasks. A main characteristic of classification is that the outputs are categorical data. Therefore, in this model type, we preserve the antecedent part of linguistic variables and change the consequent part to be a class C_j from a pre-specified class set $C = \{C_1,.......,C_M\}$. FRBCS aim at representing the knowledge of human experts in a set of fuzzy IF-THEN rules. Instead of using crisp sets as in classical rules, fuzzy rules use fuzzy sets. Rules were initially derived from human experts through knowledge engineering processes. However, this approach may not be feasible when facing complex tasks or when human experts are not available. An effective alternative is to generate the FRBCS model automatically from data by using learning methods. FRBCSs have demonstrated their ability to handle control problems, modeling, classification or data mining in a huge number of applications [13].

The automatic definition of FRBCS rules can be seen as an optimization problem. Genetic Algorithms (GAs) are global search techniques with the ability to explore a large search space for suitable solutions only requiring a performance measure. In addition to their ability to find near optimal solutions in complex search spaces, the generic code structure and independent performance features of GAs qualifies them to incorporate a priori knowledge. In the case of FRBCSs, this a priori knowledge may be in the form of linguistic variables, fuzzy membership function parameters, fuzzy rules, number of rules (Genetic rule learning), etc. These capabilities extended the use of GAs in the development of a wide range of approaches for designing FRBSs over the last few years. Therefore, GAs remain today as one of the fewest knowledge schemes available to design and optimize FRBCSs with respect to the design decisions. According to the performance measures, decision makers decide which components are fixed and which need to change [13].

In this work, we investigate the FRBCS technique for detection of SQLI; we suggest a new technique to address the uncertainty, fuzziness and adaptation problems associated with existing machine learning techniques. The rule selection mechanism in FRBCS induces competition among rules by only considering the quality of matching performed by each rule. To increase the generalization power of the classifier, we have proposed a genetic fuzzy approach that creates more cooperative rules in the final population. The proposed system uses genetic algorithm (GA) for optimizing the FRBCS technique to enhance its learning and adaptation capabilities.

The rest of the paper is organized as follows: Section 2 discusses related work reported in the literature. An overview of the proposed fuzzy genetic system is explained in Section 3. The experimental result and evaluation of the proposed system are discussed in Section 4. Finally, in Section 5 the conclusion and future research directions are presented.

II. Literature Review and Related Work

In general, SQL injection attacks can be divided into the three main categories: in-band, out-of-band and inferential [14]. In the in-band attacks, the information is extracted from the same channel that is used for the attack. For example, the list of users will appear in the current page. In out-of-band attack, the extracted information is sent back to the attacker using another channel such as email. For inferential, which is also known as a blind injection, no data is sent back directly to the attacker. However, the attacker can reconstruct the data by trying the different attacks and observing the behavior of the web application.

In the literature, SQLI detection techniques can be classified into the dynamic analysis, static analysis, combined approach, machine learning, and other approaches (e.g. Hash technique, Black Box Testing) [3][15-19]. Static analysis checks whether every flow from a source to a sink is subject to an input validation and/or input sanitizing routine [20]; whereas dynamic analysis is based on dynamically mining the programmer's intended query structure on any input and detects attacks by comparing it against the structure of the actual query issued [21].

AMNESIA, as a combined approach, is a model-based technique that combines the static and dynamic analysis for detection and prevention of SQLI attacks [3]. In the static phase, to build the models of the SQL queries that are generated at points of access to the database, AMNESIA uses a static analysis. In the dynamic phase, AMNESIA intercepts 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 identified as SQLI attacks. The accuracy of AMNESIA depends on the static analysis stage. Unfortunately, certain types of complicated codes and/or query generation techniques make this step less precise and generate both false positives and negatives [22].

As mentioned above, several approaches for detection of SQL injection were developed. The literature survey emphasizes on the machine learning techniques which are relevant to our proposed system. Valeur et al. [23] proposed an intrusion detection system capable of detecting a variety of SQL injection attacks. Profiles of normal access to the database are built using statistical methods. At runtime, queries that do not match any built model are identified as a possible attack. As with most learning-based anomaly detection techniques, the system requires a training phase prior to detection. The main problem of this technique besides the false positives and negatives is its execution and storage overhead, due to difficulty in training on all the possible normal benign queries with normal behavior [24].

In [9], the authors proposed an SQLI detection technique in adversarial environments by K-centers. They introduced a new online learning technique in which samples are learned one by one, and as a result, number and centers of the clusters are adjusted accordingly. Therefore, the K-centred technique can adapt to different kinds of attacks. The experimental results show that their method has a satisfying result on the SQLI attacks detection in the adversarial environment. The main drawback of their method is that it must receive a true label of each statement after classification [25]. The concept of pattern classifiers to detect injection attacks and protect web applications is introduced in [24]. HTTP requests are captured and converted into numeric attributes. Numeric attributes include the length and the number of keywords of parameters. Using these attributes, the system classifies the parameters by Bayesian classifier to judge whether the parameters are injection patterns or not. The main drawback is that the system depends on limited types of features.

