Anomaly Detection In Web Applications Using Gene Expression Programming

Jaroslaw Skaruz\(^1\) and Franciszek Seredynski\(^2,3\)

\(^1\) Institute of Computer Science, University of Podlasie, Sienkiewicza 51, 08-110 Siedlce, Poland, jaroslaw.skaruz@ap.siedlce.pl
\(^2\) Institute of Computer Science, Polish Academy of Sciences, Ordona 21, 01-237 Warsaw, Poland, sered@ipipan.waw.pl
\(^3\) Polish-Japanese Institute of Information Technology, Koszykowa 86, 02-008 Warsaw, Poland

Abstract

A novel approach based on applying a modern metaheuristic called Gene Expression Programming (GEP) to detecting web application attacks is presented in the paper. This class of attacks relates to malicious activity of an intruder against applications, which use a database for storing data. The application uses SQL to retrieve data from the database and web server mechanisms to put them in a web browser. A poor implementation allows an attacker to modify SQL statements originally developed by a programmer, which leads to stealing or modifying data to which the attacker has not privileges. The intrusion detection problem is transformed into a classification problem, which the objective is to classify SQL queries between either normal or malicious queries. GEP is used to find a function applied to classification of SQL queries. Experimental results are presented on the basis of SQL queries of different length. The findings show that the efficiency of detecting SQL statements representing attacks depends on the length of SQL statements.

Keywords: SQL, security, web application, anomaly detection, GEP

1 Introduction

Nowadays a lot of business applications are deployed in companies to support them with their business activity. These applications are often built with three layer manner: presentation, logical and data. Examples of the data layer are files and the databases containing data while the form of the presentation layer can be a desktop window or a Web site presenting data derived from the data layer and providing application functions. The logical layer is responsible for establishing connection to the database, retrieving data and putting them at the presentation layer. To manage data in the database usually SQL statements are used. When a user executes an application function then SQL query is sent to the database and the result of its execution is shown to the user. Possible security violations exist due to poor implementation of the application. If data provided by the user are not validated then he can set malicious values of some parameters of SQL query,
which leads to the change of the form of the original SQL query. In that case
the attacker receives not authorized access to the database. While SQL is used to
manage data in the databases, its statements can be ones of proves of potential
attacks.

The security concern related to business applications aims at ensuring data
integrity and confidentiality. One security solution is an intrusion detection system
available on the market. It is the application based on attack signatures. When
malicious activity matches a signature then an attack is detected. The drawback
of this class of security countermeasure is that only those attacks can be detected,
which signatures exist. Unfortunately, every a few weeks or even days new security
holes are discovered, which allow the attacker to break into the application and
steal data (12). The objective of this work is to build an intelligent system based
on GEP, which detects currently known attacks and those, which can occur in the
future.

In the literature there are some approaches to intrusion detection in Web ap-
plications. In (8) the authors developed anomaly-based system that learns the
profiles of the normal database access performed by web-based applications using
a number of different models. A profile is a set of the models, to which parts of
SQL statement are fed to in order to train the set of the models or to generate an
anomaly score. During training phase the models are built based on training data
and anomaly score is calculated. For each model, the maximum of anomaly score
is stored and used to set an anomaly threshold. During detection phase, for each
SQL query anomaly score is calculated. If it exceeds the maximum of anomaly
score evaluated during training phase, the query is considered to be anomalous.
The number of attacks used in that work was small and the obtained results with
the final conclusion should be confirmed.

Besides that work, there are some other works on detecting attacks on a Web
server which constitutes a part of infrastructure for Web applications. In (5) a
detection system correlates the server-side programs referenced by clients queries
with the parameters contained in these queries. It is a similar approach to detec-
tion to the previous work. The system analyzes HTTP requests and builds data
model based on the attribute length of requests, attribute character distribution,
structural inference and attribute order. In a detection phase built model is used
for comparing requests of clients.

The paper is organized as follows. The next section discusses SQL attacks. In
section 3 we describe GEP. Section 4 shows training data used for experiments.
Next, section 5 contains experimental results. Last section summarizes results.

2 SQL attacks

SQL injection attack consists in such a manipulation of the application commu-
nicating with the database, that it allows the user to gain access or to allow it to
modify data for which the user has not privileges. To perform an attack in the
most cases Web forms are used to inject part of SQL query. Typing SQL keywords
and control signs the intruder is able to change the structure of SQL query de-
veloped by a Web designer. If variables used in SQL query are under control of a
Anomaly Detection In Web Applications Using GEP

user, he can modify SQL query which will cause change of its meaning. Consider an example of a poor quality code written in PHP presented below.

```php
$connection=mysql_connect();
mysql_select_db("test");
$user=$HTTP_GET_VARS["username"];
$pass=$HTTP_GET_VARS["password"]; 
$query="select * from users where 
    login='$user' and password='$pass';";
$result=mysql_query($query);
if(mysql_num_rows($result)==0)
    echo "authorization failed";
else
    echo "authorization successful";
```

The code is responsible for authorizing users. User data typed in a Web form are assigned to variables `user` and `pass` and then passed to the SQL statement. If retrieved data include one row it means that the user filled in the form login and password the same as stored in the database. Because data sent by a Web form are not validated, the user is free to inject any strings. For example, the intruder can type: `' or 1=1 -- - in the login field leaving the password field empty. The structure of the SQL query will be changed as presented below.

```php
$query="select * from users where 
    login ='' or 1=1 --' and password=''';
```

Two dashes comment the following text. Boolean expression `1=1` is always true and as a result the user will be logged with privileges of the first user stored in the table users.

