Intrusion Detection in Cloud Systems Using Alternative Fuzzy C-Means Clustering and Artificial Neural Networks

CSC 724 - Advanced Distributed Systems, Final Report

Anjali Doneria
adoneri@ncsu.edu

Dipanjan Nag
dnag@ncsu.edu

Varun Garg
vsgarg@ncsu.edu

ABSTRACT

Although anomaly-based Intrusion Detection Systems(IDS) are able to identify previously unknown types of attacks, they suffer from low precision and recall, especially for unknown types of attacks[15]. The aim of our project is to find a way to improve these parameters. To that effect, we propose using Alternative Fuzzy C-Means Clustering (AFCM) to cluster the training data into homogeneous training subsets, then use those subsets to train multiple Artificial Neural Networks (ANN) and finally use a aggregating module to aggregate the results. We have used KDD Cup '99 dataset for training our neural networks and used live internet traffic to test our system.

Keywords: Fuzzy Clustering, Intrusion Detection, Alternative Fuzzy c-means Clustering, Artificial Neural Network, Outliers, Robust

1. INTRODUCTION

Nowadays, many companies rely on the convenient and on-demand access to the shared pool of configurable computing resources which is offered through cloud computing. It can be provided as Software as a service (*SaaS*), Platform as a service (*PaaS*) and Infrastructure as a service (*IaaS*) platforms. Security and privacy in this given scenario is the primary concern of the cloud users and hence the highly researched topic in this area. As surveyed by researchers, the major security concern after data security is the intrusion detection and prevention in cloud infrastructures.

The common intrusions that are prevalent in cloud infrastructures are insider attack, flooding attack, user to root attacks, port scanning, attacks on virtual machine or hypervisor, and backdoor channel attack [6]. The commonly used solution to some of these intrusion attacks is the use of firewalls that manage the front access points of the network. But firewalls have the limitation that they cannot detect insider attacks in the network as they tend to operate on the boundary of the network. Therefore, intrusion detection systems are used to prevent such attacks. There exist two kind of intrusion detection systems broadly, based on their location within the system, network based and host-based. Network based intrusion detection systems are placed at a common place of the network and they monitor entire network activity for prevention while host-based systems are placed at each node to monitor their activity for intrusion.

Various approaches have been suggested by researchers to detect attacks on the network. Some of the popular ones are signature based detection which use predefined rule sets, and anomaly based detection that use different machine learning algorithms.

For applications like online retail sales, online auctions etc., network security becomes of primary concern[15]. So, intrusion detection systems detect the attacks by examining the data records observed in processes on the network. They pointed out that detection precision and detection stability are the two key factors that become important in assessment of these systems.

A lot of data mining techniques, like ANN and SVM, have been employed to build an efficient intrusion detection system. Although ANN has been successful in solving many complex scenarios for this problem, they suffer from few drawbacks [15]: (a) lower detection precision for low-frequent attacks like Remote to Local (R2L), User to Root (U2R) and, (b) weaker detection stability. The reason for this can be accredited to the small sample size availability for these low-frequent attacks. This, in turn, makes the learning for ANN difficult and therefore, detection precision drops. Capturing these low-frequent attacks becomes important because some serious consequences can be caused if these attacks succeed.

To solve the problems stated above, a novel approach for ANN based IDS, FC-ANN (Fuzzy c-means Clustering and Artificial Neural Networks) was proposed [15]. The paper also points out that in case of IDS the low-frequent attacks are often outliers. [16] mentions that Fuzzy c-means Clustering method is not robust to outliers and noises, so they proposed an improvement to the existing fuzzy clustering algorithm to make it more robust. So, we propose to use this method of Alternative Fuzzy c-means Clustering (AFCM) instead of Fuzzy c-means Clustering Algorithm for designing the intrusion detection engine. The general procedure of AFCM-ANN approach is inspired from [15] which broadly divides the whole algorithm into three stages: (I) Clustering - by using AFCM to generate homogeneous training datasets, (II) training different ANNs based on generated training sets from stage I and, (III) detecting the attacks based on a fuzzy aggregation module.

The rest of the paper is organized as follows. In §2, we elaborate upon the design and algorithm for AFCM-ANN approach by explaining its principles and working procedure. §3 discusses the KDD Cup Dataset that is used to train the AFCM-ANN model and steps involved in preparation of training and testing data. We talk about the experimental set-up and different test cases in detail with focus on the evaluations from these experiments. §4 briefly discusses

some of the existing research in the domain of intrusion detection systems. §5 lists down the limitation of the proposed system and discussion on the future work. In the end, §6 concludes the research paper.

