2014 Sixth International Conference on Computational Intelligence and Communication Networks

Improved Genetic Algorithm for Intrusion Detection System

Dheeraj Pal
Department of Computer Science & Engineering
ASET, Amity University
Gwalior, India
Dheeraj.nit05@gmail.com

Amrita Parashar
Department of Computer Science & Engineering
ASET, Amity University
Gwalior, India
aparashar@gwa.amity.edu

Abstract- ***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. The purpose of this paper 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. Moreover, we embedded a soft computing approach in rule generation makes the rule more efficient than hard computing approach used in existing genetic algorithm. Generated rule can detect attack with more efficiency. This model was verified using KDD'99 data set. Empirical result clearly shows the higher detection rates and low false positive rates.***

.

Keywords— ***Intrusion Detection System (IDS), Neural network Intrusion detection system (NNIDS), Genetic algorithm (GA), Detection rate (DR), False Positive (FP).***

1. INTRODUCTION

In our 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 we first extract most relevant and effective features 15 features on the basis of information gain in order to reduce the training time and complexity. Besides, due to attribute subset reduction this system is feasible to apply in real time manner [3]. Performance of this approach is hinge upon the optimal subset of feature extraction. We then fuzzily the input [5] feature using 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 used KDD'99 cup data set. The data set has been provided by MIT Lincoln labs. 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. Our aim is to generate rules to differentiate between attack class and normal class type. The training data set contains 42674 connection records of the available 10% of the training set containing as published by Lincoln Labs which contains 494,021 connections for training of the attacks and normal type and the testing data set contains 32957 of total connection records of the attacks and normal 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.

2. BACKGROUND

The process of a genetic algorithm usually begins with a randomly selected population of chromosomes. These chromosomes are representations of the rules. According to the attributes of the rules, different positions of each chromosome are encoded in binary bits. These positions are sometimes referred to as genes and are changed randomly within a range during evolution. The set of chromosomes during a stage of evolution are called a population. An evaluation function is used to calculate the "Fitness" of each chromosome [1]. During evaluation, two basic operators, crossover and mutation, are used to simulate the natural reproduction and mutation of species. The selection of chromosomes for survival and combination is biased towards the fittest chromosomes.

A. Major Steps in Genetic Algorithm

Algorithm: Rule set generation 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

