IJCSN International Journal of Computer Science and Network, Volume 5, Issue 3, June 2016
ISSN (Online) : 2277-5420 www.IJCSN.org
Impact Factor: 1.02

Rules Based Intrusion Detection System Using Genetic Algorithm

[1] Pawan S. Maniyar, [2] Dr. Vijaya Musande

[1,2] Dr.B.A.M.University, Aurangabad

Abstract - Now a days it is very important to maintain a high level security to ensure a safe and trusted communication of information between various organizations. But secured data communication over internet or any other network is always threats of intrusions and misuses. There are different Soft computing approaches have been proposed to detect the attacks. In this paper we proposed the genetic algorithm to generate the rules with the help of network audit data and for selection of rules used fitness function. The generated rules are used to detect or classify the attacks. By using Genetic Algorithm (GA) we can classify the different types of attack. To implement and measure the performance of system we used the DARPA benchmark dataset and obtained reasonable detection rate.

Keywords - *Computer & Network Security, DARPA 98 Dataset, Genetic Algorithm (GA), Intrusion Detection System (IDS).*

1. Introduction

Local networks and internet are growing at an exponential rate in recent years. While we enjoy the advantage that the new technology has brought us, computer systems are exposed to increasing security threats that originate from external or internal hosts. Although the different mechanism of protection, it is almost impossible to have a totally secure system. Therefore, intrusion detection technology becomes more and more important that monitors traffics and identifies network intrusion.

There are two major categories of the analyze techniques of IDS (Intrusion Detection System): the **anomaly detection** and the **misuse detection**. **Anomaly detection** uses the established normal profiles to identify any unacceptable deviation as the result of an attack. In a **misuse detection system**, also known as **signature based detection system** well-known attacks are represented by signature.
The Misuse approach uses several techniques grouped into three classes:

1) The rule-based approaches or expert systems,
2) Approaches based on signature
3) Genetic Algorithms GA.

In this paper, we present a GA approach for network intrusion detection. Genetic algorithms are used to optimize the search of attack scenarios in audit files. Our implemented system contains two modules where each operates at a different stage. In the training stage, a set of rules are generated from the audit data. In the stage of intrusion detection, produced rules are used to classify incoming network connections in real time. But the main goal is the optimization of the number of signatures in the audit file to minimize the search time and increase the detection rate of attacks. This system is tested using the Defense Advanced research Project Agency (DARPA) data set which has become the standard test systems for intrusion detection.

2. Related Work

Dheeraj pal [7] has proposed attribute subset selection with information gain and Genetic algorithm for creation of rules for detection. In this approach, GA is used to detect the attack with help of rules with less attribute, on Network Security Laboratory-Knowledge Discovery and Data Mining (NSLKDD) dataset.

Weiming Hu, Jun Gao[8] has proposed, two online Adaboost-based intrusion detection algorithms. In the first algorithm, a traditional online Adaboost process is used where decision stumps are used as weak classifiers. In the second algorithm, an improved online Adaboost process is proposed, and online Gaussian mixture models (GMMs) are used as weak classifiers. They further propose a distributed intrusion detection framework, in which a local parameterized detection model is constructed in each node

using the online Adaboost algorithm. A global detection model is constructed in each node by combining the local parametric models using a small number of samples in the node. This combination is achieved using an algorithm based on particle swarm optimization (PSO) and support vector machines. The global model in each node is used to detect intrusions.

Zhenwei Yu, Jeffrey Tsai[9] has proposed, automatically tune the detection model on-the-fly according to the feedback provided by the system operator when false predictions are encountered. The system is evaluated using the KDDCup'99 intrusion detection dataset. They has used anomaly detection technique.

Dong Song, Malcolm I. Heywood and A. Nur Zincir-Heywood[10] has proposed, The hierarchical RSS-DSS algorithm is introduced for dynamically filtering large datasets based on the concepts of training pattern age and difficulty, Such a scheme provides the basis for training genetic programming (GP) on a data set of half a million patterns in 15 min and used anomaly detection technique.

2.1 Genetic Algorithm

Genetic Algorithms is an optimization technique using an evolutionary process [4][5]. A solution of a problem is represented as a data structure known as chromosome. An evaluation function is used to calculate the goodness of each chromosome according to the desired solution; this function is known as "Fitness Function". GA process begins with series of initial solutions is initially generated (random population) and through a combination of algorithms similar to an evolutionary process (often a combination of elitism, crossover, and mutation) the process works towards evolving solutions having better "goodness" as evaluated by the fitness function.

