2009 First International Conference on Computational Intelligence, Communication Systems and Networks

Efficient Snort Rule Generation using Evolutionary computing for Network Intrusion Detection

Raghavan Muthuregunathan[1], Siddharth S[2], Srivathsan R[3], Rajesh SR[4]
[1,2,3,4] *Undergraduate students, Madras Institute of Technology, Anna University, Chennai*
{[1]raghavan.mit, [2]mitsiddharth, [3]srivathsmit, [4]srrajesh1989}@gmail.com

Abstract

Network Intrusion Detection (NIDS) tool has become an important tool in detecting malicious activities in a network. Snort is a free and open source Network Intrusion Detection and prevention tool which is basically a rule driven system. Hence rule development for such NIDS tools becomes a sensitive task. Clustering techniques had been widely used to cluster the network traffic and to derive rule sets based on the resultant clusters. We propose a parallel Clustering technique followed by usage of evolutionary computing comprising of Genetic Algorithm and Hill climbing to optimize the clusters formed. Rules are generated by analyzing each individual clusters formed. The proposed system was specifically developed with a view to generate rule set for Snort based IDS efficiently. The results show that careful selection of fitness function could improve the efficiency of rule set generated. The computing power offered by Grid is used to accomplish the parallel computing task. Parallel Computation requires Cluster based resources which are offered by Grid.

Keywords*:* Network Intrusion Detection, Clustering, Genetic Algorithm, Hill Climbing, parallel Computing, Snort, Grid

1. Introduction

Now days, flooding attacks, port scanning have become more common that network intrusion detection system becomes an unavoidable entity of any network environment. NIDS tool monitors a network environment for attacks like DDOS, port scanning etc. NIDS accomplishes this task by analyzing each packet flowing in the network and trying to match with the rules from its Rule base. Snort is a free and open source network intrusion prevention and detection system. It utilizes a rule driven language which involves protocol matching, anomaly inspection and signature machine to match the flowing packets. It can take any action like generating an alert, dropping the packet and log data about the packet. Since the detection process of Snort heavily relies on the rule set present, it is very important to produce an efficient rule set that is able to detect most of the network based attacks with less overhead. It is also known that the maximum computational load on snort detection engine is totally based on the number of rules. Research works on network intrusion detection based on data mining techniques like Clustering and evolutionary are largely available in the literature. We have proposed a parallel version of a clustering method proposed by [4] to produce initial clusters and use a parallel evolutionary computing consisting of Genetic Algorithm and Hill Climbing to optimize the clusters formed initially. Then an analysis of the final clusters is done to generate a rule set. Since parallel processing is involved, it requires parallel cluster computing resource. We turn to the enormous computing power offered by Grid to meet this requirement. A sniffer in the target computing node collects information about the network traffic and stores in a traffic file. Based on size of traffic file, a grid scheduler schedules a Cluster Computing resource capable of running the parallel MPI code on the traffic file and generates the rule set for the snort installed target node. All parallel implementations of the algorithms to be mentioned are written using MPI libraries in C++.

The paper is organized as follows. Section 2 describes related work literature survey. Section 3 provides the proposed architecture. Section 4, 5 and 6 deals with the parallel algorithms proposed and Rule generation for Snort based IDS. Section 7 analyzes the experimental results. Section 8 provides future work and ends with conclusion

2. Related Work

Terrance.P. Fries [1] tried intrusion detection based on Genetic Clustering. In clustering the data set using Genetic Algorithm, he gave importance to inter cluster and intra cluster distance. He designed fitness functions accordingly. He used a fuzzy approach to generate the rules based on the final clusters formed. Similarly in [2] and [3], the authors used intra and inter cluster distance for optimal cluster distance. It shows that a good clustering algorithm should provide clusters which are loosely coupled with other clusters with more inter cluster distance and tightly coupled within a cluster with less intra cluster distance.