2. DESIGN AND ALGORITHM

In this section, we will discuss the AFCM-ANN framework in more detail. It divides the training data into several subsets using the AFCM clustering method to obtain homogeneous clusters. These subsets are then used to train the ANNs. Then, the membership grades for all the data points are obtained for the generated subsets and it is combined to train a new ANN for the final results. In the training phase, the following steps are performed:

Stage I An arbitrary dataset is divided into training set TR and testing test TS. Then it is passed through the clustering module to divide into different subsets TR_1, TR_2,...TR_k

Stage II For each training subset generated TR_i (i = 1,2,3,..k), k different ANN are trained by a learning algorithm to obtain k different base ANNs

Stage III A new ANN is trained using the output of whole training set TR from Stage II ANNs and memberships of each point in TR to the clusters generated in Stage I. This step, also referred to as Fuzzy Aggregation step, ensures that the error for every ANN_i is reduced for better detection

In the testing phase, the dataset TS is directly fed into k different ANN obtained from Stage II above. Based on the outputs obtained from them and membership values for data points in TS, a new input is created for Stage III ANN and the final result is predicted for anomaly detection.

2.1 Stage I - Alternative Fuzzy c-mean Clustering

The primary purpose of this step to generate different homogeneous datasets from the heterogeneous training dataset based on fuzzy membership values. The data within the same cluster must have homogeneity and the data belonging to different clusters should have heterogeneity. So, in this phase, the whole training data TR undergoes alternative fuzzy c-means clustering method [16]. The fuzzy membership values for the data points is obtained using given distance function,

$$1 - exp(-\beta||x_j \ - \ z_i||^2) \tag{1}$$

where, β is a constant defined by,

$$\beta \ = \ (\frac{\sum_{j=1}^{n} ||x \ - \ \bar{x}||^2}{n})^{-1} with \ \bar{x} = \frac{\sum_{j=1}^{n} x_j}{n} \tag{2}$$

This function ensures the robustness of AFCM. The points lying farther from the cluster are assigned lower weights than the points lying closer to the cluster. The cluster groups are defined based on the optimization of fuzzy membership function,

$$J \ = \sum_{i=1}^{c}\sum_{j=1}^{n}(\mu_{ij})^m \ (1 - exp(-\beta||x_j \ - \ z_i||^2) \tag{3}$$

The process of clustering is an iterative process which is performed by calculating membership values, cluster centers at every step and optimization of given cost function,

I. Find membership of each data point using,

$$\mu_{ij} \ = \ \frac{[1/(1 \ - \ exp(-\beta \ (||x_j \ - \ z_i||)^2)]^{1/(m-1)}}{\sum_{k=1}^{c}[1/(1 \ - \ exp(-\beta \ (||x_j \ - \ z_i||)^2)]^{1/(m-1)}} \tag{4}$$

where m is the index of fuzziness > 1

II. Calculate cluster centers by,

$$z_i \ = \ \frac{\sum_{j=1}^{n} \ (\mu_{ij})^m \ exp(-\beta \ ||x_j \ - \ z_i||^2)x_j}{\sum_{j=1}^{n} \ (\mu_{ij})^m \ exp(-\beta \ ||x_j \ - \ z_i||^2)} \tag{5}$$

III.

$$max_{ij} \ \{\mu_{ij}^{(q+1)} \ - \ \mu_{ij}^{(q)}\} \ < \ \epsilon, \tag{6}$$

where q is the number of iterations. So, we STOP when (6) is satisfied, else we go back to Step 1 and iterate again for q = q + 1

2.2 Stage II - Artificial Neural Network

The ANN module is used to learn about the patterns present in the dataset and make decisions on how closely a given attack or normal interaction in the network is related to the given patterns. The module is a feed forward neural network, trained via back-propagation. The module has an input layer, a hidden layer and an output layer.
An input to the ANN would be multiplied by a weight and fed to the hidden layer. Every node of the hidden layer would be fed the following input :-

$$inp(j) \ = \ \sum_{i=1}^{n} \ x_j w_{ij} \tag{7}$$

The above input is then fed to the activation function. A bipolar sigmoid activation function is used between the input and hidden layer.

$$f(x) = \frac{2}{(1 \ + \ exp(-x))} - 1 \tag{8}$$

The output of the above function varies from -1 to 1. Here, 1 means the given node is activated. The output of this layer is multiplied by another sets of weights and then fed via a linear function to the output layer. The final output of the module obtained is :-

$$y_k \ = \ \sum_{j=1}^{m} \ w_{jk} f(inp(j)) \tag{9}$$

The above output is then compared with the expected output to calculate the new weights via back propagation. The mean squared function is used as the error function.

$$E_m \ = \ \frac{1}{n}\sum_{k}(T_k \ - \ Y_k)^2 \tag{10}$$

Finding the optimum weight for the neural network is done via gradient descent. Gradient descent is used to look for points where the function reaches its lowest value (local minima). Gradient descent uses the slope to reach such points.