The major contributions of the work in [26] are the proposal of a novel method based on the genetic algorithm applied to SQLI attack detection task and correlation of a number of detection tools altogether with the novel method. In this work, the authors prove that correlating several sources of information and then performing reasoning on the correlated information can improve the results of attacks detection. The main disadvantage of this algorithm is the overhead in performance and storage caused by the correlation approach.

The implementation of Artificial Neural Networks (ANN) as a biologically inspired computing is investigated in [2] to detect SQLI attacks. Multilayer Feed forward Networks (MLN) was used in the implemented system. It has the ability to learn and store the empirical knowledge, the nonlinearity nature of the neural networks, the ability to generalize the solutions and to adapt when the context changes, and suitable computational performance. The limitations include depending on the appearance of certain SQL keywords along with suspicious characters without considering the relative order between them. For this reason, despite the different order of the keywords, if a normal signature contains many keywords and suspicious characters that often appear together in an SQLI, it is highly likely to be misclassified. Another work, related to ANN-based SQLI detection, is introduced in [27, 28]. It depends on limited SQL patterns for training so it is susceptible to generate false positives.

TF-IDF has been used in [8] for weight calculation of tokens to evaluate the performance of three machine learning approaches: SVM, Naive-Bayes, and K-NN. This method has low computation time complexity but susceptible to generating false positives [9]. Furthermore, Gene Expression Programming (GEP) for detection of SQLI is discussed in [29]. At the beginning, chromosomes are generated randomly. Then, in each iteration of GEP, a linear chromosome is expressed in the form of expression tree and executed. The fitness value is calculated and termination condition is checked. The best individual is preserved through the next iteration. Afterward, the populations are subjected to genetic operators with defined probability. New individuals in temporary population constitute the current population. Classification accuracy received from GEP depicts great efficiency for SQL queries constituted from 10 to 15 tokens. For longer statements, the averaged FP and FN is approximately 23%.

Among the approaches, genetic algorithm for detection of SQLI is proposed in [30]. In this technique, levels of SQLI are detected using template matching. The ultimate goal of the genetic algorithm is to optimize the matching rules of SQLI queue in the template library. These rules are in the form of IF (condition) THEN (execution); where conditions refer to attack sequence matches. However, the algorithm relies on template sequence to define SQLIA. Therefore, the system fails to detect the attacks of different sequences that are not included in the template library.

The main objective of this paper is to propose a combined approach where FRBS and the genetic algorithm can be used together to improve the accuracy of the system for detection of SQLI, consequently, new SQLI attacks can be processed and detected. To the best of our knowledge, there is no previous work that uses FRBS for detecting SQLI attacks. To enhance the accuracy of learning capability, we extend FRBS with the genetic algorithm to find the most suitable rules for FRBCS.

III. PROPOSED SQL INJECTION DETECTION SYSTEM USING FUZZY GENETIC

This paper introduces a GA based method to generate a fuzzy rule base for SQLI detection. With the specific structure of the chromosome, the GA operations and the adequate fitness function, the proposed method produces a fuzzy rule base (FRB) with proper rules. Designers usually cannot guarantee that the fuzzy control system designed with trial-and-error for building fuzzy rules has a reliable performance. Fig. 1 illustrates the flow diagram of the proposed system.

In this work, the fuzzy rule base is tuned automatically by GA, known as Genetic Fuzzy System (GFS). The fuzzy logic produces controllers that are suitable for dealing with uncertainty and imprecision. Second, fuzzy behaviors can be conveniently synthesized by a set of IF-THEN rules using easy-to-understand linguistic terms to encode expert knowledge. Finally, the interpolative nature of fuzzy systems helps express partial and simultaneous simulations of SQLI features, and the smooth transitions between these features [30].

GA starts with a population of randomly generated chromosomes, and advance towards better chromosomes by applying genetic operators inspired by the genetic process occurring in nature. The population undergoes evolution in a form of natural selection. During successive iterations, called generation, chromosomes in the population are evaluated for their adaptation as solutions, and on the basis of this evaluation, a new population of chromosomes is formed using a selection mechanism, crossover, and mutation operators. A fitness function must be devised for each problem to be solved. Each chromosome is evaluated using the fitness function, returning a single numerical value. The probability of selection of a certain chromosome is directly proportional to its fitness function [31]. A GA-tuned fuzzy system with seven inputs and one output will be illustrated to explain the SQLI detection process.