In [4], Hae sang park et al devised a hybrid clustering algorithm based on partitioning around medoids algorithm and nearest neighbor k means algorithm to cluster large data set. It could be parallelized because of independent nature of the algorithm and hence the algorithm speed can be considerably increased with the

978-0-7695-3743-6/09 $25.00 © 2009 IEEE
DOI 10.1109/CICSYN.2009.19

availability of parallel cluster computing resources offered by the Grid.

In [6], Fang-Yie Leu et al used the computation resources offered by Grid to solve the intrusion detection problem. Network traffic in a network is sampled every t seconds and a flow file is developed from the sampled network information. This flow file is sent to a detection node which is a computing resource available in grid. The detection node is allotted by a scheduler based on its own scheduling mechanism and requirements. The scheduling mechanism is not in our scope of research. Having scheduled a detection node, the flow file is sent to detection node which detects for attacks by analyzing various packet data present in the flow file. The detection nodes are installed with intrusion detection modules which are capable of detecting only specific attacks.

Most of the literature work involved utilized Genetic Algorithm for clustering process. However, a hybrid utilization of both genetic algorithm and Hill Climbing was not found. Also with the advent of Grid computing, availability of parallel computing resources is no longer a dream. Hence algorithms can be parallelized to decrease the time complexity and increase the efficiency of intrusion detection.

3. Proposed Architecture

The proposed architecture is shown in figure 1. The three main components of the architecture are Snort installed target node, Scheduler used to scheduler an available resource from grid and finally the parallel computing resource present in grid environment.

3.1. Sniffer

Sniffer is a component within the target node installed with Snort IDS. It samples the network for every t units of time. It stores the various packet data, Network connections data and network related information. It generates a traffic file based on sampling information. The size of the traffic file depends on the flow of packets in the network. A sample traffic file is shown in figure 2. It also requests the Scheduler to allocate a parallel processing enabled cluster resource based on the size of traffic file size. Later it transfers the traffic file and the MPI based parallel code as job to the allocated resource. It receives the solution rule set and updates it to Snort NIDS.

3.2. Scheduler:

The main task of scheduler is to discover a parallel Cluster resource in the Grid environment and allocate it to the target node that requested it. The scheduling algorithm used by the scheduler to allot is out of scope of the paper. It is assumed that, on receiving the traffic file, the scheduler allots the idlest and suited parallel cluster resource that is available in the Grid environment.

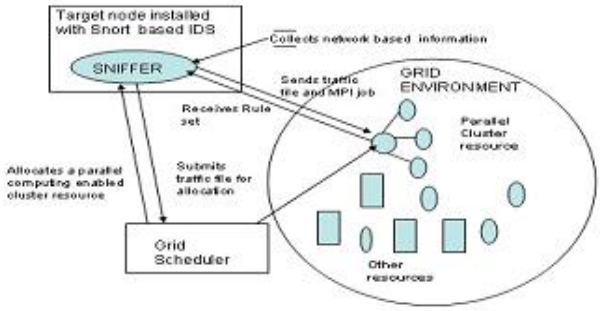

Figure 1. Overall Architecture of the System

3.3. Parallel Clustering Resource:

It receives the MPI job (code) and the corresponding traffic file. It operates on them, executing the parallel algorithms for clustering mentioned in later sections. It then finally generates the rule set file by analyzing the individual clusters formed. It is then sent back to the target node. The MPI code used in implementation requires at least one master node and one slave node in the cluster.

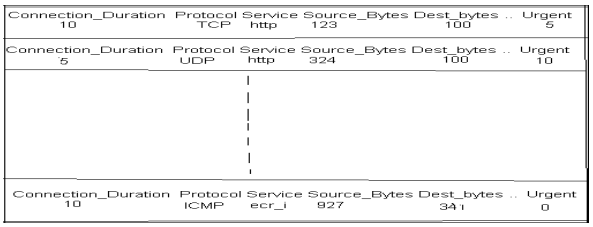

Figure 2. Sample Traffic file

4. Initial Cluster Development - Parallel Clustering Algorithm

The parallel Clustering resource allocated by the scheduler executes a parallel algorithm to cluster the traffic file received from the target node to form the initial set of clusters. The clustering algorithm was taken from [4] proposed by Hae sang park. We propose a parallelized version of this algorithm. It is a hybrid algorithm which is a combination of both k means clustering algorithm and Partitioning around medoids algorithm. The sequential algorithm has been mentioned below.