In every generation the fitness of these chromosomes is checked. To determine the fitness of the chromosomes fitness function is used and then fittest chromosomes are selected. The chromosomes which have poor fitness value are discarded. The selected fit chromosomes undergo crossover, mutation to form a new population. This new population is used for the next generation. Normally, the algorithm terminates when either a set number of generations or a satisfactory fitness level has been achieved. Genetic algorithm is composed of three operators. They are reproduction or selection, crossover or recombination and mutation.

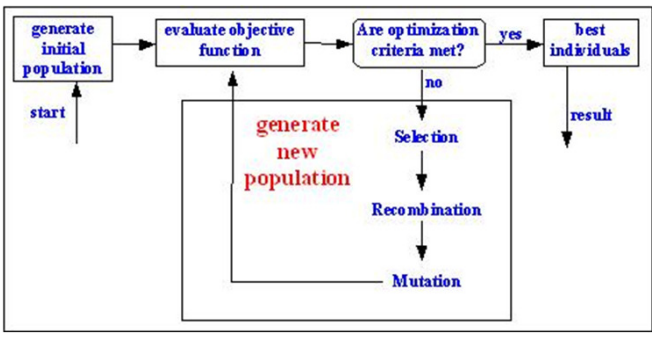

Fig. 1 Structure of simple Genetic algorithm

The basic concepts of Genetic Algorithms are simple, yet the process of choosing the gene representation, a good fitness function, and even application of the recombination [Whitley] can be the key to successful use of Genetic Algorithms.

2.2 DARPA Data Set

A key dependency of the work done by Gong and Li and as will be shown with netGA is the usage of DARPA data sets for training data. Creating this training data is not a trivial task and is considered beyond the scope of this project. The MIT Lincoln laboratory provides an excellent description of the process followed for creating the data. This DARPA training data is actually a result of test network traffic data, a Sun Microsystems Solaris and the use of Sun's Basic Security Module [Sun]. The data sets used in both papers were created in 1998[3]. Today's attacks have changed with regard to rule based systems, but the training data still works well for developing Genetic Algorithms.

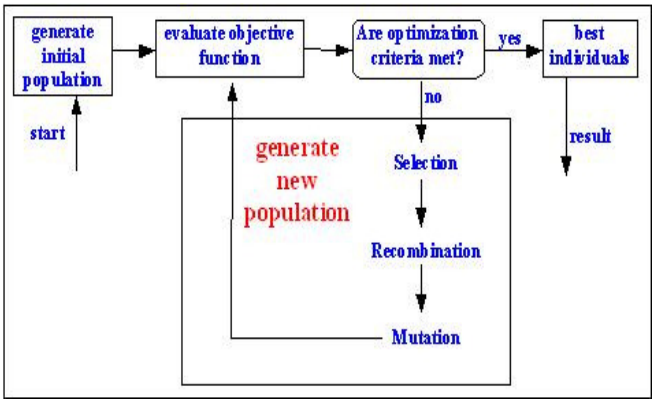

Fig. 2 Flow of Genetic Algorithm

3. Proposed System

3.1 Data Representation

The way that Genetic Algorithms are used with netGA is that rules are randomly created to match attacks encoded as a integer array with the seven elements shown in Figure 2. The first six attributes of the chromosome match the gene characteristics of an attack. The seventh attribute describes the attack type that the first six rules identify when they match. This representation uses the same approach as used by Gong[2].

Table 1. Chromosome Representation for Rule

Feature Name	Format	Number of genes
Duration	h:m:s	3
Protocol	Int	1
Source_port	Int	1
Destination_port	Int	1
Source_IP	a.b.c.d	4
Destination_IP	a.b.c.d	4
Attack_name	Int	1

In order to evaluate a rule represented by a chromosome, the DARPA audit data is parsed and loaded into a list of audit connections. The attributes loaded from the DARPA audit data directly match the attributes used in the chromosome representation. The gene representation follows the simple rule if A then B, where if the first six attributes are logically and-ed together are true (A), then the rule matches the attack (B**).** Following is the sample example [4] that classifies a network connection as the denial-of-service attack Neptune.

if (duration = "0:0:1" and protocol = "finger" and source_port = 18989 and destination_port = 79 and source_ip = "99.19.99.19" and destination_ip = "192.168.254.10") then (attack_name = "Neptune")

Above rule specifies that if a network packet is originated from IP address 99.19.99.19 and port number 18989 and send to IP address 192.168.254.10 at port number 79 using finger protocol for duration of connection 1 second then most likely it is Neptune attack which eventually make destination host out of service.