978-1-4799-6929-6/14 $31.00 © 2014 IEEE
DOI 10.1109/CICN.2014.178

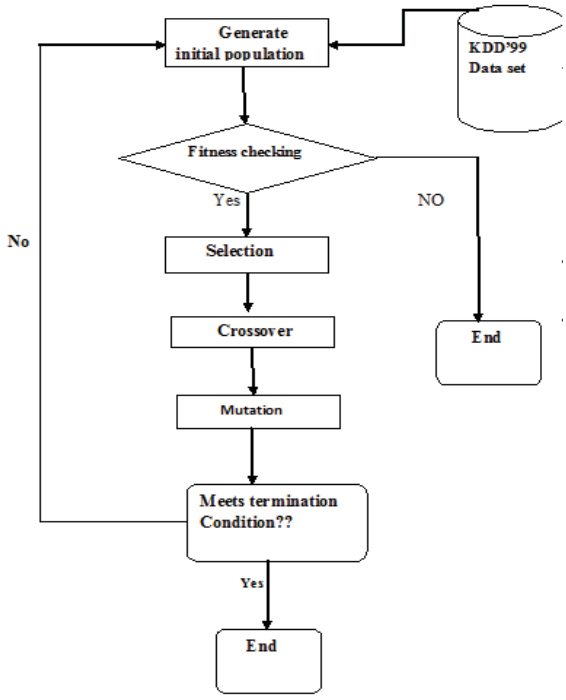

Figure1: Genetic algorithm

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

A. Experimental Setup & Evaluation Methodology

For our implementation, we have used the GALIB java library especially suited to develop Gas and java runtime environment jre1.7. It is also well-suited for developing new machine learning schemes. We decided to use GALIB since it is a java library, has been widely used by other researchers and well documented. We used a windows operating system with a Pentium 4 processor, 80GB of hard disk space and 512 MB of RAM to execute the computer program. We also used a tool known as RapidMiner, formerly YALE (Yet Another Learning Environment), is an environment for machine learning, data mining, text mining, predictive analytics, and business analytic. RapidMiner provides data mining and machine learning procedures including: data loading and transformation (ETL), data preprocessing and visualization, modeling, evaluation, and deployment. The data mining processes can be made up of arbitrarily nestable operators, described in XML files and created in Rapid Miner's graphical user interface (GUI). RapidMiner is written in the Java programming language.

1) Preprocess 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. We use the RapidMiner tool to convert the symbolic features into numeric one. Preprocess operator can be used to access the repositories introduced in RapidMiner 5.1 It should replace all file access, since it provides full file, it will provide the complete meta data of the data, so that all meta data transformations are possible.

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 process all the network traffic before making any decision about normal or abnormal. We used this pre-processed data as a input to the and generate output as only the top 15 features based on information gain using 3.2.3. The figure below shows the top 15 attribute based on information gain. See appendix for attribute number.

Attribute	Information Gain
Att 23	1
Att 34	0.885
Att 2	0.850
Att 38	0.795
Att 25	0.770
Att 39	0.762
Att 26	0.723
Att 1	0.711
Att 37	0.652

Att 10	0.622
Att 8	0.613
Att 22	0.585
Att 5	0.570
Att 22	0.511
Att 4	0.501

Figure 2: Top 15 Attribute Based On Information Gain.

3) Normalization and Fuzzification of the Preprocessed Data Set

We first reduced the data set based on selected 15 attribute in section 4.3.2. Now we normalize the reduced data set so that every value lies in the range of (0, 1). Besides, we fuzzify it by using the triangular function as shown below.

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

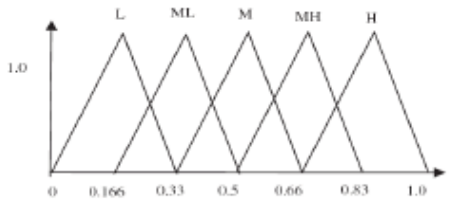

Figure 3: Division of Class

4) Initialize the Population

We now form the rule based on the attribute values lies in the range (0, 1). The rule formation takes place using the substring (L, LM, M, MH, and H). Initial rule formation is take place using Decision tree.

att36 > 0.005
| att23 > 0.001: ipsweep. {normal.=0, buffer_overflow.=0, loadmodule.=0, smurf.=0,
ipsweep.=72, multihop.=0, ftp_write.=0, nmap.=0,
neptune.=0, warezclient.=0, spy.=0,
satan.=0, rootkit.=0, teardrop.=0, pod.=0, portsweep.=0}
| att23 ≤ 0.001
| | att1 > 0.001: ftp_write. {normal.=0, buffer_overflow.=0, loadmodule.=0, smurf.=0,
ipsweep.=1, multihop.=0, ftp_write.=1, nmap.=0, neptune.=0,
warezclient.=0, spy.=0, satan.=0,
rootkit.=0, teardrop.=0, pod.=0, portsweep.=0}
| | att1 ≤ 0.001
| | | att1 > 0.000: pod. {normal.=0, buffer_overflow.=0, loadmodule.=0, smurf.=0,
ipsweep.=0, multihop.=0, ftp_write.=0, nmap.=0, neptune.=0,
warezclient.=0, spy.=0, satan.=0,
rootkit.=0, teardrop.=0, pod.=56, portsweep.=0}
| | | att1 ≤ 0.000: buffer_overflow. {normal.=0, buffer_overflow.=2, loadmodule.=0,
smurf.=0, ipsweep.=0, multihop.=0, ftp_write.=0, nmap.=0,
neptune.=0, warezclient.=0, spy.=0,
satan.=0, rootkit.=0, teardrop.=0, pod.=0, portsweep.=0}
att36 ≤ 0.005
| att34 > 0.000
| | att35 > 0.000
| | | att35 > 0.000
| | | | att26 > 0.000: portsweep. {normal.=0, buffer_overflow.=0, loadmodule.=0, smurf.=0,
ipsweep.=0, multihop.=0, ftp_write.=0, nmap.=0, neptune.=0,
warezclient.=0, spy.=0, satan.=0,
rootkit.=0, teardrop.=0, pod.=0, portsweep.=43}
| | | | att26 ≤ 0.000: pod. {normal.=0, buffer_overflow.=0, loadmodule.=0, smurf.=0,
ipsweep.=0, multihop.=0, ftp_write.=0, nmap.=0, neptune.=0,
warezclient.=0, spy.=0, satan.=0,
rootkit.=0, teardrop.=0, pod.=2, portsweep.=0}