Suppose there are N records in the flow file, which has to be converted into a group of K clusters. Medoid is a record in the traffic file which is intended to best represent the cluster.

1. *Select initial Medoids:* Randomly select few records from traffic file as initial K medoids.
2. *Form clusters:* Assign each record from traffic file to the nearest medoid.

3. *Select new Medoids*: Calculate the sum of distance from all records to their medoids.
4. *Optimize each cluster:* For each cluster, select an object as a new medoid such that the sum of distances of all objects to their medoids is reduced.
5. *Form new clusters:* Assign each object from traffic file to the nearest medoid.
6. Repeat step 4 and 5 arbitrary number of times (or) if the new medoids obtained in step 4 are same as in the previous iteration.

Distance calculation between two objects or records in traffic file is described as follows. A simple Euclidian distance formula is used. In case of non numerical data, for example, service, protocol, flags set in a packet a constant value is taken as distance difference in that particular attribute direction.

A data parallel approach has been used to parallelize the algorithm. In step 2, each record is assigned to the nearest medoid by a distance calculation measure. The records in the traffic file are split up among the various slave nodes in the cluster. Each node does the process of Cluster assigning independently.

4.1. Parallel Algorithm – Master Node

- Read Traffic File and randomly select records as initial medoids and send them to all slave nodes
- Split the records evenly for each Slave node. For example, in case of 1000 records on 10 slaves, assign 100 records for each slave node
- For arbitrary number of times,
  - Slave nodes does Cluster forming process mentioned in step 2 of the sequential algorithm.
  - Master receives part solution from slaves and merges to form complete solutions.
  - The complete solution is sent to all slaves
  - Master assign one cluster to each of the slaves
  - Slaves do the process of finding new medoid that reduce the sum of all distances from non medoid objects to their respective medoid in the cluster
  - The newly found medoids are received by master from each slave.
  - Master then sends the newly found medoids to all slaves

4.2. Parallel Algorithm – Slave Node

- Receive traffic file from Master node and receive the initial set of medoids from master.
- Receive the split part of records on which the current slave node is going to operate upon , from the master node
- For arbitrary number of times,
  - Do the clustering process of assigning each record in the operational limit to the nearest medoid as per step 2 of sequential algorithm
  - Send the part solution to the the Master Node.
  - Receive the complete clusters formed at that iteration from Master node
  - Receive the cluster to be operated from the Master node.
  - For the obtained cluster, replace the current cluster medoid by a new item in the cluster that minimizes the sum of all distances from non medoid objects to the new item. The new item is selected as new medoid.
  - Send the new medoid to the allotted cluster to the master node.
  - Receive the new set of medoids (medoids that would have been generated by other slaves) from the root

5. Cluster Optimization using parallel Evolutionary computing

Execution of parallel algorithm mentioned previously would have generated clusters. However accuracy of clusters would not be high since the iterations were performed arbitrary number of times in the parallel algorithm mentioned previously. Hence a parallel evolutionary computing based approach is used to generate optimal clusters. We have used a hybrid approach using both Genetic Algorithm and Hill Climbing approaches.

5.1. Genetic Algorithm

It was first proposed by Holland [7]. It is based on the Darwinian evolution principle. A standard flow of Genetic Algorithm is shown in the figure 3. We have used Genetic Algorithm to optimize clusters and increasing the efficiency of the solution.

5.2. Hill Climbing

Hill Climbing can be considered as Genetic Algorithm without the Crossover operator. A standard flow of Hill Climbing is shown in figure 4.