Gradient descent when applied on the error function would give the lowest error possible.

$$w(t+1) = w(t) - \eta\delta E(t)/\delta w(t) + \alpha w(t) \quad (11)$$

The above process of training the ANN is used for every ANN with its corresponding dataset TR.

2.3 Stage III - Fuzzy Aggregation

After the ANNs of Stage II have been trained with their respective datasets, another ANN is trained with the combined details of the previous ANNs. This ANN has the same number of inputs as the number of outputs of the ANNs in Stage II.

1. The outputs of the ANNs in Stage II are multiplied with the corresponding membership value to obtain a new input for Stage III.

   $$Y_j^{TR} = [y_{j1}^{TR}, y_{j2}^{TR}, ...y_{jk}^{TR}], j = 1, 2..., n \quad (12)$$

   In the above equation, n is the number of training sets (ANNs in stage II) and y_{jk}^{TR} is the output of ANN_k for input point j.

2. The input for the new ANN is formed using matrix multiplication. Every value of the output nodes of stage II are multiplied with the membership values of that point. This gives preference to ANNs which have data fed into them with higher membership values as the training data was created using alternative fuzzy clustering.

   $$Y_{input} = [Y_1^{TR}.U_1^{TR}, Y_2^{TR}.U_2^{TR}..., Y_n^{TR}.U_n^{TR}] \quad (13)$$

   where U_n^{TR} is the membership value of how much dataset TR_n belongs to C^{TR}.

3. The final ANN is trained with the above input and the output labels for each of the data points. This is the only ANN that is trained with the combined data set and not small clusters.

During testing, the cluster centers and equation 4 are used to generate the membership values of the test inputs. These test inputs are fed through the ANNs in stage II. The outputs of stage II are multiplied with the calculated membership values and fed to the ANN in stage III. The output of stage III is taken as the label of the test input.

3. EXPERIMENTS AND EVALUATIONS

In this section, we explain the experiments we conducted to test our proposed methodology. For training the neural networks we use the KDD Cup Intrusion Detection Dataset. We then test the trained data against live Internet data and attack toolkit and also against the KDD Cup Testing Dataset. We also present the results and evaluate them in the final subsection.

3.1 Data Preparation

For training our neural net, we have selected the KDD Cup '99 Dataset[5], which is specifically designed for intrusion detection. The training dataset has 498000 labeled data points, each point containing 42 attributes pertaining to the basic features of TCP connections, domain related features and certain time based statistics. The labels consist of either normal or one of 22 attack labels which can be broadly categorized into 4 categories[15]

1. *Denial of Service (DoS):* making some computing or memory resources too busy to accept legitimate users access these resources.

2. *Probe (PRB):* host and port scans to gather information or find known vulnerabilities.

3. *Remote to Local (R2L):* unauthorized access from a remote machine in order to exploit machine's vulnerabilities.

4. *User to Root (U2R):* unauthorized access to local super user (root) privileges using system's susceptibility.

For our training purposes, we first remove duplicates from the original dataset. We end up with less than a million unique data points. From this, we select 11 attributes from the original 42, based on the recommendations of previous works [13],[12]. The 11 selected attributes are presented in Table 1.
Also for our experiments we also decided not to train our ANNs with R2L kind of attacks. Based on our investigations, we realized that most types of R2L attacks in the dataset target vulnerable proprietary protocols, which existed back in 1999 but have been fixed or patched now. Thus, training the ANNs with those attacks would only reduce the efficacy of our intrusion detection system to detect modern-day attacks. From the modified dataset, we select 9600 data points for training. 2500 of those are labeled as Normal, 3500 are DoS, 3550 are Probe, and 52 are U2R. There were only 52 unique U2R labeled points in the dataset, and so we used them all for training.

3.2 Experimental System Setup

For our experiments, chose the following values. In the Fuzzy Clustering Module (Stage I) as described in §2.1, we choose 2 as the value for fuzziness(m). We choose $\epsilon = 0.00005$
In Stage II, we use 11 inputs and 4 outputs of each ANN, corresponding to the 11 attributes selected for training. The hidden layer has 14 neurons based on the following equation[9]:

$$\sqrt{I+O} + \alpha \quad (14)$$

where I equals the number of inputs, O equals the number of outputs and α is a random integer between 1 and 10. Based on the results we observed, we selected $\alpha = 10$
Similarly, the ANN in the aggregating module has 4 inputs, 4 outputs and 13 neurons in its hidden layer. All the ANNs were trained for 500 iterations. The momentum and learning rate (α and η respectively, in Eqn 11.) used are 0.2 and 0.01 respectively.

We tested our IDS on two different setups.

