LEC Journal 2022, 4(1): 38-42
ISSN: 2565-5205 (print)

Research Article

ANOMALY-BASED NETWORK INTRUSION DETECTION SYSTEM

[1]Anil Verma, [1]Enish Paneru, [1]Bishal Baaniya*

[1]Department of Electronics and Computer Engineering, Pulchowk Campus, Tribhuvan University, Pulchowk, Lalitpur, Nepal

*Corresponding author email: bishalbaaniya@gmail.com

ABSTRACT

Network security has been a really hot topic since the inception of the internet in the early '80s. With millions of people entrusting their life savings in the hands of an organization, it is really necessary to keep the network intruders out of the system. The most alarming thing is that - even today, many organizations are detecting these intrusions through manual labour. Many researchers have proven that these intrusions have a certain pattern i.e. they can be detected with an Artificial Intelligence (AI) based system with enough training which can prove to be a really an effective substitute for manual labour. This paper explains the current trends in Network Intrusion Detection and the technologies that have been implemented to detect them. CICIDS2017 dataset containing around 3 million data points was used in this experiment. K-Nearest Neighbours (KNN) and Random Forest algorithms are used as the AI tools and their performance has also been compared

Keywords—Network, Intrusion, Artificial Intelligence, Random Forest, KNN

I. INTRODUCTION

Computing technology has its roots in ny of the day-to- day activities. It is a rapidly growing field with a lot of money involved. This has attracted a lot of people to earn a lot of money in a very short time in the form of cyber-crime. As the use of information technology has increased, we have observed a similar rising trend in terms of cyber-crimes. What's more is that, in this age of globalization, cyber criminals have found it ever easier to increase their span of possible victims. Unfortunately, most people are not aware of the disturbing increase in cybercrimes and ways to prevent them which calls for a new solution – an advanced system that can quickly detect any anomalous behavior in the network and inform the network manager about the criticality of the situation.

A.Cyber Attack

In short, cyber-attack is any type of malicious activity which intends to illegally breach computer-based system in order to gain access to information systems, network or data in general. Some of the most recognizable cyber-crimes are:

1) *Malware*: Any type of software that is intended to disrupt or gain unauthorized access to a computer-based system is known as malware. These are mainly present in the form of worms, viruses and trojan software which when installed on a computer can provide access to the computer to the attacker.
2) *Phishing*: Also referred to as "social engineering attack", it is a kind of attack in which the attacker claims to be from a reputable source and asks the victim about sprosensitive information like login credentials, credit card numbers, etc. This form of attack has been increasingly common over the last decade.
3) *DoS Attack*: Denial-of-service (DoS) attack is a type of attack in which the infiltrator renders the victim's service unavailable to its intended users by deliberately flooding the network with exorbitant surge of requests consequently exhausting the bandwidth of the server. If this same attack is done by synchronizing multiple compromised computers, then this is known as a distributed-denial-of-service (DDoS) attack.

5) SQL Injection: In this type of attack, the attacker uses malicious SQL queries/statements in order to infiltrate data-driven applications. The attacker uses this technique to carry out CRUD operations on the system in order to gain advantage.

B.Intelligent NIDS

The main goal of an Intrusion Detection System (IDS) is to monitor the network traffic and to report any malicious activities to the network monitor (a human) so that the attackers can be identified as soon as possible and corrective measures can be taken. Though most traditional IDS can detect systems they are trained to identify, they fail to give desired output when they are exposed to an unknown attack. Machine Learning is a rapidly emerging technology. It has found its place in security analysis. Anomaly based IDS are very effective which can be created with various machine learning algorithms like ANN, Random Forest, SVM. Random Forest is a tree-based classifier that has good accuracy in detecting intrusions. Artificial Neural Network (ANN) is an algorithm that uses a neural network to filter the incoming data and forwarded to the expert system. Support Vector Machine (SVM) is a type of learning method that attempts to find a global solution to the optimal value of non-linear classification problems. From several studies showed that ANN and SVM has a good detection accuracy on anomaly-based IDS. Intrusion Detection System is one component of network security that is used for monitoring data traffic and detect if there is a specific activity which is recognized as an intrusion.