5.3. Parallel Evolutionary computing Solution Representation

Each individual solution represents to which Cluster each and every record or object of the traffic file belongs to. Suppose there are N=50 records and K=3 clusters, a solution 1212332113… -23-43-44 represent records 1 belongs to cluster 1, record 2 belongs to cluster 2 and record 3 belongs to cluster 1 and so on. The numbers 23,43 and 44 represent the records in traffic file that are the cluster medoids of initial clusters that are formed.

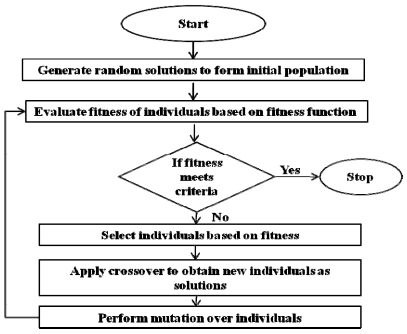

Figure 3. Schematic Diagram of Genetic Algorithm

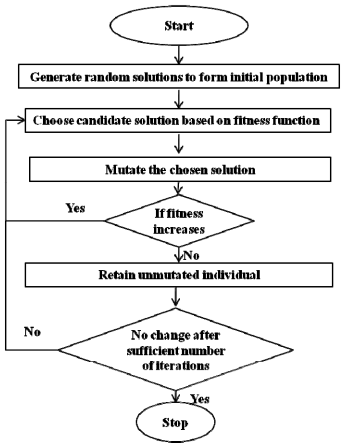

Figure 4. Schematic Diagram of Hill Climbing

5.4. Fitness Function

The three factors have been taken into account for fitness function design. They are intra cluster distance (Average distance between all non medoids of a cluster to the cluster medoid), inter cluster Distance (Average distance between various cluster medoids) and the density of Clusters (Number of Cluster points per unit area in the Cluster). Density of Cluster is derived by assuming that Cluster is circular.

Radius of Cluster R = distance between medoid and a non medoid item with maximum distance.

Density of Cluster = Total number of records in the cluster) / (∏ * R * R)

Fitness $F(x) = W_1$ * InterCluster Distance + W_2 * IntraCluster Distance + W3 * Cluster Density

W_1, W_2, W_3 are weights that can be fixed constant depending on the implementer's wish.

5.5. Parallel Algorithm – Master Node

- Generate candidate solutions using parallel clustering algorithm mentioned in previous section.
- Each solution consists of a string representing to which cluster each item belongs to as shown above and the medoids associated with it.
- For each generation,
  - Evaluate Fitness for each solution in the population and sort them in descending order based on fitness
  - Send the first half of the population to slaves performing Genetic Algorithm based breeding
  - Send the second half of the population to the slaves performing Steepest ascent Hill Climbing
  - Receive the new generation of population from the Slaves

The above for loop can be executed arbitrary number of times or it can be terminated if the fitness of the population does not increase substantially.

5.6. Parallel Algorithm – Slave node (GA based slave)

- Receive the sub set of population from the Master Node. The received population is generally fitter because first half of the population is sent by the master node.
- Select individuals based on fitness value
- Apply GA operations of crossover, mutation
- Form the new population and send it back to the Master

5.7. Parallel Algorithm – Slave node (Hill Climbing based slave)

- Receive the sub set of population from the Master Node. The received population is generally weaker because second of the population is sent by the master node.
- Select individuals based on fitness value
- Apply Hill Climbing operation mentioned in figure 4. It generally involves only mutation. Crossover is not done in order to prevent characters of weak individuals passing on to next generation
- Form the new population and send it back to the Master

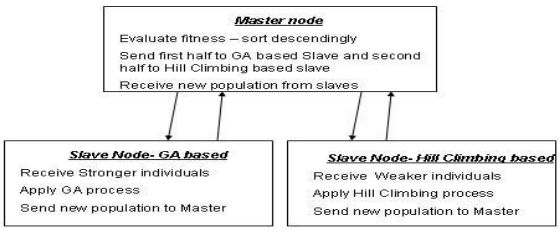

Figure 5. Parallel Evolutionary computing to optimize clusters

Selection: A roulette wheel fitness based selection is done. A highly fit individual have more probability of getting selected for crossover.