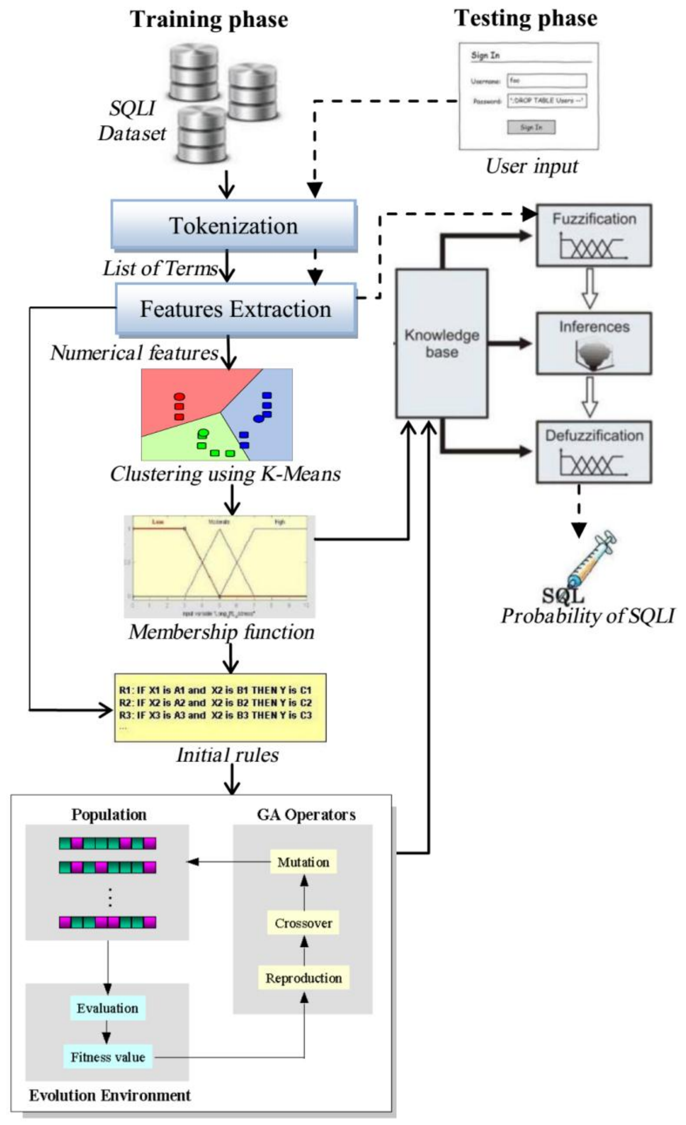

Fig. 1. Proposed SQL Injection Detection System

A. *Extracting SQLI features from dataset*

The SQLI attack keywords are the popular keywords in SQL language which are generally used in order to perform operations on the tables inside a given SQL database. This approach extracts tokens (keywords) which consist of specific terms of malicious web code as features. Tokenization is a process of breaking a sentence into a list of words. In other words, a tokenizer parses a sentence into a list of tokens. Based on these tokens, the system extract the features, and represent each query as a sequence of numbers, each number represents one of the features mentioned below [2] [8] [9] [27] [26].

- f_1 : Frequency of special characters (dangerous characters) like (--, #, /*, ', ", ||, \\, =, /**/,@@).
- f_2 : Frequency of special tokens (dangerous tokens) like (rename, drop, delete, insert, create, exec, update, union, set, Alter, database, and, or, information_schema, load_file, select, shutdown, cmdshell, hex, ascii).
- f_3 : Frequency of punctuations like (<, >, *, ; , _, -, (,), =, {, }, @, ., , &, [,], +, -, ?, %, !, :, \, /).
- f_4 : Frequency of SQL tokens like (where, table, like, select, update, and, or, set, like, in, having, values, into, alter, as, create, revoke, deny, convert, exec, concat, char, tuncat, ASCII, any, asc, desc, check, group by, order by, delete from, insert into, drop table, union, join).
- f_5 : Length of SQL statement.
- f_6 : Frequency of spaces within the parameter of the query, which leads to the possibility of attacks.
- f_7 : Existence of statements that always result in true value, for example "1=1" or "@=@" or "124=124".

The appropriate selection of these features plays a crucial role in several aspects of the design of robust and feasible SQLI detection systems. The rationale for choosing these types of features is its ability to identify most of SQIA types like tautologies, union, piggybacked, illegal/logically incorrect, alternate encodings and stored procedures which are treated the same as SQL queries. Other features can be included to increase the scalability of the system to detect new malicious code. In addition, by reducing the number of features (by eliminating redundant or irrelevant features), the performance of rule induction system can be improved as well as the classification performance of the rules produced.

B. *K-means clustering*

To transfer the extracted numerical features (all mentioned above features except the 7th feature into linguistic terms low (*L*), medium (*M*) and high (*H*); the system utilizes K-means clustering algorithm. k-means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed Apriori. Let $X=\{x_1,x_2,\ldots,x_n\}$ be the set of data points, that represents the $f_i(i=1,\ldots 6)$ values across the queries and $V=\{v_1,v_2,\ldots,v_n\}$ be the set of initial centers. Algorithmic steps [32, 33] for *k*-means clustering are:

1) Randomly select k cluster centers, k =3 in our case.

2) Calculate the distance between each data point and cluster centers.