3.2 Fitness Function

Every chromosome is selected after applying fitness function to them. To determine the fitness of a rule, the support confidence framework [6] is used. If a rule is represented as if A then B [4] then the fitness of the rule is as follows:

$$support = |A \text{ and } B| / N$$

$$confidence = |A \text{ and } B| / |A|$$

$$fitness = w1 * support + w2 * confidence$$

Here, *N* is the total number of network connections in the audit data, |*A*| stands for the number of network connections matching the condition *A*, and |*A and B*| is the number of network connections that matches the rule *if A then B*. The weights *w1* and *w2* are used to control the balance between the two terms and have the default values of *w1*=0.2 and *w2*=0.8.

3.3 Crossover and Mutation

Crossover is one of the important steps in GA. There are three types of crossover techniques. They are one point, two point and uniform cross over technique. In this paper we used two point crossovers. Crossover involves splitting two chromosomes and then combining first part of a chromosome with the second part of the other chromosome.

Each gene in each chromosome is checked for possible mutation by generating a random number between zero and one and if this number is less than or equal to the given mutation probability then the gene value is changed. Mutations create diversity to search in domain regions that may otherwise be excluded.

Algorithm Steps

Listing 1 shows the major steps of the employed detection algorithm as well as the training process. It first generates the initial population, sets the defaults parameters, and loads the network audit data. Then the initial population is evolved for a number of generations.

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. $W1 = 0.2$, $W2 = 0.8$, $T = 0.5$

3. N = total number of records in the training set

4. For each chromosome in the population

5. $A = 0$, $AB = 0$

6. For each record in the training set

7. If the record matches the chromosome

8. $AB = AB + 1$

9. End if

10. If the record matches only the "condition" part

11. $A = A + 1$

12. End if

13. End for

14. Fitness = $W1 * AB / N + W2 * AB / A$

15. If Fitness > T

16. Select the chromosome into new population

17. End if

18. End for

19. For each chromosome in the new population

20. Apply crossover operator to the chromosome

21. Apply mutation operator to the chromosome

22. End for

23. If number of generations is not reached, then goto line 4

Listing 1. Major steps of the detection algorithm.

In each of the qualities rules are firstly calculated, then a number of best-fit rules are selected, and finally the GA operators are applied to the selected rules. The training process starts by randomly generating an initial population of rules (line 1). The weights and fitness threshold values are initialized in line 2. Line 3. Calculates the total number of records in the audit data. Lines 4-18 calculate the fitness of each rule and select the best-fit rules into new population. Lines 19-22 apply the crossover and mutation operators to each rule in the new population. Finally, line 23 checks and decides whether to terminate the training process or to enter the next generation to continue the evolution process.

4. Experimental Results

The genetic algorithm for rule generation is implemented using Java language (JDK1.8) in NetBeans8.0. The front end development environment used is NetBeans8.0. Two subsets were developed from DARPA 1998 data.

Table 1 gives the distributions of record types in both training and testing data set. The first row gives the number of normal network records. The second row gives the distributions of Smurf attack whereas the third row gives the distribution of Neptune attack.

The implementation is done in two phases. In the first phase the classification rules are generated using genetic algorithm. Support confidence function as fitness function. The GA parameters used were w1 = 0.2, w2 = 0.8, 200 generations, population of 2000 rules, mutation rate of 0.001. In the second (testing / 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 last generation finished. The group of the chromosome which is closest relative of only surviving chromosome of test data is returned as the predicted type.

Table 2: Results

Record Type	Training	Testing	Match %
Normal	73	64	87%
Smurf	799	790	98%
Neptune	96	96	100%

4.1 Execution Time

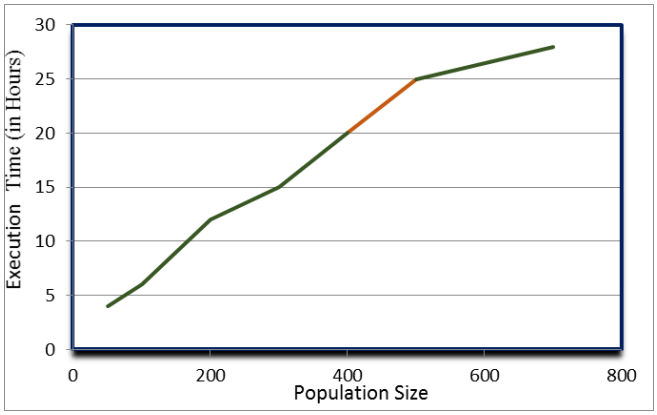

Fig 3. Effects of GA Population Size on Execution Time

The time taken by a genetic algorithm to reach to a required solution is an important aspect. This execution time increases linearly as the population size increases for the equal number of generations. Graph 4.1 shows the population size and corresponding execution time taken by the GA. The maximum number of generations is set to 200.