5) Slection

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 mentioned in section 3. We select only those rules whose fitness value is greater than the threshold value i.e. Fitness >0.5.

6) Crossover

Crossover Operator randomly chooses a pairs individuals among those previously elected to breed and exchange substrings (L, LM, M, MH, and H) between them. The exchange occurs around randomly selected crossing points.

7) Mutation

Some of the substrings of rule generated is flipped randomly. After the 500th generation, the best chromosome was selected.

4. Performance Resut and Analysis

1) Detection Rate (DR): Detection rate is computed as the ratio between the number of correctly detected intrusions and the total number of intrusions.

$$DR = \frac{\text{TruePositive Rate}}{\text{FalseNegative Rate} + \text{TruePositive Rate}}$$

2) False negative: It is defined as the number of intrusion that is not detected as intrusion but in real it is an intrusion.

$$DR = \frac{\text{TruePositive Rate}}{\text{FalseNegative Rate} + \text{TruePositive Rate}}$$

4.1 Distribution of Record Type

Record Type	Training Dataset	Testing Dataset
Normal	7500	6000
Attack	1000	869
Total	8500	6869

Table1: Distribution of record set

4.2 Results

A) Performance Measure on Different Algorithm

Type	Proposed Algorithm		Existing Algorithm	
	Detection Rate	False Positive	Detection Rate	False Positive
Normal	96.86	3.1	93.64	6.13
Attack	97.46	2.5	91.88	6.72

Table 2: Comparative performance measure

B) Performance Measure on Different Dataset

Experiment 1:- When Threshold value T=0.5 is taken

Type	Training Dataset		Testing Dataset	
	Detection Rate	False Positive	Detection Rate	False Positive
Normal	96.86	3.1	91.78	9.21
Attack	97.46	2.5	91.88	18.06

Table 3: Result on different data sets when T=0.5

The experimental results show that the proposed method yielded good detection rates when using the generated rules to classify the training data itself. That is what we expected. When the resulting rules were used to classify the testing dataset, the detection rates of network attacks were decreased by great extent. The results have indicated that the generated rules were biased to the training data.

Experiment 2:- When Threshold value T= 0.6 is taken

Type	Training Dataset		Testing Dataset	
	Detection Rate	False Positive	Detection Rate	False Positive
Normal	95.3	4.7	90.58	9.21
Attack	95.05	4.9	81.1	18.06

Table 3: Result on different data sets when T=0.6

In the second experiment it is clear that the detection rate is little lower than the accuracy obtained in experiment 1.The detection rates could be higher if the fitness function and the GA parameters were chosen more appropriately. This is often done by trial and error.

C) Accuracy Graph

This graph represents the variation of accuracy with respect to attribute. It is clear from the graph that initially, accuracy increases gradually with number of attribute but after attaining a certain level the increase in accuracy is very low, on the flip side number of attribute rise sharply. This very little improvement in accuracy contributes greatly to complexity as number of attribute rises.