3) Assign the data point to the cluster center whose distance from the cluster center is the minimum of all the cluster centers.

4) Recalculate the new cluster center:

$v_i=\left(\frac{1}{c_i}\right)\sum_{j=1}^{c_i}x_i$, where c_i represents the number of data points in ith cluster.

5) Recalculate the distance between each data point and the new obtained cluster centers. If no data point is reassigned then stop, otherwise repeat from step 3. This algorithm aims at minimizing an objective function, in this case, a squared error function.

$$J(V) = \sum_{i=1}^{k} \sum_{j=1}^{c_i} (\|x_i - v_j\|)^2 \ , \qquad (1)$$

c_i is the number of data points in i_{th} cluster, and k is the number of cluster centers.

C. Fuzzy Logic System

Owing to low computational requirement and capability of modeling human perception, fuzzy Logic (FL) is an efficient and flexible method for managing degrees of uncertainty in attack detection. Problems can be described in natural descriptions, linguistic terms, rather than the numerical values. The FL system consists of (i) fuzzifier that takes input values and determines the degree to which they belong to each of the fuzzy sets via membership functions (MFs); (ii) fuzzy inference system that defines a non-linear mapping of the input data vector into a scalar output, using fuzzy rules and (3) defuzzifier that maps output fuzzy sets into a crisp number [34]. A fuzzy set [35] is defined as [2]:

$$D = \{(x, \mu_D(x)) \mid x \in X, \mu_D(x) \in [0,1]\}, \qquad (2)$$

where X represents the universal set, x is an element of X, D is a fuzzy subset in X and $\mu_D(x)$ is the membership function of fuzzy set D. A membership function is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1 [35].

Next we will fuzzify the input (features of SQLI) and the output (probability of injection), i.e. input and output are mapped into a set of fuzzy partitions. Here, a seven-input single-output fuzzy system is used, which is given by $f : U \subset R^m \to Z \subset R^n$, where $U = U_1 \times .... \times U_7$ is the input space and Z is the output space. Three fuzzy variables including 'low', 'medium' and 'high' (L, M, H) are used to describe the features. Their respective MFs (μ_A) [36] are triangular function calculated as:

$$f(x; a, c) = \max(\min(\frac{x-a}{b-a}, \frac{c-x}{c-b}), 0) \ , \qquad (3)$$

where a, b and c are the outputs of the k-mean clustering that represent lower, center and upper limits of a cluster respectively. To achieve overlap between the membership functions (overlapped fuzzy-sets) of each feature, the system makes an intersection with 15% -20% between the consecutive MFs.

Once the system acquires the fuzzy descriptions of the features distance, the Mamdani rule base (fuzzy reasoning) can be built to make an inference of detection of SQLI. Fuzzy reasoning, which is formulated by the group of fuzzy IF–THEN rules, presents a degree of presence or absence of association or interaction between the elements of two or more sets. In the proposed system, reasoning is carried out through the following rules:

- If more than half input variables are '*H*', the output variable is set to '*H*'.
- If both f_7 and f_2 are '*H*', the output variable is set to '*H*'.
- If both f_7 and f_1 are '*H*', the output variable is set to '*H*'.
- If both f_1 and f_2 are '*H*', the output variable is set to '*H*'.
- If f_7 , f_2 and f_1 are '*H*', the output variable is set to '*H*'.
- If any of f_7 , f_2 and f_1 is '*H*', the output variable is set to '*M*'.
- If both f_1 and f_2 are 'M', the output variable is set to '*M*'.

Other rules are obtained using the Cartesian product method of the seven features; which is to consider all the combinations of antecedent linguistic values and generate a fuzzy rule for each combination. The output variable of each case depends on the nature of dataset. The rules altogether deal with the weight assignments impliedly in the same way that humans think. The fuzzy inference processes all of the cases in a parallel manner, which makes the decision more reasonable.

The output of the fuzzy system is the probability of SQLI (PSQLI) and it is also described by three fuzzy variables, including 'high', 'medium' and 'low' with triangular MFs. The outputs of fuzzy values are then defuzzified to generate a crisp value for the variable. The most popular defuzzification method is the centroid, which calculates and returns the center of gravity of the aggregated fuzzy set [36] and is given by

$$\theta = \frac{\sum_{r=1}^{s} \mu^{(r)} \theta^{(r)}}{\sum_{r=1}^{s} \mu^{(r)}} \ , \qquad (4)$$

where $\theta^{(r)}$ is the center of the suggested output at rule r, n is the number of rules and $\mu^{(r)}$ is the MF at rule r. The obtained crisp value is then mapped to its range (low, medium, high) to indicate the potential of SQLI attack.

