Improved Algorithm for Intrusion Detection Using Genetic Algorithm and SNORT

Mit H. Dave¹, Dr. Samidha Dwivedi Sharma²

¹Student, Department of Information Technology, NRI Institute of Science & Technology, Bhopal, India
²Head Of Department, Department of Information Technology, NRI Institute of Science & Technology, Bhopal, India

Abstract—Intrusion Detection Systems (IDSs) detects the network attacks by scanning each data packets. There are various approaches were done for making better IDS, though traditional IDS approaches are insufficient in terms of automating detection of attacks as attackers evolved. This research is focused on Genetic Algorithm approach used in Intrusion Detection. This research shows that how the traditional SNORT and Genetic Algorithm combined together so that the number of rules of SNORT is decreased. Experimental results shows that using Genetic Algorithm and SNORT can decrease the amount of detection time, CPU Utilization and Memory Utilization.

Keywords—Genetic Algorithm, SNORT, Fitness Function,

I. INTRODUCTION

In this research Genetic Algorithm is used as evolution factor for traditional SNORT (Network Based Intrusion Detection System). Because of SNORT’s large rule set detecting attacks very time consuming. SNORT sniffs both incoming and outgoing data packets and checks it carefully with each and every rule of its rule set.

A number of soft computing based approaches have been proposed for detecting network intrusions [9]. For low cost of result and strengthening system various soft computing approaches are available which exploits tolerance of uncertainty, approximation and ambiguity [9]. There are some popular areas of soft computing such as Fuzzy Logic, Artificial Neural Networks and Genetic Algorithms. When soft computing is used for intrusion detection it often correlates with the rule-based approach. Though there are various approach is done for applying soft computing methods for Intrusion Detection they are less effective and less-utilized.

We showed Genetic Algorithm based technique for intrusion detection. Genetic Algorithm is selected because of some of its attributes, e.g., strong to noise, no gradient information is required to find a global optimal solution, self-learning, etc. [9].

Using Genetic Algorithms for intrusion detection has proven to be an effective approach. In this research, we implemented software based approach. The software is experimented using KDD Cup 1999 data sets on intrusions, which has become standard for testing intrusion detection systems. The experimental results show that our approach is effective, and it has the liveness to either generally detect network intrusions or precisely classify the types of misuse intrusions.

Rest of the paper is organized as follows. Section II gives an overview of Genetic Algorithm method used for intrusion detection. Section III gives detailed implementation method. Section IV reviews the experimental results. Section V concludes this paper.

II. OVERVIEW OF GENETIC ALGORITHM

Genetic Algorithms helps in creating evolved population by iteration of genes of biology and genes [9]. Where each problem is solved by fixed number of evolved genes. The number of possible values of each gene is called the cardinality of the gene [9]. Figure I illustrates the operation of a general genetic algorithm. The operation starts from an initial population of randomly generated individuals. Then the population is evolved for a number of generations finally, the best chromosome is selected as the final output once the high optimization is achieved.

All the possible ways towards solution of a problem is depended upon evolved genes. As per the rule of “Survival of the fittest” we have to calculate of each evolved genes. For this purpose Genetic Algorithm uses a function called “Fitness Function”. An evaluation functions used to calculate the “goodness” of each chromosome. During evaluation [3], it calculates the best solutions from the amount of solutions located.

Genetic Algorithm uses two operators, crossover and mutation [3]. They are responsible for any evolved population’s reproduction and mutation. Selection of evolved population is forced towards fittest chromosome [3].
III. IMPLEMENTATION OF SNORTGA SYSTEM

3.1 Pre-calculation phase

Algorithm: Initialize chromosomes for comparison
Input: Network audit data (for training)
Output: A set of chromosomes
1. Range = 0.125
2. For each training data
3. If it has neighboring chromosome within Range
4. Merge it with the nearest chromosome
5. Else
6. Create new chromosome with it
7. End if
8. End for

Above algorithm shows the major steps in pre-calculation phase in which a set of chromosome is created using training data. This chromosome set will be used in the detection phase for the purpose of comparison.

3.2 Detection phase

Algorithm: Predict data/intrusion type (using GA)
Input: Network audit data (for testing), Pre-calculated set of chromosomes
Output: Type of data.
1. Initialize the population
2. Crossover Rate = 0.15, Mutation Rate = 0.35
3. While number of generation is not reached
4. For each chromosome in the population
5. For each pre-calculated chromosome
6. Find fitness
7. End for
8. Assign optimal fitness as the fitness of that chromosome
9. End for
10. Remove some chromosomes with worse fitness
11. Apply crossover to the selected pair of chromosomes of the population
12. Apply mutation to each chromosome of the population
13. End while

Above algorithm shows major steps of detection phase in which a population is being created for a test data and going through some evaluation processes (selection, crossover, mutation) the type of the test data is predicted. The pre-calculated set of chromosome is used in this process to find the fitness of each chromosome of the population.

