Improved Genetic Algorithm for Intrusion Detection System

Ms. Dimpi Patel[1] Mr. Indr Jeet Rajput[2]
[1]Student [2]Assistant Professor
[1,2]Department of Computer Science and Engineering
[1,2]Hasmukh Goswami college of Engineering, Ahmedabad, Gujarat, India

Abstract— The Internet has become a part of daily life and an essential tool today. Internet has been used as an important component of business models. Therefore, It is very important to maintain a high level security to ensure safe and trusted communication of information between various organizations. Intrusion detection is one of the important security constraints for maintaining the integrity of information. Various approaches have been applied in past that are less effective to curb the menace of intrusion. There are large amount of network traffic captured in terms of number of features and number of record, so it is very difficult to process all the network traffic before making any decision about normal or abnormal. So it is having longer training time and complexity. Thus the purpose is to provide an intrusion detection system (IDS), by modifying the genetic algorithm to network intrusion detection system. As we have applied attribute subset reduction on the basis of Information gain. So the training time and complexity reduced considerably. we embedded a soft computing approach in rule generation, so Generated rule can detect attack with more efficiency.

Key words: Intrusion Detection System (IDS), network Intrusion detection system (NNIDS), Genetic algorithm (GA), Detection rate (DR), False Positive (FP)

I. INTRODUCTION

In this approach intrusion detection can be considered as data analysis process. Due to large amount of network traffic captured in terms of number of features and number of record, it is very difficult to process all the network traffic before making any decision about normal or abnormal. So first extract most relevant and effective 15 features on the basis of information gain in order to reduce the training time and complexity. We then fuzzify the input feature by using triangular function and finally apply these extracted features to Genetic algorithm to train the model to generate rule for classifying the intrusion attack. To test the performance of this model we use KDD'99 cup data set. The data set has been provided by NSL dataset. It consists of 41 features, 38 of them are numeric and 3 are symbolic. We have to convert these symbolic features into numeric before applying to genetic algorithm. Our aim is to generate rules to differentiate between attack class and normal class type. Attacks are fall into four main category i.e. Denial of Service (DOS), User to Root attack (U2R), Remote to Local (R2L) and Probing.

The paper organized as follows: Section 2 presents background, Section 3 presents the proposed algorithm, Section 4 presents the experimental methodology, Section 5 presents performance results and analysis and Section 6 presents conclusion.

II. BACKGROUND

The process of a genetic algorithm generally starts by selecting a population of chromosomes randomly. These chromosomes are representations of the rules. Based on the properties of the rules, different positions of individual chromosome are encoded in binary bits. These individual positions are called as genes. During evolution they can be changed within range randomly. During evolution stage the set of chromosomes are called a population. An evaluation function is used to calculate the "Fitness" of each chromosome[1]. There are two operators, mutation and crossover, which are used during evaluation process to form the natural reproduction and mutation of species.

Figure 1[2] shows the structure of a simple genetic algorithm.

It starts by generation of initial population randomly, then this process evaluate and evolve through selection, crossover, and mutation. And Finally, once the optimization meet its target, the best individual is selected as the final result.

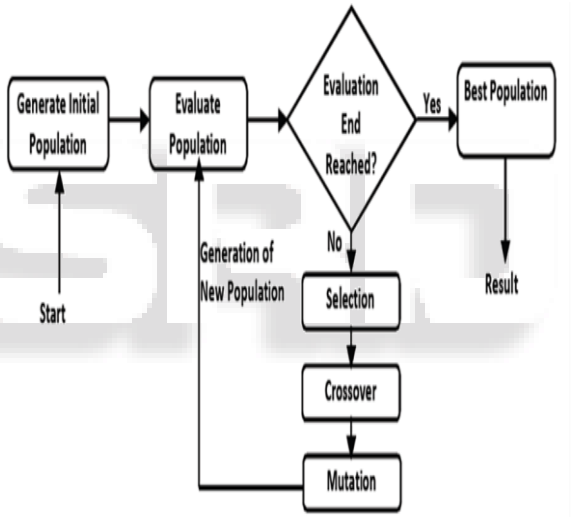

Fig. 1: Strucure & Processing in Genetic Algorithm

A. Major Steps in Genetic Algorithm

Algorithm: Rule set generation by using genetic algorithm.
Input: Network audit data, number of generations, and population size.
Output: A set of classification rules

(1) Initialize the population
(2) Check the fitness function
(3) Select only those rules that that meets the fitness criteria.
(4) Perform crossover for reproduction of new rule by exchanging some bits
(5) Perform mutation by flipping some bits
(6) Again go to line 2, until the specified numbers of rules are not generated.

III. PROPOSED ALGORITHM

Algorithm: Rule set generation using genetic algorithm.
Input: Network audit data, number of generations, and population size.