II. OBJECTIVES

The objectives of this research are as follows:

- Detect anomalies: to develop a system capable of detecting attacks and anomalies as they occur in real time
- Provide network traffic information: provide traffic flow information and features for deeper analysis of network traffic behavior

III. METHODOLOGY

A. System Architecture

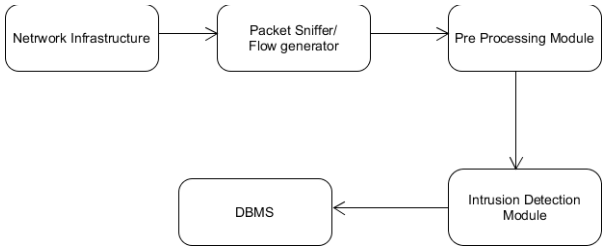

Fig. I: Block Diagram for model train

This is the baseline architecture for the research and with the following sub-components:

1) Network Infrastructure: Network Infrastructure refers to any source that can feed the system with raw TCP and UDP packets. It can be a live computer network, any specific device generating network traffic or an offline source of packet data. This subsystem can be said to be the infrastructure under analysis for the intrusion. For best results, it is desired to be the gateway, so that maximum traffic can be interpreted.

2) Packet Sniffer/flow generator: Network traffic capture is the subsystem responsible to sniff the network packet from the network infrastructure. It consists of a computer program that can intercept and log the network traffic that passes over a network. This module is responsible for generating a flow from the live TCP and UDP traffic and extracting flow features necessary to feed into the system like backward packet minimum length, mean of forwarding flow IAT, ACK flag count, etc. These feature values are then fed into the system.

3) Preprocessing module: This module is responsible for making the data ready for analysis as required by subsequent subsystems. Its features overlap a little with the packet sniffer module as it also selects/ calculates the required features from the traffic flow/ connection. It also applies dimensionality reduction algorithm on the extracted data so that it can be fed to the intrusion detection engine.

4) Intrusion Detection Module: This is the core of the system, responsible for classifying the flows captured as normal or anomalous. During the training phase of the system, a Random forest classifier model and a K nearest Neighbors classifier model were trained and validated with the CICIDS2017 dataset. Those models were used to predict the behavior of any traffic flow captured

5) *DBMS:* This is the core of the system, responsible for classifying the flows captured as normal or anomalous. During the training phase of the system, a Random forest classifier model and a K nearest Neighbors classifier model were trained and validated with the CICIDS2017 dataset. Those models are used to predict the behavior of any traffic flow captured.

B. System Implementation

The whole system development can be characterized in the following subtasks:

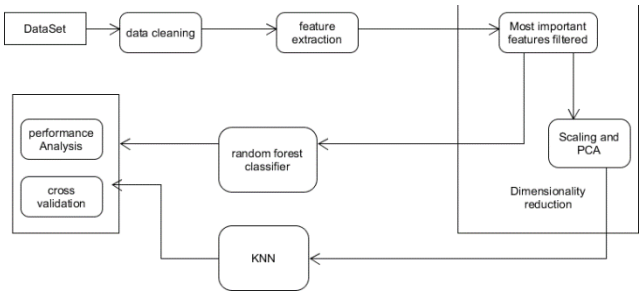

Fig. II: System Architecture

1) Tasks during training:

a) Dataset Analysis: The dataset consists of around 3 million data points with a few anomaly labels ranging to hundreds of thousands while some are even less than 100 in number. All the others are of benign labels. There are more than 85 features in the dataset. However, only TCP and UDP protocols are available in the dataset.

b) Dataset Preparation: Data with missing label values were dropped since they posed no advantage to the training. Other missing values were taken as the average of the column values. Infinite values were dropped or taken as the maximum of the column values as per the characteristics of the feature.

$$Gini\ index = 1 - \sum_j p^2_j \qquad (1)$$

The data were fed into the Random forest classifier and random forest regression. Then the feature importance of each feature in the dataset was analyzed as per the impurity reduction based on the Gini index and variance from the above two models. This was done to filter out the most important features from the whole dataset.