I. Firstly, we tested on a distributed environment as shown in Figure 1. We installed QEMU hypervisor on a Ubuntu 16.04 operating system, and created three VMs on these hypervisors. We run the IDS engine on the Host OS, which will listen to the virtual bridge interface connecting the host and the VMs. One of the VMs will execute normal Internet workloads like browsing Youtube.com, wikipedia.org etc. The second VM will also run normal workloads e.g downloading the big

S.No.	Attribute	Description	Type	Domain Type
1	duration	duration of the connection	continuous	real
2	service	destination service (e.g. telnet, ftp)	discrete	integer
3	src_bytes	bytes sent from source to destination	continuous	real
4	dst_bytes	bytes sent from destination to source	continuous	real
5	count	number of connections to the same host as the current connection in the past two seconds	continuous	real
6	srv_count	number of connections to the same service as the current connection in the past two seconds	continuous	real
7	dst_host_count	count of connections having the same destination host	continuous	real
8	dst_host_srv_count	count of connections having the same destination host and using the same service	continuous	real
9	dst_host_diff_srv_rate	% of connections to the current host having the same src port	continuous	real
10	dst_host_same_src_port_rate	% of connections to the current host having the same src port	continuous	real
11	dst_host_serror_rate	% of connections to the current host that have an S0 error	continuous	real

Table 1: Attribute Description of Dataset

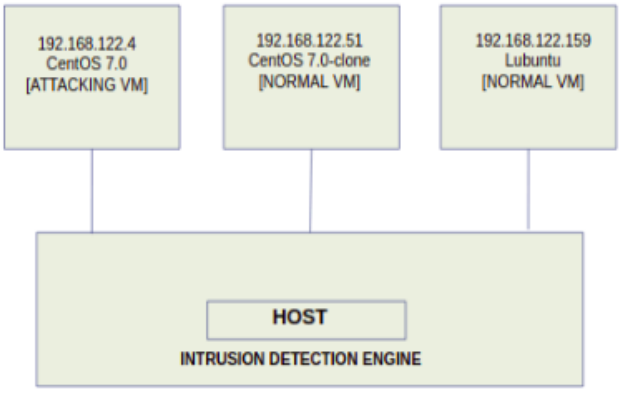

Figure 1: System Design

text file from the internet. The third VM will execute a SYN-flood DoS attack[4] on the 1st VM, and an ICMP-flood DoS attack[3] on the second VM, using the hping3[2] tool.
We perform ten experiments of 1000 packets each of every workload and present the accuracy with which our IDS identified each packet in the next section. We use the scripting tool called Bro[1] to convert raw data packets captures into a format understood by the engine.

II. Secondly, we use the labeled KDDCup'99 Testing Dataset directly on our engine to test the accuracy. We process the testing dataset similarly to how we processed the training dataset in section 3.1. We removed the duplicates, processed the labels as Normal or 3 types of attacks and filtered the 11 attributes we used for training. We present the results of this experiment in the next section. This experiment was conducted since we could not find tools to satisfactorily implement Probe or U2R attacks.

3.3 Results & Discussion

For the clustering phase of the AFCM-ANN framework, data points were distributed, as shown in Figure 2, into 4 clusters. As stated in §2, the clustering algorithm ensures homogeneity within the clusters and heterogeneity among the different clusters. Since 11 features are used for a given packet, different kind of packets are clustered together due to similarity in these feature values. We can see that around 5500 packets are clustered together in Cluster 0 abundantly

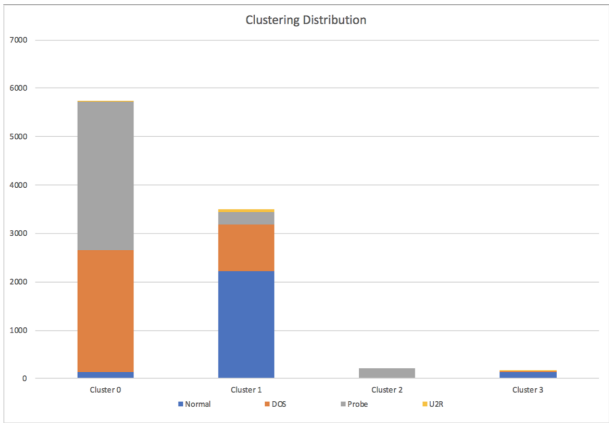

Figure 2: Labels Distribution

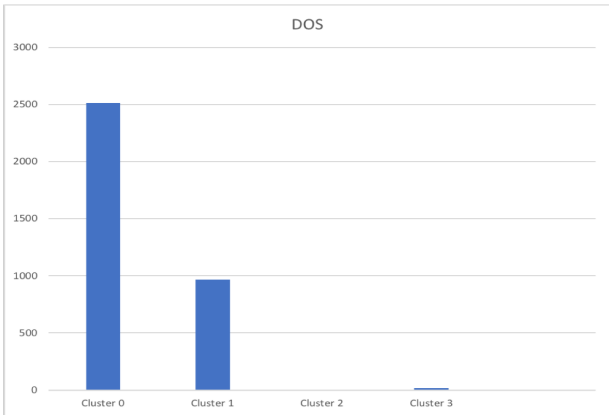

Figure 3: Cluster 0 (DoS Distribution)