4.2 Population Size

Graph 4.2 shows the percentage detection for different number of generations of GA. As the number of generations is increased, the detection rate is improved at the cost of increased time required for the generation of rules. The best results are obtained after 200 generations.

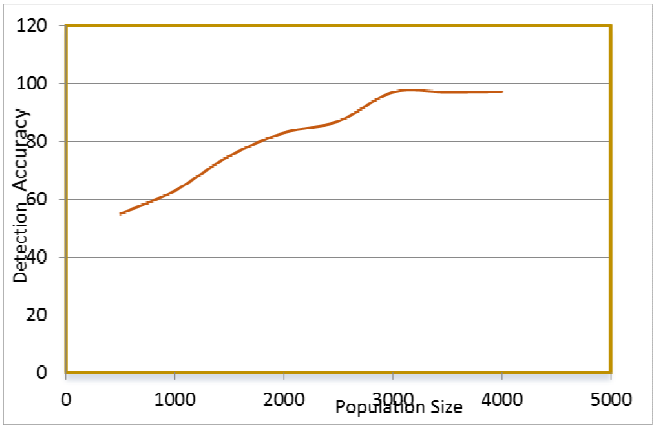

Fig 4. Effects of Population on Detection Accuracy

5. Conclusion

IDS is implemented using GA in two steps. In the first step, GA is used to generate classification rules where as in the second step these rules are used for intrusion detection. This reduces the search space and yields more accurate results while using smaller population and lesser number of generations compared to Gong et al.'s approach. This has reduced the time required for the generation of fittest rules. The given system is run for different generations. As the number of generations is increased, more accurate intrusion detection rates are obtained.

6. Future Scope

In future our plan is to apply single or multiple filters to enhance the system performance and to reduce time complexity of execution. Again we are planning to apply the proposed system output to the security system like Firewall machine to block the traffic whose IP address entries are made available to the pfirewall.log file and which is detected as vulnerable.

References

[1] W. Li, "A Genetic Algorithm Approach to Network Intrusion Detection", SANS Institute, USA, 2004.

[2] Li, Wei. 2002. "The integration of security sensors into the Intelligent Intrusion Detection System (IIDS) in a cluster environment." Master's Project Report. Department of Computer Science, Mississippi State University.

[3] MIT Lincoln Laboratory, DARPA datasets, MIT, USA, in November2004). *http://www.ll.mit.edu/IST/ideval/data/data_index.html*

[4] H. Pohlheim, "Genetic and Evolutionary Algorithms: Principles, Methods and Algorithms", http://www.geatbx.com/docu/index.html (accessed in January 2005).

[5] M. Crosbie and E. Spafford, "Applying Genetic Programming to Intrusion Detection", Proceedings of the AAAI Fall Symposium, 1995

[6] W. Lu and I. Traore, "Detecting New Forms of Network Intrusion Using Genetic Programming", *Computational Intelligence,* vol. 20, pp. 3, Blackwell Publishing, Malden, pp. 475-494, 2004.

[7] *Dheeraj Pal and Amrita Parashar "Improved Genetic Algorithm for Intrusion Detection System", 2014 Sixth International Conference on Computational Intelligence and Communcation Networks.*

[8] Weiming Hu, Jun Gao, Yanguo Wang, Ou Wu, and Stephen Maybank "Online Adaboost-Based Parameterized Methods for Dynamic Distributed Network Intrusion Detection", IEEE Trans. On Cybernetics.

[9] Zhenwei Yu, Jeffrey J. P. Tsai and Thomas Weigert, "An Automatically Tuning Intrusion Detection System", IEEE Trans. On Systems, Man and Cybernetics-Part B: Cybernetics Vol.37, No.2, April 2007.

[10] Dong Song, Malcolm I. Heywood and A. Nur Zincir-Heywood, "Training Genetic Programming on Half a Million Patterns: An Example From Anomaly Detection", IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 9, NO. 3, JUNE 2005.