D. Rule Induction using Genetic Algorithm

In general, a rule base can be constructed by human experts or by machine learning techniques from datasets. The machine learning approach is useful where it is desired to extract rules from the analysis that can be related to conceivable human behavior. The essential feature of a GA is that a population of proposed solutions (coded using a "chromosome") is modified using biologically inspired operators (especially crossover and mutation), and incorporating a random component, to explore a solution space [37]. Formally, let $P(g)$ and $S(g)$ be parents and offspring in generation g; the GA is working as follows:

TABLE I. GENETIC PARAMETERS

Parameter	***Value***	***Description***
Population type	Bit string	Input data type to the fitness function
The initial population	Randomly	Cartesian product of all features
Population size	200 -1458	Number of chromosomes in each generation
Number of generation	Satisfying criteria	The highest ranking solution's fitness is reaching
Gene length	2 bits for all features + 1 bit for feature7 + 2 bits for the output	Length of unsigned bit string for each variable
Chromosome length	15 bits	(2×6)+1+2
Probability of cross-over	0.5	Default, combine two parents to form children in the next generation
Probability of mutation	0.015	Default, apply random changes to individual parents to form children
Selection function	Tournament	A number Tour of individuals is chosen randomly from the population and the best individual from this group is selected as parent.
Fitness function	Maximize attack detection	The objective is to maximize detection ratio between parents and children.

```
Procedure (GA)
    BEGIN
        $g \leftarrow 0$
          Initialize P(g)
          Evaluate P(g)
          While (not matching the ending conditions)
          Recombine P(g) to yield S(g)
          Evaluate S(g)
          Select P(g+1) from P(g) and S(g)
        $g \leftarrow g+1$
    END
```

In general, the methods that combine the genetic and fuzzy approaches for generation of knowledge bases (KBs) can be divided into two main groups: genetic tuning and genetic learning [13]. If there exists a KB, we apply a genetic tuning process for improving the FRBS performance while preserving the existing RB. That is, to adjust FRBS parameters for improving its performance, maintaining the same RB. The second possibility is to learn KB components (an adaptive inference engine can be included). That is, to involve the learning of KB components among other FRBS components. Our system employs the genetic learning to learn the flexible inference engine.

The first step in applying GAs to the problem of rule learning is to map the initial rules (initial RB) into a suitable representation for genetic operations. The system that has been used is Michigan GA [13]. In Michigan GA, the population consists of multiple individuals, each individual codifies single rule, and the whole rule set is provided by combining several individuals in a population. For this problem, the variables (genes) are the linguistic values of each feature (low, medium, high). Many trials of different values of GA parameters were performed. Finally, the best-evaluated values of parameters were chosen as mentioned in Table I. The codification scheme and the fitness calculation are described below.

1) Chromosome Codification

The pre-selection of candidate rules (initial RB) used here allows each rule to be uniquely identified. The identification induces a simple binary codification of each rule in each chromosome and, consequently, the use of simple processes to create and handle the chromosomes. Fig. 2 illustrates a chromosome with 15 bits, represented in binary system, where each consecutive two bits from the position 1 to position 12 indicate a feature f_i (i=1 to 6) with "00" for low, "01" for medium and "11" for high linguistic term; whereas the bit at position 13 identifies a feature f_7 with values "0" to non-existing and "1" for existing and finally the last two bits represents PSQLI with "00" for low, "01" for medium and "11" for high linguistic term.

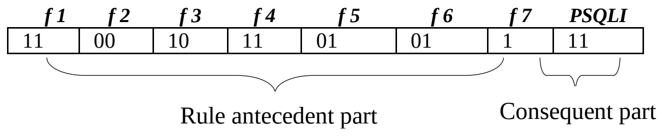

Fig. 2. Example of codification of rules

2) Fitness Function

In order to use genetic algorithms as the search procedure, it is necessary to define a fitness function which properly assesses the rules. The fitness function must be able to discriminate between the legal and illegal classification of queries. Finding an appropriate function is not a trivial task, due to the variations associated with the SQLI types. In general, the quality of rules in an FRBS is one of the parameters that favor accuracy, while the number of rules is the parameter that favors transparency, an FRBS with a small number of rules can make the model easily understood by the user. Several approaches in the field of FRBS reduce the FRB size at the expense of accuracy [38]. In this work, the primary objective is to enhance the accuracy; therefore, the numbers of rules are left as they are.

In this work, the method suggested by the authors in [39] is refined to improve the evaluation step performed by the GA in order to optimize the rules in the final FRB during the search process. The fitness value is calculated using the Correct Classification Rate (CCR) represented by each chromosome.

$$\text{Fitness} = \text{CCR} = \frac{\alpha}{A} - \frac{\beta}{B}, \tag{5}$$

where A is the total number of attack records, B is the total number of normal records, α is a total number of attack records correctly identified as attack and β is the total number of normal records incorrectly classified as attack (False positive).