consisting of DoS and Probe packets, as shown in Figure 3 and Figure 5 respectively. Cluster 1 contains close to 3500 packets predominated by Normal packets, as shown in Figure 4, with around 900 DoS packets also falling in that cluster and around 45 U2R packets, as shown in Figure 6. Cluster 2 has around 100 packets which are only Probe packets as shown in Figure 2. Cluster 3 also has close to 100 packets composed of Normal packets primarily with traces of U2R as in Figure 4 and Figure 6 respectively.

With these clustered packets, we train 4 Artificial Neural Networks as mentioned in Stage II, Section 2. The accuracy and loss graphs shown for each ANN are plotted as a function of epoch. The accuracy for ANN_0, as reported in Figure 7, is 54.16% for 500 epochs. One of the reasons for this low accuracy can be the presence of all 4 kinds of packets in Cluster 0 owing to the close related feature values present in the training data for certain packets. The loss percentage, which is calculated as mean-square error over epochs is around 50% for ANN_0, as shown in Figure 8. We observe that the accuracy and loss percentages for ANN_1 are 96.56% and 6.58% respectively as in Figure 9 and Figure 10. Since Cluster 1 primarily contains Normal packets, so the prediction for these data points turns out to be of better accuracy than Cluster 0. Similarly, for ANN_2, the accuracy is 97.26% and loss percentage is 5.37% as shown in Figure 11 and Figure 12. The reason for these numbers are again

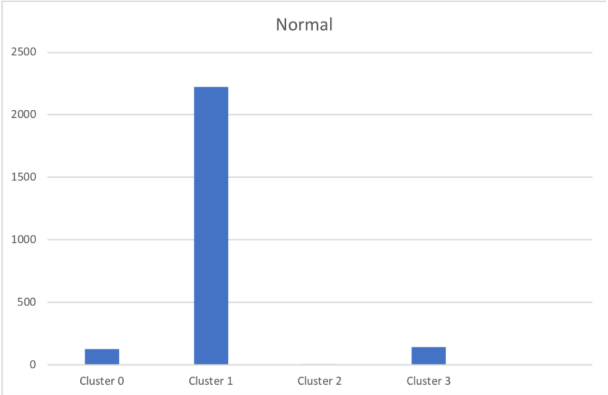

Figure 4: Cluster 1 (Normal Distribution)

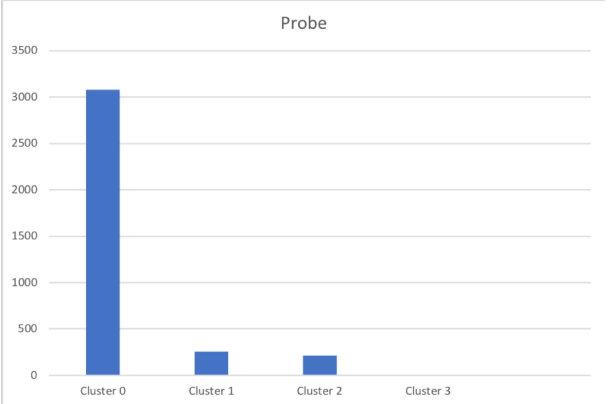

Figure 5: Cluster 2 (Probe Distribution)

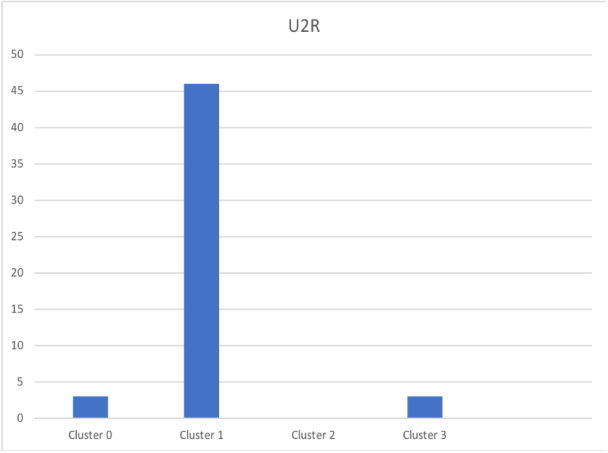

Figure 6: Cluster 3 (U2R Distribution)