### 3 Gene Expression Programming

#### 3.1 Overview

GEP is a modern metaheuristic originally developed by Ferreira (1). It incorporates ideas of natural evolution derived from genetic algorithm (GA) and evolution of computer programs, which comes from genetic programming (GP) (4). Since its origination GEP has been extensively studied and applied to many problems such as: time series prediction (6)(11), classification (9)(10) and linear regression (2). GEP evolves a population of computer programs subjected to genetic operators, which leads to population diversity by introducing a new genetic material. GEP incorporates both linear chromosomes of fixed length and expression trees (ET) of different sizes and shapes similar to those in GP. It means that in opposite to the GP genotype and phenotype are separated. All genetic operators are performed on linear chromosomes while ET is used to calculate fitness of an individual. There is a simple method used for translation from genotype to phenotype and inversely. The advantage of distinction between genotype and phenotype is that after any genetic change of a genome ET is always correct and solution space can be searched through in a more extent.
At the beginning chromosomes are generated randomly. Next, in each iteration of GEP, a linear chromosome is expressed in the form of ET and executed. The fitness value is calculated and termination condition is checked. To preserve the best solution in a current iteration, the best individual goes to the next iteration without modifications. Next, programs are selected to the temporary population, they are subjected to genetic operators with some probability. New individuals in temporary population constitute current population.

3.2 The Architecture of Individuals

The genes of GEP are made of a head and a tail. The head contains elements that represent functions and terminals while the tail can contain only terminals. The length of the head is chosen as a GEP parameter, whereas the length of the tail is calculated according to the eq. 1:

$$\text{tail} = h(n - 1) + 1,$$

where $h$ is the length of the head and $n$ is a number of arguments of the function with more arguments.

Consider an example of a gene presented in eq. 2:

$$+Qd/ + cabdbca.$$  (2)

Its encoded form is represented by ET and shown in figure 1. The length of the gene head presented in eq. 2 equals to 6 and the length of the tail equals to 7 according to eq. 1. The individual shown in figure 1 can be translated to the mathematical expression 3:

$$\sqrt{a + b} + d.$$  (3)

To construct ET from the linear gene, the analysis must start from the left to the right of the gene elements. The first element of the gene is a root of ET. Next, take such a number of the following elements of the gene that equals to the number of parameters of the function previously taken and put them below it. If a node
is a terminal then a branch is completed. If this algorithm of constructing ET is followed it can be seen that some of elements in the tail of the gene do not occur in ET. This is a great advantage of GEP as it is possible to build ET of different sizes and shapes. A genetic change of the gene causes lengthen and shorten of ET.

A chromosome can be built from a few genes. Then sub-ETs are linked by a function - parameter of GEP. For detailed explanation of all genetic operators see (1), (2).

3.3 Fitness function

In the problem of anomaly detection there are four notions, which allow to look inside performance of the algorithm. True positives (TP) relates to correctly detecting attacks while false positive (FP) means that normal SQL queries are considered as an attack. False negative (FN) concerns attacks as normal SQL queries and true negative (TN) relates to correctly classified normal SQL statements. It is obvious that the larger both TP and TN the better mechanism of classification.

To assess an individual, its fitness should be evaluated. In this work we use sensitivity and precision, which are the most widely used statistics used to describe a diagnostic test (7). The sensitivity measures proportion of correctly classified attacks and precision refers to the fraction of correctly classified attacks over the number of all SQL queries, which are classified as attacks. Sensitivity and precision are calculated according to eq. 4 and eq. 5:

\[
\text{sensitivity} = \frac{TP}{TP + FN}, \quad (4)
\]

\[
\text{precision} = \frac{TP}{TP + FP}. \quad (5)
\]

Eq. 6 relates to fitness calculation of an GEP individual:

\[
\text{fitness} = 2 \ast \frac{\text{precision} \ast \text{sensitivity}}{\text{precision} + \text{sensitivity}}. \quad (6)
\]

An individual representing the optimal solution of the problem has fitness equals to 1.0 and the worst chromosome has 0.0. GEP evolve the population of individuals to maximize their fitness value.

4 Training data

All experiments were conducted using synthetic data collected from a SQL statements generator. The generator takes randomly a keyword from selected subset of SQL keywords, data types and mathematical operators to build a valid SQL query. Since the generator was developed on the basis of the grammar of the SQL language, each generated SQL query is correct. We generated 3000000 SQL statements. Next, the identical statements were deleted. Finally, our data set contained thousands of free of attack SQL queries. The set of all SQL queries was divided into 20 subsets (instances of the problem), each containing SQL statements of different length, in the range from 10 to 29 tokens (see below). Data with
attacks were produced in the similar way to that without attacks. Using available knowledge about SQL attacks, we defined their characteristic parts. Next, these parts of SQL queries were inserted randomly to the generated query in such a way that it provides grammatical correctness of these new statements. Queries in each instance were divided into two parts: for training GEP and testing it. Each of the part contains 500 SQL statements.