IV. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed system, a desktop application that integrates Java and MATLAB has been implemented. The dataset is downloaded from testbed [40] which is used to evaluate Amnesia approach in [3]. The test bed has two sets of inputs: "legit" set, which consists of legitimate inputs for the application, and "attack" set, which consists of attempted SQLIAs. All types of attacks were represented in this set except for multi-phase attacks. The multi-phase attacks include inference attacks and illegal/logically incorrect queries, such attacks require human intervention and interpretation. The testbed includes seven folders, three of them are used for training and the rest for testing. The result is analyzed using: (1) True negative (*TN*), which means the label of the SQL statement is normal and the classifier has classified it as normal. (2) True positive (*TP*) means that the classified label is the same as the true label which is abnormal. (3) False negative (*FN*) means that the classifier has made a classification mistake concerning the abnormal SQL statements. (4) False positive (*FP*), which means that a normal statement is misclassified to be an abnormal statement [9]. The tests are conducted according to Table I, and the fuzzy parameters are set as membership function type is triangular with 20% overlap.

The first experiment was conducted on the three different training sets to investigate the performance of the proposed system under both original Cartesian rules and enhanced version. The original Cartesian rules were the initial rules of the GA formed by all combinations of the seven features that have been previously mentioned in Section 3. Whereas the enhanced version contains the Cartesian version plus seven rules that contain only three features (f_1, f_2, and f_7), which have the higher impact on SQLI detection. Each other individual rule in the original Cartesian set contains all the seven features together.

TABLE II. RESULTS OF SQLI DETECTION OF TRAINING SET

Training Folder's subject	***No. of attacks URLS***	***No. of legit URLS***	***Original Cartesian Rules (1458 rules)***		***Enhanced Cartesian rules (1465 rules)***	
			TP%	***FP%***	***TP%***	***FP%***
Bookstore	3033	608	76.5	0	95.8	0.5
Checkers	3442	1359	67.3	0	96.8	1.2
Classifieds	3346	576	69.8	0	95.0	0

TABLE III. RESULTS OF SQLI DETECTION OF TESTING SET

Testing Folder's subject	***No. of attacks URLS***	***No. of legit URLS***	***Original Cartesian Rules (1458 rules)***		***Enhanced Cartesian rules (1465 rules)***	
			TP%	***FP%***	***TP%***	***FP%***
Events	3002	900	89.7	0	100	1.5
Employee	3497	660	89.9	0	100	0.6
Office Talk	3612	424	72.6	0	94.2	1.1
Portal	2968	1080	90.8	0	100	2.6

TABLE IV. RESULTS OF SQLI DETECTION OF TESTING SET (TOTAL OF 13079 ATTACKS, 3016 LEGIT URLS OF THE FOUR TESTING FOLDERS)

Type of population	***Population size***	***Membership function of 20% intersection***		***Membership function of 15% intersection***	
		TP%	FP%	TP%	FP%
Random	200	79.7	0	76.5	0
Random	600	81.8	0	77.6	0
Random	1000	82.6	0	80.3	0
Original Cartesian	1458	84.5	0	82.8	0
Enhanced Cartesian	1465	98.38	1.6	98	1.7

Table II indicates that the system can achieve high accuracy using the enhanced Cartesian rule with 96.8% *TP* for the attack set and 1.2% *FP* for the legit set of the Checkers dataset. The enhanced Cartesian rules improve the detection accuracy of attacks with a slightly increased false positive rate for the legit set. One reason for this increase is the fact that using the enhanced version activates some rules concerned with f1 which exists with high frequency in normal URLs. For example, the frequency of a character like '=' can be high in normal URLs.

When compared to the results in Table III, the enhanced Cartesian rules achieve higher accuracy in detection of SQLI in the three folders of the testing set with average increase of 3% *TP* and average decrease of 1% *TN*. One explanation of such result is that there are some attack URLs like "Password='&ret_page=""&querystring='" resulted after preprocessing stage in the training set. Such URLs do not consider most of SQLI features; thus they are not matched with any rule resulting in false negative (wrong classification). Regarding the Office Talk testing set, it has been noticed that the obtained results have the same range as the results of the training set due to the similarity of the URLs structure.

From the obtained results in table IV, it is confirmed that the proposed system provides good accuracy on the subject of increasing population size (number of initial rules). In general, increasing the population size leads to increasing the diversity of chromosomes. This diversity results in new offspring's in each generation that gives rise to increasing accuracy. However, increasing the number of rules increases the overhead. Furthermore, the experiments reveal that the system's accuracy can be improved as the level of fuzzification increases inside the membership function (intersection area). In the case of increasing the intersection by 5%, the accuracy changed by 2-4%. One explanation for this result is that the system's ability to deal with uncertainty is directly proportional with increasing the fuzzification level. Increasing the intersection level between membership functions for each feature will increase the ability of the fuzzy logic system to infer the result from the activated rules through fuzzy logic operation (min, and max).