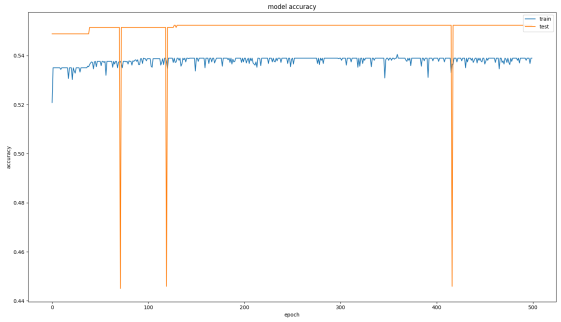

Figure 7: Accuracy for ANN 0 in Stage II

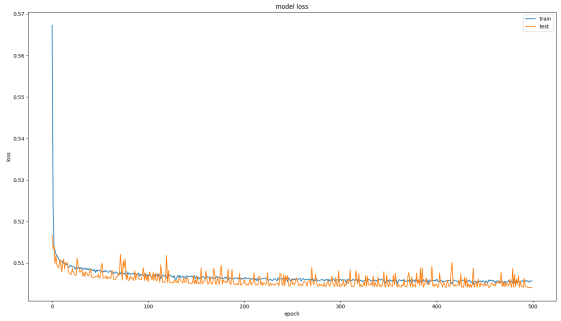

Figure 8: Loss for ANN 0 in Stage II

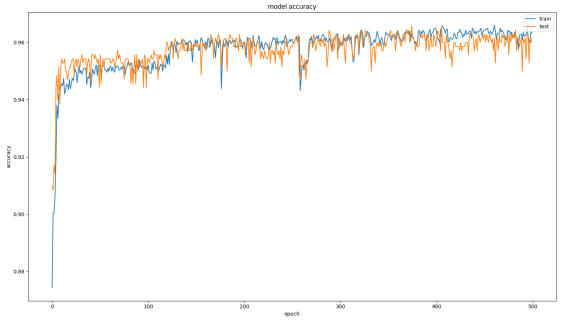

Figure 9: Accuracy for ANN 1 in Stage II

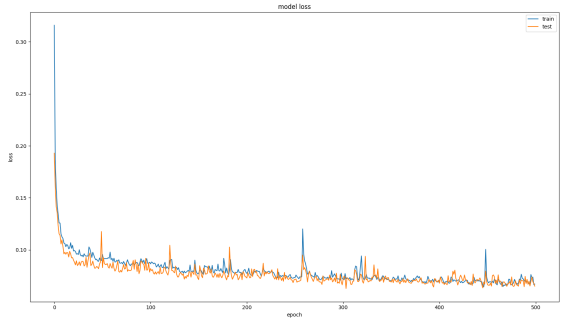

Figure 10: Loss for ANN 1 in Stage II

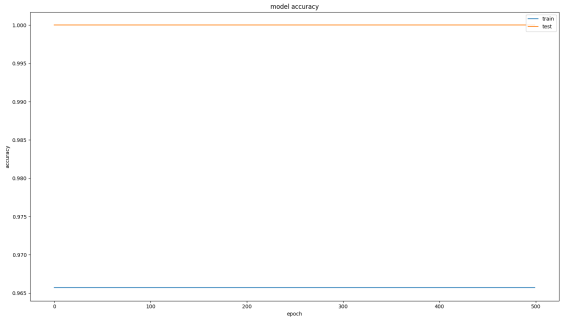

Figure 11: Accuracy for ANN 2 in Stage II

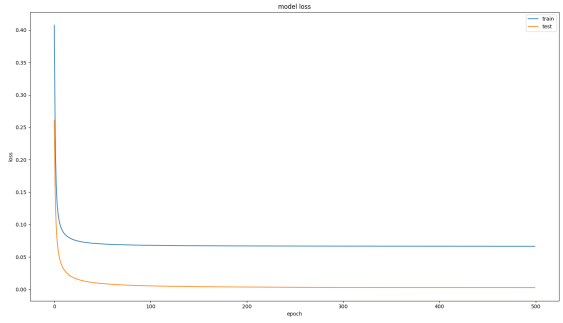

Figure 12: Loss for ANN 2 in Stage II

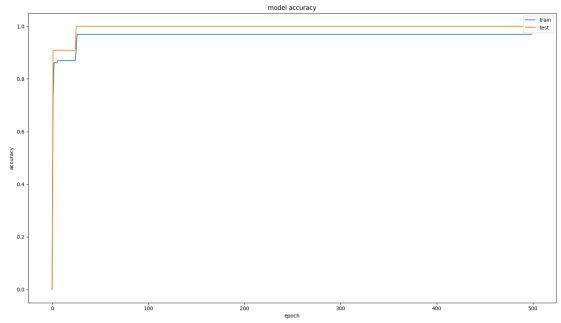

Figure 13: Accuracy for ANN 3 in Stage II

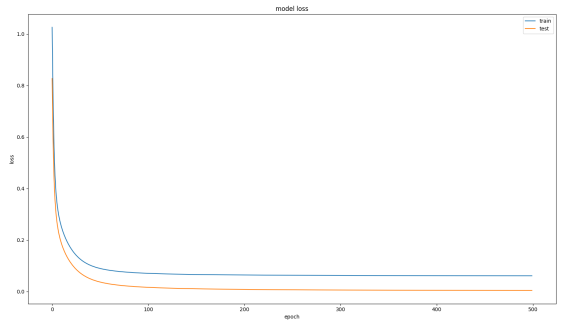

Figure 14: Loss for ANN 3 in Stage II

homogeneity of Cluster 2 which is completely composed of Probe packets. And for ANN_3, the reported accuracy as in Figure 13 over 500 epochs is 97.55% and loss percentage is lowest of all the ANNs, which is 4.98%, as in Figure 14.

After obtaining all the ANNs trained over 4 clusters, we

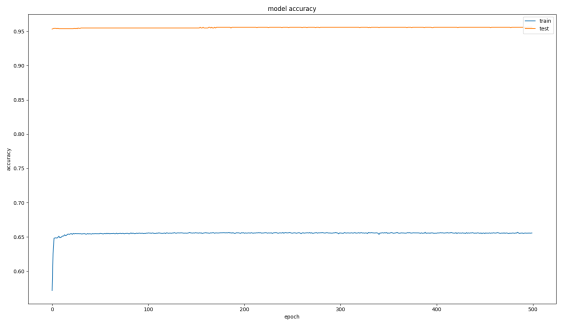

Figure 15: Accuracy for ANN in Stage III

bring up another new ANN for the Fuzzy Aggregation stage of the AFCM-ANN framework. As we can see in Figure 15, the accuracy for this model is 76.77% and loss percentage is 29.04%, Figure 16. These values are obtained after training this ANN over all the training data, TR, combining output from Stage II ANN and membership values from Stage I.

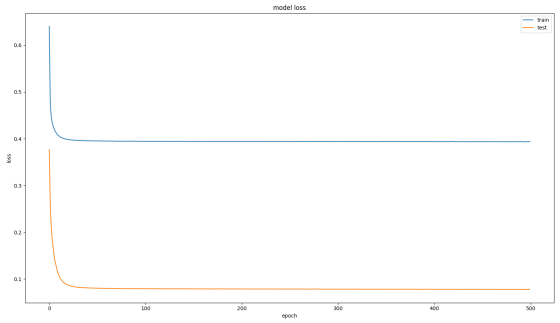

Figure 16: Loss for ANN in Stage III

The system was tested across three test cases. Two DoS attacks and a normal internet workload like browsing youtube.com, wikipedia.org etc. One of the VMs also performed a wget to download a large file from the Internet. The results of the above tests are given in Table 2.