Crossover: A multipoint Crossover has been implemented with crossover probability P_c. Its value is kept constant at 0.7 throughout the experiment.

Let's say two individuals selected for crossover are,

1212**3321**13… -23-43-44 and 123**2133**123… -23-43-44

The generated offspring would be,

1212**2133**13… -23-43-44 and 123**3321**123… -23-43-44

Mutation: Mutation is done with a probability P_m which is kept constant throughout the experiment at 0.04. A sample mutation is shown in the figure.

Before Mutation	12**12**332113… -23-43-44
After Mutation	12**31**12332113… -23-43-44

5.8. Reason for going to a hybrid approach

Genetic Algorithm is good at carrying parent's behavior to the next generation. It is felt that less fit individuals should not breeded and their attributes should not pass on to the next generation. Genetic Algorithm also suffers from local maxima problem if the fitness function is weak. It is important to maintain the diversity of solutions. Hence first half stronger solutions are processed with GA while second half weaker solutions are processed with Hill Climbing which mainly involves mutation only. If a better solution is formed by Hill Climbing Slave with higher fitness, it would be passed on GA based Slave in the next iteration or generation for breeding. It is because the master node evaluates fitness and sorts the population in descending order.

6. Rule Generation

Snort based rules are generated by analyzing the final set of clusters formed after performing evolutionary computing process. All records in each cluster are analyzed. For attributes of record which are continuous like number of bytes sent by source in a TCP connection, a range is taken based on records in cluster with minimal and maximal value of the particular attribute in the cluster. For attributes of record which are discrete like service involved, all discrete values of the attribute in the cluster are taken into account in rule generation. It is assumed that cluster with maximum number of records is assumed to represent normal packet traffic. Other clusters are labeled as attacks.

Suppose a cluster K which is labeled as attack, it contains records in which the value of source bytes attribute to range from 20 to 200 and Flag attribute have discrete values like SF, S , ACK etc., the rule would be written as: *If (source bytes >20 and source bytes < 200 and flags set= SF or S or ACK) then declare its attack.*

6.1. General format of Snort rules

action protocol src_ip src_port direction dst_ip dst_port [*options*]

Snort can take one of three actions
Log – Log information about a packet
Alert – record an alert about a packet
Pass – drop a packet
A sample snort rule testing the above condition would be,

*PASS tcp any any <> any any (flags:SF * S * A; dsize:20<>200; msg:"Possible attack")*

First field describes action to be taken, second field indicates the protocol, any indicates any IP address or port number. Option flag is used to check the flags set in tcp packet and option dsize is used to check the size of payload in bytes. Option msg shows the message to be logged and the packet is dropped because action mentioned here is PASS.

7. Experimental Results

All parallel algorithms described were written using MPI libraries in C++. A Cluster resource with *one* master node and *four* slave nodes were used for testing purpose. KDD cup 99 dataset for intrusion detection was used to generate packets to test the target node installed with Snort. The features of network information captured by sniffer were also according to KDD cup dataset 99. To mention a few, Duration of connection, protocol, service, number of source bytes etc.[9].

Experiment was conducted to test the efficiency of rule generated for the Snort NIDS. Ten percent - KDD Cup 99 dataset was chosen. The dataset was split into ten sets. Each set consisted of about 500 thousand entries. Packet construction tools like jpcap APIs, Hping and ANTs were used to generate packets according to the KDD dataset. The detection rate (TRUE positive) and the False positive was shown in the graph figure 6 and table 1 as operating characteristic curve for each part of KDD data set that was split. Each point represents the detection rate (TRUE positive) and wrongly detected (FALSE positive) of the dataset which contained about 500000 entries. The average false positive rate was 0.43% and true positive rate was 64.02%. Figure 7 compares the results of accuracy of detection by the rules produced in the work of [5] and the rules produced by our work.