TABLE V. COMPARATIVE STUDY

Methods	***Correct Responses***
The proposed system	98.4%
Neural Network system [28]	96.8%

In the last experiment, the accuracy (correct responses) of the proposed system that employs genetic fuzzy algorithm to detect SQLI and the comparative algorithm suggested by N. Sheykhkanloo [28] is given in Table V. The comparative system utilizes an effective pattern recognition Neural Network (NN) model for detection and classification of the SQLI attacks. From the illustrated results, our system outperforms the other one by 1.5%. In general, the correct responses of the neural network system depend mainly on the number of hidden layers that is commonly determined by the user. On contrast, the accuracy of our system depends on the number of utilized features and consequently the constructed rules, which characterize the flexibility factor for our system. Furthermore, the initialization parameters of GA can affect the performance of our system. These parameters are configured in the learning phase (offline processing), which consumes more time. In the testing phase (online processing), the computation time for detection is reasonable and acceptable. The duration is about 217 sec for 13079 attacks, i.e. 16.6 *ms* for one attack.

V. CONCLUSION

In this article, we proposed a genetic- fuzzy rule-based classification system for the SQLI attack detection. In the proposed system, the SQL statement is treated as a feature vector that characterizes the SQLI attack keywords. The genetic algorithm is adapted for FRBCS to improve the quality of matching implemented by each rule by means of adjusting FRBS parameters to increase the generalization power of the classifier. The quality of the proposed system depends mainly on the selection of the attributes used to build the feature vector. It has the potential to give a higher accuracy when comparing to other solutions that use SQL keywords separately for the problem of SQL injection. For new patterns that the proposed system cannot recognize, the system can be retrained so that it can detect the new patterns without a significant increase in processing time. The solution can be used in combination with positive logic based filtering as in the prototype implementation. We have presented the evaluation methodology and reported the results that prove that: (i) The proposed method outperforms other state-of-the art NN method. (ii) Enhanced Cartesian of GA population type improves detection results. Future work includes investigating more features to enhance the performance of the detection, and the ability to automatically reduce the number of rules in the rule base to improve the detection.

REFERENCES

[1] OWASP. (1 November, 2015). O.W.A.S.P. Top 10 Vulnerabilities. Available: https://www.owasp.org/index.php/Top_10_2013-Top_10.

[2] A. Moosa, "Artificial neural network based web application firewall for SQL injection," World Academy of Science, Engineering & Technology, vol. 64, no. 4, pp. 12-21, April 2010.

[3] W. G. Halfond and A. Orso, "AMNESIA: analysis and monitoring for neutralizing SQL-injection attacks, " Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, pp. 174-183, 2005.

[4] W. Y. Win and H. H. Htun, "Detection of SQL Injection Attacks by Combining Static Analysis and Runtime Validation," International Conference on Advances in Engineering and Technology, pp. 95-99, Singapore, March 2014.

[5] V. Nithya, R. Regan, and J. Vijayaraghavan, "A survey on SQL injection attacks, their detection and prevention techniques," International Journal of Engineering and Computer Science, vol. 2, issue 4, pp. 886-905, 2013.

[6] A. Tajpour, S. Ibrahim, and M. Masrom, "SQL injection detection and prevention techniques," International Journal of Advancements in Computing Technology, vol. 3, no. 7, pp. 82-91, 2011.

[7] S. Rohilla and P. K. Mittal, "Database security by preventing SQL injection attacks in stored procedures," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 11, pp. 915-919, 2013.

[8] R. Komiya, I. Paik, and M. Hisada, "Classification of malicious web code by machine learning," 3rd International Conference on Awareness Science and Technology, pp. 406-411, China, Sept. 2011.

[9] X.-R. Wu and P. P. Chan, "SQL injection attacks detection in adversarial environments by K-centers," International Conference on Machine Learning and Cybernetics, pp. 406-410, China, 2012.

[10] J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, "An overview of machine learning," in Machine learning, ed: Springer, pp. 3-23, 1983

[11] B. Hammer and T. Villmann, "How to process uncertainty in machine learning?," European Symposium on Artificial Neural Networks, pp. 79-90, Belgium, 2007.

[12] L. S. Riza, C. Bergmeir, F. Herrera and J. M. Benítez, "Frbs: fuzzy rule-based systems for classification and regression in R," Journal of Statistical Software, vol. 65, no. 1, pp. 1-30, 2015.

[13] F. Herrera, "Genetic fuzzy systems: taxonomy, current research trends and prospects," Evolutionary Intelligence, vol. 1, no. 1, pp. 27-46, 2008.

[14] A. Sadeghian, M. Zamani and S. Ibrahim, "SQL injection is still alive: a study on SQL injection signature evasion techniques," International Conference on Informatics and Creative Multimedia, pp. 265-268, Malaysia, 2013.

[15] G. Wassermann and Z. Su, "An analysis framework for security in Web applications," Proceedings of the FSE Workshop on Specification and Verification of component-Based Systems, pp. 70-78, 2004

[16] G. Buehrer, B. W. Weide and P. A. Sivilotti, "Using parse tree validation to prevent SQL injection attacks," ACM Proceedings of the 5th International Workshop on Software Engineering and Middleware, pp. 106-113, 2005.