The engine was also tested with KDD Testing dataset. The results are given in Table 3. We can see that the results for probe are low and U2R as non existent. Although the engine is unable to pinpoint the attack specifically, it identifies a majority of probe and U2R connections as attacks and so raises an alarm, which serves the purpose of blocking that particular packet for security reasons.

4. RELATED WORK

Intrusion Detection Systems(IDS) have been developed in two formats. There is a signature based IDS and an anomaly based IDS. The signature based IDS is based on a set of rules which were derived from the definitions of packets from attacks. Signature based IDSs can easily detect existing attacks with high accuracy, however they fail against unknown attacks for which the rules and patterns are not present. Anomaly based IDSs were introduced to be able to detect unknown attacks. They are based on a model which learns to distinguish trustworthy activities from unsafe activities. Due to this decision they suffer from false positives.

Signature Based IDS : Lindqvist and Porras (1998) [10] worked on a signature based IDS which relied on creating definitions for attacks and rules to determining them. This failed with new attacks and also required frequent manual updates to detect new attacks. Shyu, Chen, Sarinnapakorn & Chang (2003) [14] expanded on the rule based IDS by using statistics and principle component analysis, however creating a statistical based IDS involves creating a complicated mathematical model which is not possible for network traffic which has lots of parameters and variations.

Anomaly Based IDS : Since signature based IDS weren't able to detect previously unknown attacks, anomaly based IDSs began to emerge. Dokas et al (2002) [7] worked with data mining to encounter issues with unknown attacks. This however presented another issue of low accuracy. Mukkamala et al [11] worked with supervised neural networks to increase accuracy. Gang Wang et al [15] added fuzzy clustering and fuzzy aggregation to neural networks in order to increase accuracy for attacks with low frequency.