Data Set	False positive rate in %	True positive rate in %
1	0.1	56.4
2	0.6	58.2

3	0.6	67.3
4	0.4	70.5
5	0.8	73.4
6	0.3	62.6
7	0.2	64.7
8	0.5	61.2
9	0.3	66.6
10	0.5	59.3

Table 1: False positive rate and true positive rate results for each data set.

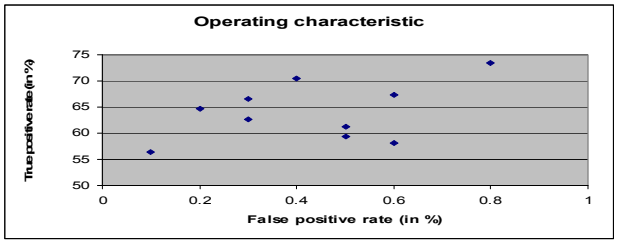

Figure 6: Operating Characteristic curve

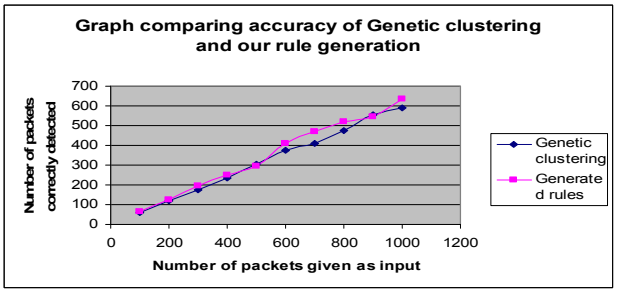

Figure 7: Comparison Graph

8. Future Work and Conclusion

Our work showed that hybrid evolutionary computing using both genetic algorithms and hill climbing could produce new results. It is important that utilization of parallel computing should be encouraged since enormous computing power is provided by Grid. In future, it is important to work on utilizing other evolutionary computing techniques like Ant Colony Optimization and Swarm Intelligence replacing Genetic Algorithms. Since the fitness function contained many objectives like less intra cluster distance and more inter cluster distance etc, a multi objective pareto GA or Nash GA can be utilized for the same problem.

9. References

[1] T.P. Fries, "A fuzzy genetic approach for intrusion detection", Proceedings of the GECCO conference companion on Genetic and evolutionary computation, pp. 2141-2146, 2008.

[2] Hwei-Jen Lin, Fu-Wen Yang and Yang-Ta Kao, "An efficient Genetic Algorithm based Clustering", Tamkang journal of science and engineering, volume 8, pp. 113-122, 2005.

[3] U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," Journal of Pattern Recognition, vol. 33, pp. 1455–1465, 2000.

[4] Park, Hae-sang, Lee, Jong-seok, Jun, Chi-hyuck. "A K-means-like Algorithm for K-medoids Clustering and Its Performance.", Proceedings of the 36th CIE Conference on Computers & Industrial Engineering, pp.1222-1231, 2006

[5] Liu, Y., Chen, K., Liao, X., and Zhang, W. "A Genetic Clustering Method for Intrusion Detection." Journal of Pattern Recognition, vol. 37 (5), pp. 927-942, May 2004.

[6] Fang-Yie Leu, Jia-Chun Lin, Ming-Chang Li, Chao-Tung Yang, Po-Chi Shih, "Integrating Grid with Intrusion Detection", Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05), vol. 1, pp. 304-309, 2005

[7] J. H. Holland, "Adaptation in Natural and Artificial Systems". The University of Michigan Press, Ann Arbor, MI, 1975.

[8] Krishna K, Narasimha murthy M, "Genetic K means algorithm", IEEE transactions on Systems, Man and Cybernetics, volume 29, pp. 433-439, 1999.

[9] KDD-Cup 1999, Computer network intrusion detection cup 1999 data set, http://www.sigkdd.org/kddcup/index.php,1999, Last visited 05/05/2009

[10] Snort , www.snort.org last visited 05/05/2009