Output: A set of classification rules.
Begin
Step 1: Pre-process data by converting the symbolic feature into numeric data
Step 2: Select 15 features based on information gain
Step 3: For each extracted features
Step 4: Normalize and Fuzzify each selected attribute and divide into fuzzy classes
Step 5: Initialize the population
Step 6: *W1* = 0.2, *W2* = 0.8, *T* = 0.5
Step 7: *N* = total number of records in the training set
Step 8: For each chromosome in the population
Step 9: *A* = 0, *AB* = 0
Step 10: For each record in the training set
Step 11: If the record matches the chromosome
Step 12: AB=AB+1
Step 13: End if
Step 14: If the record matches only the "condition" part
Step 15: A=A+1
Step 16: End if
Step 17: End for
Step 18: Fitness = *W1* * *AB* / *N* + *W2* * *AB* / *A*
Step 19: If Fitness > *T*
Step 20: Select the chromosome into new population
Step 21: End if
Step 22: End for
Step 23: For each chromosome in the new population
Step 24: Apply crossover operator to the chromosome
Step 25: Apply mutation operator to the chromosome
Step 26: End for
Step 27: If number of generations is not reached, go to line 4
End

IV. IMPLEMENTATION METHODOLOGY

Implementation is being divided in two parts. First is generating rules for Intrusion detection using genetic algorithm and second is using rules for intrusion detection. Before introducing proposed implementation we will introduce some background of tools used for it.

A. Tools/Experiments for implementation

1) KDD NSL dataset:

NSL-KDD is a new version of KDD'99 data set to solve some of the inherent problems of the KDD'99 data set which are mentioned in [4]. New version of the KDD data set still suffers from some of the problems discussed by McHugh [5] and may not be a perfect representative of existing real networks, because of the lack of public data sets for network-based IDSs. Yet is can be used for research and experimental purposes. NSL-KDD is a data set recommended to resolve various integral problems of the KDD'99 data set. The amount of records in the NSL-KDD test and train sets are reasonable. This advantage forms it affordable to run the experiments on the complete set without demanding randomly selection of a small portion.

2) JGAP: Library for foundation class of Genetic Algorithm

The proposed method is implemented using the java language. This java language is built on third party software package JGAP. JGAP is a comprehensive GA/GP java toolkit. It provides basic genetic mechanisms that can be easily used to apply evolutionary principles to problem solutions. The package provides a rich set of GA foundation classes. JGAP is a Genetic Programming and Genetic Algorithms component which is provided as a Java framework. It specifies basic genetic mechanisms that can be easily used to apply evolutionary principles to problem solutions. It is also used to develop new machine learning schemes.

B. Implementation of GA for creating rules

This module is for generating GA rules. Basically GA rules are of if...then type of rules. They are being derived from the training data set. These rules are being optimised using genetic algorithm.

1) Preprocess Data Set:

We used KDD'99 cup data set. It consists of 41 features, 38 of them are numeric and 3 are symbolic. Convert these symbolic features into numeric before applying to genetic algorithm. Symbolic features can be converted into numeric according to following pseudo-code:

```
Create Encoding( AttributeName ,uniqueValueSet)
{
        set i=0;
        for each unique value in uniqueValueSet
                set
uniqueValueSet.intEncoding=i;
                i++;
        end for
}
```

2) Feature Extraction:

Due to large amount of network traffic captured in terms of number of features and number of record, it is very difficult to deal with all the network traffic before making any decision about normal or abnormal. So we used this pre-processed data as a input and generate output as only the top 15 features based on information gain. The figure below shows the top 15 attribute based on information gain.

Information Gain	Attribute Number	Attribute Name
1.0.806777083	5	src_bytes
2.0.672035304	3	Service
3.0.631947382	6	dst_bytes
4.0.515902949	30	diff_srv_rate
5.0.507754237	29	same_srv_rate
6.0.47284563	33	dst_host_srv_count
7.0.43902981	34	dst_host_same_srv_rate
8.0.412562311	35	dst_host_diff_srv_rate
9.0.403611513	38	dst_host_serror_rate
10.0.401731617	12	logged_in
11.0.396138161	39	dst_host_srv_serror_rate
12.0.390504636	25	serror_rate
13.0.382406912	23	Count
14.0.377279134	26	srv_serror_rate

Table 1: Top 15 features based on information gain

3) Normalization and Fuzzification of the Preprocessed Data Set:

We first reduce the data set based on selected 15 attribute. Then we normalize the reduced data set so that every value lies in the range of (0, 1). Beside we fuzzify it by using the triangular function as show below[3]:

$$\text{triangle}(x; a, b, c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right).$$

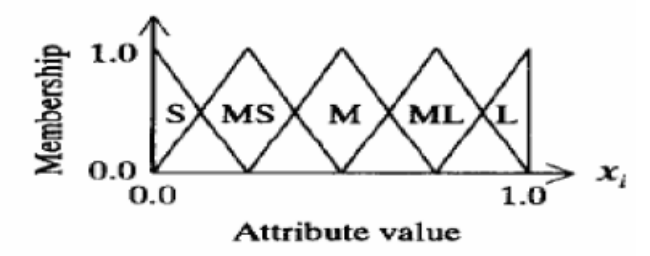

Fig. 2: Division of class

4) Selection

Selection Operator always ensures the best individual must be chosen. Selection of any rule can be takes place if its value is greater than the threshold value of the fitness. Fitness of the rule can be determined by line 18 of the Proposed algorithm. We select only those rules whose fitness value is greater than the threshold value.