Classification of SQL statements is performed on the basis of their structure. Each SQL query is divided on distinct parts, which we further call tokens. In this work, the following tokens are considered: keywords of SQL language, numbers and strings. We used the collection of SQL statements to define 36 distinct tokens.

Table 1 shows selected tokens, their indexes and the coding real values.

Each token has assigned the real number. The range of these numbers starts with 0.1 and ends with 0.9. The values assigned for the tokens are calculated according to eq. 7:

$$\text{coding value} = 0.1 + (0.9 - 0.1)/n \times k,$$

where $n$ is the number of all tokens and $k$ is the index of each token.

Below, there is an example of a SQL query:

$$\text{SELECT name FROM users}$$

To translate the SQL query the table 1 is searched through to meet a token. The first token of the SQL query shown in eq. 8 is SELECT and the corresponding coding value equals to 0.1222. This step is repeated until all tokens are translated. Finally a vector: 0.1222, 0.9, 0.1444, 0.9 is received as an encoded form of query represented by eq. 8. All elements of the vector are terminals used to generating individuals of GEP.

5 Experimental results

In this section we are going to use GEP to find a function that can be used to classify SQL statements. Each of twenty instances of the problem consists of two parts: training and testing data. Each time GEP was run first on training part...
of an instance of the problem and next found classification rule was tested on the second part of the instance. In this work we apply in the most cases the same values of parameters as in (3). As the search space is bigger than this in (3), the number of individuals was increased to 100.

In this classification problem very simple set of functions was chosen, which consists of arithmetic operators (+, -, *, /). The set of terminals depends on the length of SQL queries used for training. The number of terminals equals to the number of tokens, which constitute SQL query. As a selection operator roulette wheel was chosen. Figure 2 shows detection system performance during training and testing.

It is easily noticeable that the false alarms rate in both figures 2 a) and b) are very similar. Presented percentage values of false alarm rate are averaged over 10 runs of GEP. Such results of the experiment allow to say that the best evolved mathematical expression classifies SQL queries in the testing set with nearly the same efficiency as in the training set. One of the reasons this happens is that although SQL statements are placed randomly to both data sets, they features similar structure in both data sets, which were used in the classification task. From the figure 2 b) it can be seen that the false negative rate changes in small extent for SQL queries of different length. The averaged FN over SQL queries with various number of tokens equals to 6.85 and the standard deviation equals to 2.6. At the same time the averaged FP rate equals to 33.43 with the standard deviation equals to 14.54. SQL statements constituted from 10 to 15 tokens are classified with lower error than longer SQL queries. For these shorter queries, the averaged sum of false alarms for each length of SQL equals to 17.4%. Figure 3 shows fitness of the best individual for SQL queries consisting of 10, 20 and 29 tokens. The findings show a great ability of GEP to find a good solution. At the beginning of the algorithm better solutions are found quickly. Next, they are improved a few times. The best fitness is obtained for SQL queries made of 10 tokens. The longer SQL statements are classified with nearly the same extent, which is confirmed by the charts in 40 generation for queries with 20 and 29 tokens. The form of the best individual for the instance with SQL statements constituted from 10 tokens
Figure 3: Algorithm run for SQL queries made of 10, 20 and 29 tokens

Figure 4: The form of the classification rule evolution
is shown in figure 4. The numbers in the expression tree represent positions of tokens within SQL statements. ET transformed to the mathematical expression is presented in eq. 9:

\[ 1 + 7 + (9 \times 5 \times 10) + 7 + 3. \]  

(9)

The classification function for the instance with 10 tokens was discovered in 36 iteration of GEP. Figure 5 shows the number of various tokens used for constructing the classification rule during each iteration of GEP. The number of tokens was calculated among the best 20 individuals. In the first iteration of GEP the initial number of tokens contained in 20 individuals is nearly the same. During evolutionary process some tokens, which make the classification function more powerful, occur more often in the individuals. At the same time there are also some useless tokens, which does not allow classify SQL queries. Tokens at position 8, 6 and 4 within SQL queries are not needed for classification of SQL queries made of 10 tokens and their existence in the individuals decreases in the next iterations of GEP.

6 Conclusions

In the paper we have presented an application of modern evolutionary metaheuristic GEP to the problem of detecting intruders in Web applications. We have shown a typical SQL attack and a transformation of the problem of anomaly detection to the classification problem. Classification accuracy depicts a great efficiency for SQL queries constituted from 10 to 15 tokens. For longer statements the averaged FP and FN equals to about 23%. We have also presented dynamics of GEP, which reveals some important features. On the one hand a minimal change in genotype leads to a great change in phenotype. On the other hand search space is searched through in more extent. We believe that it is possible to keep the advantages of GEP and at the same time to make ETs less susceptible to great changes.
Acknowledgment

This work was supported by the Ministry of Science and Higher Education under grant no. N N519 319935.

References