[17] S. Bandhakavi, P. Bisht, P. Madhusudan, and V. Venkatakrishnan, "CANDID: preventing sql injection attacks using dynamic candidate evaluations," Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 12-24, 2007.

[18] S. Ali, A. Rauf, and H. Javed, "Sqlipa: An authentication mechanism against sql injection," European Journal of Scientific Research, vol. 38, no. 4, pp. 604-611, 2009.

[19] Y. Huang, S. Huang, T. Lin, and C. Tsai, "Web application security assessment by fault injection and behavior monitoring," Proceedings of the 12th international conference on World Wide Web, pp. 148-159, Hungary, 2003.

[20] L. K. Shar and H. B. K. Tan, "Defeating SQL injection", Computer, vol. 46, no. 3, pp. 69-77, March 2013.

[21] A. Tajpour and M. JorJor Zade Shooshtari, "Evaluation of SQL injection detection and prevention techniques," Second IEEE International Conference on Computational Intelligence, Communication Systems and Networks, pp. 216-221, UK, 2010.

[22] R. Dharam and S. G. Shiva, "Runtime monitors to detect and prevent union query based SQL injection attacks," Tenth International Conference on Information Technology: New Generations, pp. 357-362, USA, 2013.

[23] F. Valeur, D. Mutz and G. Vigna, "A learning-based approach to the detection of SQL attacks," Proceedings of the Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 123-140, Austria, 2005.

[24] E. H. Cheon, Z. Huang, and Y. S. Lee, "Preventing SQL injection attack based on machine learning," International Journal of Advancements in Computing Technology, vol. 5, issue 9, pp. 967-974, 2013.

[25] M. Kaushik and G. Ojha, "SQL injection attack detection and prevention methods: a critical review," International Journal of Innovative Research in Science, Engineering and Technology, vol. 3, issue 4, pp. 11370-11377, April 2014.

[26] M. Choraś, R. Kozik, D. Puchalski, and W. Hołubowicz, "Correlation approach for sql injection attacks detection," Advances in Intelligent Systems and Computing, Springer, vol. 189, pp. 177-185, 2013.

[27] N. M. Sheykhkanloo, "Employing neural networks for the detection of sql injection attack," Proceedings of the 7th International Conference on Security of Information and Networks, pp. 318-323, UK, 2014.

[28] N. M. Sheykhkanloo, "SQL-IDS: evaluation of SQLi attack detection and classification based on machine learning techniques," Proceedings of the 8th International Conference on Security of Information and Networks, pp. 258-266, USA, 2015.

[29] J. Skaruz, J. P. Nowacki, A. Drabik, F. Seredynski and P. Bouvry, "Soft computing techniques for intrusion detection of SQL-based attacks," Lecture Notes in Computer Science,Springer, Vol. 5990, pp. 33-42, 2010.

[30] J. Chen, L. Yang, H. Zhang, and Y. Liu, "A GA-based approach for SQL-injection detection", Future Information Engineering, vol. 49, p. 291, 2014.

[31] A. Adriansyah and S. H. M. Amin, "Knowledge base tuning using genetic algorithm for fuzzy behavior-based autonomous mobile robot," Proceeding of 9th International Conference on Mechatronics Technology, pp. 120-125, Malaysia, 2005.

[32] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 881-892, 2002.

[33] E. Forgey, "Cluster analysis of multivariate data: Efficiency vs. interpretability of classification", Biometrics, vol. 21, no. 3, pp. 768-769, 1965.

[34] S. M. Saad, "Application of fuzzy logic and genetic algorithm in biometric text-independent writer identification", IET Information Security, vol. 5, no.1, pp. 1-9, 2011.

[35] I. Elamvazuthi, P. Vasant, and J. F. Webb, "The application of mamdani fuzzy model for auto zoom function of a digital camera", arXiv preprint arXiv:1001.2279, 2010.

[36] M. Abdulghafour, "Image segmentation using fuzzy logic and genetic algorithms," Journal of WSCG, vol. 11,no. 1, pp.1-8, 2003.

[37] J. Ricketts, "Tuning a modified Mamdani fuzzy rulebase system with a genetic algorithm for travel decisions," 18th World IMACS / MODSIM Congress, Australia, pp. 768-774, 2009.

[38] M. E. Cintra and H. D. A. Camargo, "Fuzzy rules generation using genetic algorithms with self-adaptive selection," IEEE International Conference on Information Reuse and Integration, pp. 261-266, USA, 2007.

[39] R. B. Jadhav and M. B. B. Gite, "Real time intrusion detection with fuzzy, genetic and apriori algorithm," International Journal of Advance Foundation and Research in Computer (IJAFRC), Vol. 1, Issue 11, pp. 34-40, 2014

[40] Willian Halfond, 'Testbed', [Online]. Available: http://www-bcf.usc.edu/~halfond/testbed.html. [Accessed: 16- JUNE- 2016]