5) Crossover

This operator of genetic algorithms changes the characteristics of two different solutions. The selection of two different pair is random and it continue until a completely new solution is carried out.

6) Mutation

This operator of genetic algorithms is used to change some of the random bits in a particular solution. It maintains the genetic diversity of the mutated algorithms.

V. Performance Results and Analysis

For Experiments we have used NSL dataset. NSL dataset is refined and derived from DARP Dataset. Details of it are in following table.

	Training Dataset	Testing Dataset
Normal	6668	9711
Anomaly	5888	12833
Total	12556	22544

Table 2: Distribution of Data

	Normal	Anomaly
Normal	9544	137
Anomaly	179	12654

Table 3: Confusion metrics

The following metrics are calculated to evaluate the performance of the methods:

- Detection Rate : Detection rate is calculated as the ratio between the number of correctly detected intrusions and the total number of intrusions.
  Detection Rate (DR) = TruePositive Rate / (FalseNegative Rate + TruePositive Rate)
- False Positive Rate: It is calculated as the number of normal patterns classified as attack divided by total number of normal patterns.
  False Positive Rate = FalsePositive / (FalsePositive + TrueNegative)

	Existing Algorithm	Improved Algorithm
Detection Rate	93.64	98.59%
FalsePositive Rate	6.13	0.014%

Table 4: Comparative Analysis

The experimental results clearly shows that the proposed method yielded good detection rates and low False Positive rate than existing method.

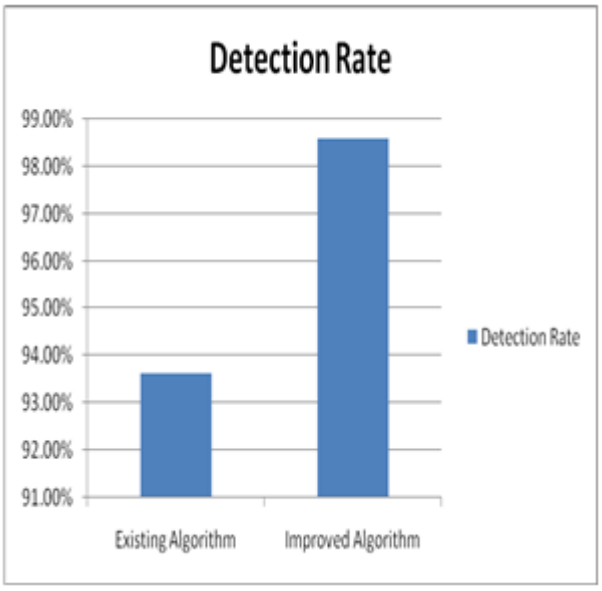

Fig. 3: Detection Rate Chart

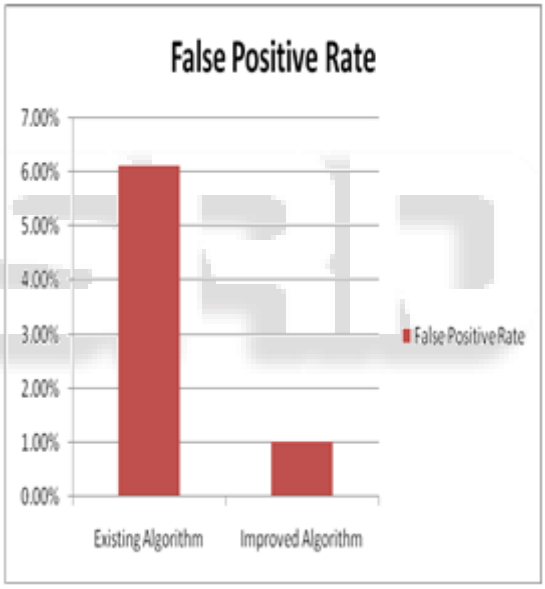

Fig. 4: False Positive Rate Chart

From the above charts it is clearly show that the detection rate of improved algorithm is high and it yields low false positive rate.

VI. Conclusion

Genetic algorithm is one of the best proven algorithms for NP-Hard problems. Here in this work we have used support confidence framework which need each chromosome to be evaluated for n number of times. As over here in this experimentation we are getting little improvement, but yet support-confidence as a fitness function remains very resource intensive. As we use only 15 features to describe the rules, its time of training is considerably reduced. Method of normalization of data works effectively for network audit data. Information gain as a feature selection works well in conjunction with GA.

REFERENCES

[1] http://en.wikipedia.org/wiki/Intrusion_detection_system
[2] W. Li, "Using Genetic Algorithm for Network Intrusion Detection". SANS Institute, USA, 2004.
[3] J. Zhao and B. Bose, "Evaluation of Membership Functions for Fuzzy Logic Controlled Induction Motor Drive", The university of Tennessee, Knoxville, TN-37996-2100, USA.
[4] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.
[5] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," ACM Transactions on Information and System Security, vol. 3, no. 4, pp. 262–294, 2000.