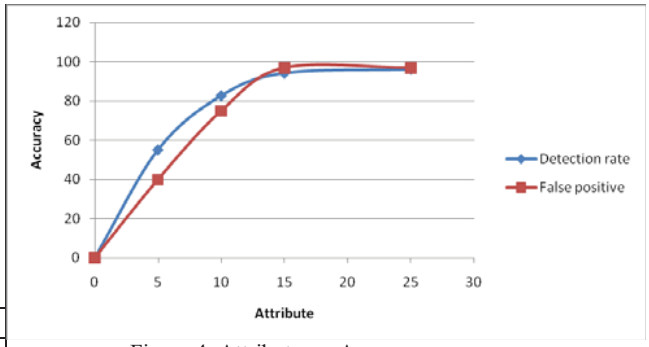

Figure 4: Attribute vs. Accuracy

5. CONCLUSION

Fuzzy-genetic Intrusion detection system combined with feature selection, it enable the system to produce optimal subset of attribute in the midst of huge network information. As we use only 15 features to describe the rules, its time of training is considerably reduced. Fuzzy logic generate rule that cover more vulnerability than rule formed by expert system using hard computing. Embedding a soft computing approach in rule generation makes the rule more efficient than hard computing. Further, genetic algorithm is used to tune the rule and generate the necessary rules. Genetic algorithm is an unsupervised learning contrary to neural network approach. It eradicates the problem of topology of network that is inherent to neural network. In neural network we assume any network topology and initial weight also then perform operation using back propagation algorithm and check the output by some threshold value. If output is below the threshold then we again change network topology or weight or both network topology and weight. But in this approaches there is no question of assuming any network topology and weight and gives globally optimal solution.

Besides, a simple but efficient and flexible fitness function, *i.e.* the support-confidence framework, is used to select the appropriate rules. The support-confidence framework is simple to implement and provides improved accuracy to final rules, it requires the whole training data to be loaded into memory before any computation. Depending on the selection of fitness function weight values, the generated rules can be used to generally detect network intrusions. The accuracy of this algorithm depends highly on fitness function and value of fitness function can be changed by simple trial and error approach. However, some limitations of the method are also observed. First, the generated rules were biased to the training dataset as we described in table3.

REFERENCES

[1] Wang Yunwu "Using Fuzzy Expert System Based on Genetic Algorithms for Intrusion Detection System" Information Technology and Applications, IFITA , International Forum Vol 2, 2009.

[2] Norbik Bashah Idris, Bharanidharan Shanmugam "Novel Attack Detection Using Fuzzy Logic and Data Mining", International Conference of Soft Computing and Pattern Recognition, SOCPAR '2009.

[3] Ajith Abraham, Ravi Jain, Johnson Thomas, Sang Yong Han, “Distributed soft computing intrusion detection system”, Journal of Network and Computer Applications,2007

[4] “Comparison of two feature selection methods in Intrusion Detection Systems,” Seventh International Proceedings of the 7th IEEE International Conference on Computer and Information Technology, pp. 83-86 , 2007.

[5] Murkami, Honda, “Compararive Study IDS method and Feed forward neural network”: International Joint Conference on neural Network, IJCNN,2005.

[6] Ren Hui Gong, Mohammad Zulkernine, Purang Abolmaesumi “A Software Implementation of a Genetic Algorithm Based Approach to Network Intrusion Detection” , Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks, 2005

[7] Suhail Owais, Václav Snášel, Pavel Krömer, Ajith Abraham “Survey: Using Genetic Algorithm Approach in Intrusion Detection Systems Techniques”, 7th Computer Information System and Industrial management Applications,2008.

[8] B.A. Fessi, S. BenAbdallah, M. Hamdi, N. Boudriga “ A New Genetic Algorithm Approach for Intrusion Response System in Computer Networks”,

[9] S. BenAbdallah, M. Hamdi, N. Boudriga “Genetic Algorithm for Intrusion Response System in Computer Networks”,