A. Detection Algorithm Overview

To implement the proposed algorithm and to evaluate the performance of SNORTGA, we have used the standard dataset of KDD Cup 1999[11] “Computer network intrusion detection” competition [12, 13].

B. Implementation Procedure:

In the pre-calculation phase, we have made 23 groups of chromosomes according to training data. There were 23 (22+1) groups for each of attack and normal types presented in training data. Number of chromosomes in each group is variable and depends on the number of data and relationship among data in that group. Total number of chromosomes in all groups were tried to keep in reasonable level to optimize time consumption in testing phase.

In the detection phase, for each test data, an initial population is made using the data and occurring mutation in different features. This population is compared with each chromosomes prepared in training phase.
Portion of population, which are more loosely related with all training data than others, are removed. Crossover and mutation occurs in rest of the population which becomes the population of new generation. The process runs until the generation size comes down to 1 (one). The group of the chromosome which is closest relative of only surviving chromosome of test data is returned as the predicted type.

Among the extracted features of the datasets, we have taken only the numerical features, both continuous and discrete, under consideration for the sake of the simplification of the implementation.

IV. EXPERIMENTAL RESULTS

The simulation results are depicted in the following tables. For most of the data files, the detection rate is almost similar in case of Snort and SNORTGA. But as far as CPU and RAM utilization is concerned, SNORTGA outperform Snort.

### TABLE I

<table>
<thead>
<tr>
<th>File Size (Mb)</th>
<th>SNORTGA</th>
<th>SNORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>2870</td>
<td>2868</td>
</tr>
<tr>
<td>200</td>
<td>4762</td>
<td>4757</td>
</tr>
<tr>
<td>500</td>
<td>9614</td>
<td>9600</td>
</tr>
<tr>
<td>1000</td>
<td>15898</td>
<td>15897</td>
</tr>
<tr>
<td>2000</td>
<td>30128</td>
<td>30102</td>
</tr>
</tbody>
</table>

As shown in Table I, the detection rate varies with file size and for both the IDS; the detection rate remains almost similar.

### TABLE II

<table>
<thead>
<tr>
<th>File Size (Mb)</th>
<th>SNORTGA</th>
<th>SNORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>19%</td>
<td>21%</td>
</tr>
<tr>
<td>200</td>
<td>22%</td>
<td>34%</td>
</tr>
<tr>
<td>500</td>
<td>23%</td>
<td>40%</td>
</tr>
<tr>
<td>1000</td>
<td>25%</td>
<td>44%</td>
</tr>
<tr>
<td>2000</td>
<td>25%</td>
<td>51%</td>
</tr>
</tbody>
</table>

Table II shows the CPU utilization which clearly states that SNORT consumes more processing power during execution then SNORTGA.

### TABLE III

<table>
<thead>
<tr>
<th>File Size (Mb)</th>
<th>SNORTGA</th>
<th>SNORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>12MB</td>
<td>21MB</td>
</tr>
<tr>
<td>200</td>
<td>14MB</td>
<td>35MB</td>
</tr>
<tr>
<td>500</td>
<td>17MB</td>
<td>56MB</td>
</tr>
<tr>
<td>1000</td>
<td>21MB</td>
<td>89MB</td>
</tr>
<tr>
<td>2000</td>
<td>29MB</td>
<td>121MB</td>
</tr>
</tbody>
</table>

Table III shows the RAM utilization for both the IDS. It is clearly seen from the table data that SNORTGA consumes less RAM than Snort. These will in-turn provides faster execution of the files then Snort. Hence, the proposed algorithm has outperformed the existing snort algorithm.

V. CONCLUSION

In this work, we have discussed the methodology of Genetic Algorithms with SNORT. By classifying all the rules of SNORT based on their functionality, is helpful to make them faster in detecting attacks. It also supports automatic detection of new emerging attacks. Use of Genetic Algorithm with SNORT will decrease the amount of time to detect intrusion, decrease CPU utilization and memory utilization. This work improves the efficiency of traditional SNORT with the help of Genetic algorithm. Also it is not compromising security risk as it reduce the SNORT rule set based on similarities.

REFERENCES


KDD-CUP-99 Task Description; http://kdd.ics.uci.edu/databases/kddcup99/task.html


Results of the KDD’99 Classifier Learning Contest; http://cseweb.ucsd.edu/~elkan/clresults.html