Test Case	Total Number of Connections	True Positive	False Negative	Std. Deviation
SYN Flood	10000	9890	110	0.2242
ICMP Flood	10000	9990	10	0.0094
Normal Activity	10000	8809	1174	15.04

Table 2: Results of attacks and normal activity

Test Case	Total Number of Connections	True Positive	False Negative
Normal	47893	47672	221
DoS	19385	16136	3249
Probe	2430	666	1764
U2R	202	0	202

Table 3: KDD Test Dataset results

5. LIMITATIONS AND FUTURE WORK

While our project showed promising results with normal packets and DoS attacks, our engine fails to identify Probe or U2R packets with high accuracy. There are certain factors which can attributed to this. There were only 52 packets in the training data set with U2R label. The lack of proper training for this kind of attack might be a reason for low accuracy. We also selected the 11 attributes for training based on recommendation from previous work. We suspect that if we used the entire feature list for training, we would have certainly ended up with better results. Alternatively, we could have experimented with different sets of attributes to see which ones provide better results.

Also, currently our experiments were conducted on a single physical host distributed system. Due to connectivity limitations with VCL, we could not run our engine on a actual cloud system. Future work can make efforts in this direction.

Our engine is an offline machine learning system. Future work can include transforming this into an online learning model or a batch learning model.[8] This might be challenging in terms of system overhead and performance, but definitely worth looking into.

Another avenue worth exploring is to synergize an intrusion detection system with an intrusion prevention system. For example, if a certain number of attacks are received from a host, that host might be blocked from sending any more packets by using iptables and firewalls.

6. CONCLUSION

Intrusion Detection Systems become essential in ensuring the security of cloud systems in modern time. Firewall acts as the first wall of defense but if any security breach successfully passes the firewall or appears as an internal threat, presence of IDS can help in blocking such threats. In this paper, the proposed algorithm uses Alternative Fuzzy c-means Clustering method and Artificial Neural Networks to achieve this purpose for different kinds of attacks. The experiments ran on the live internet traffic show promising results in detecting the attacks. If the engine is able to differentiate well between a attacking and non-attacking packet, the severe consequences that any particular attack might have had can be prevented.

7. APPENDIX

The code base for the project can be found on https://github.ncsu.edu/dnag/fuzzyclustering-ids

8. REFERENCES

[1] The bro network security monitor. https://www.bro.org/.

[2] hping3 manual. https://linux.die.net/man/8/hping3.

[3] Understanding icmp flood attacks. https://www.juniper.net/documentation/en_US/junos/topics/concept/denial-of-service-network-icmp-flood-attack-understanding.html.

[4] What is a syn flood attack? https://www.incapsula.com/ddos/attack-glossary/syn-flood.html.

[5] Bolon-Canedo, V., Sanchez-Marono, N., and Alonso-Betanzos, A. Feature selection and classification in multiple class datasets: An application to kdd cup 99 dataset. *Expert Systems with Applications 38*, 5 (2011), 5947–5957.

[6] Chiba, Z., Abghour, N., Moussaid, K., Omri, A. E., and Rida, M. A survey of intrusion detection systems for cloud computing environment. In *2016 International Conference on Engineering MIS (ICEMIS)* (Sept 2016), pp. 1–13.

[7] Dokas, P., Ertoz, L., Kumar, V., Lazarevic, A., Srivastava, J., and Tan, P. N. Data mining for network intrusion detection. In *Proceedings of the NSF Workshop on Next Generation Data Mining,* Baltimore (2002).

[8] Ghosh, A. K., Schwartzbard, A., and Schatz, M. Learning program behavior profiles for intrusion detection. In *Workshop on Intrusion Detection and Network Monitoring* (1999), vol. 51462, pp. 1–13.

[9] Haykin, S. *Fundamental Principles of Optical Lithography.* Prentice Hall, 1999.

[10] Lindqvist, U., and Porras, P. A. Detecting computer and network misuse through the production-based expert system toolset (p-best). In

Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No.99CB36344) (1999), pp. 146–161.

[11] Mukkamala, S., Janoski, G., and Sung, A. Intrusion detection using neural networks and support vector machines. In *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on* (2002), vol. 2, pp. 1702–1707.

[12] Natesan, P., and Balasubramanie, P. Multi stage filter using enhanced adaboost for network intrusion detection. *International Journal of Network Security & Its Applications 4*, 3 (2012), 121.

[13] Pandeeswari, N., and Kumar, G. Anomaly detection system in cloud environment using fuzzy clustering based ann. *Mob. Netw. Appl. 21*, 3 (June 2016), 494–505.

[14] Shyu, M.-L., Chen, S.-C., Sarinnapakorn, K., and Chang, L. *Principal Component-based Anomaly Detection Scheme*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 311–329.

[15] Wang, G., Hao, J., Ma, J., and Huang, L. A new approach to intrusion detection using artificial neural networks and fuzzy clustering. *Expert Syst. Appl. 37*, 9 (Sept. 2010), 6225–6232.

[16] Wu, K.-L., and Yang, M.-S. Alternative c-means clustering algorithms. *Pattern Recognition 35*, 10 (2002), 2267 – 2278.