It was found that the most important features were around the same ballpark as per the results from both models.

The data after this step was used as it is for the random forest classifier model. But before feeding it to the K nearest neighbor model, the dataset was first scaled with a standard scaler and then principal component analysis (PCA) was applied to it to reduce the feature set into 20 principal components.

c) Standard Scaler: During standard scaling, features are scaled based upon:

$$(x_0 - m(x))/standardDeviat(x) \qquad (2)$$

The total explained variance ratio after the application of PCA was obtained to be at 98.2 %.

d) Model Training: The models used for training were random forest and K nearest neighbors. All the labels in the processed dataset were grouped into two; benign and anomaly.

Then the data were fed into the model and thus trained binary random forest and KNN classifiers were obtained.

For Random Forest classifier:

Three models for random forest classifier were developed with 10, 20 and 30 trees respectively. The following were the specifications for the random forest classifier.

- max_features: The number of features to consider when looking for the best split was set as the square root of total features.

- min_samples_split: The minimum number of samples required to split an internal node was set as 2.
- max_depth: No maximum depth was set for each tree in the forest. Hence the nodes were expanded until all leaves were pure or until all leaves contain less than min_samples_split.

The predictions were done by taking majority votes from classification trees.

For KNN:

Similarly, 3 models were used for the KNN classifier with k= 10, 15 and 20. Following were the specifications for the KNN classifier:

- Weights: All points in each neighborhood were weighted equally.
- Algorithm: The algorithm used to compute the nearest neighbors was set to auto so that the most appropriate algorithm among BallTree, KDTree and brute-force search was decided based upon the data the model was fitted with.
- Leaf_size: The leaf size passed to BallTree or KDTree was set to 30.
- Metric: Standard Euclidean metric was used as the distance metric for the tree.

2) Flow Generation:

Even though the model has been trained it is still needed that the features fed into the model should be extracted from a live network.

During real-time traffic capture, TCP and UDP packets are initially sniffed from the network. The captured packets are then assigned to their corresponding traffic flows based on their source and destination IPs and ports. After flow establishment, whenever the flow closes, either due to a FIN flag or timeout (set to 120 sec as default) all the aforementioned features are extracted from the flow. Those values are fed into the trained classifier model for behavior prediction.

3) Database setup:

MongoDB has been used for this research for storing all the flow-based features along with its classified behavior and date of capture. Whenever the analysis of the flow capture is to be accessed the DBMS is invoked, whether it may be live capture or reviewing the historical records.

4) Deployment:

The system follows a client-server architecture. The research has been currently deployed as a WSGI application and been tested on a green unicorn server. The sessions, user authentication, and all other middleware are custom made.

The processed dataset was cross-validated and split into training and testing data in a 9:1 ratio. After the random forest classifier and KNN models were trained, they were subjected to predict upon both the training and testing data.

Fivefold cross-validation test was applied with the scoring as a mean squared error. During the 5 fold cross-validation test, the dataset is split into 5 parts with 4 parts used for training and the remaining part for testing. This is repeated 5 times with the parts between training and testing interchanged. During each fold negative mean square error is calculated.

The model was cross-validated with random forest classifiers with 10, 20 and 30 trees and similarly for KNN with K=10, 15 and 20.

The cross-validation scores are given as:

TABLE I: Cross-validation Score for Random Forest Classifier

Number of trees	1st fold	2nd fold	3rd fold	4th fold	5th fold
10	0.01155	0.01150	0.01175	0.01119	0.01167
20	0.01122	0.01148	0.01116	0.01143	0.01153
30	0.01130	0.01100	0.01135	0.01136	0.01291

TABLE II: Cross-validation Score for K Nearest Neighbours

K	1st fold	2nd fold	3rd fold	4th fold	5th fold
10	0.01384	0.01359	0.01362	0.01388	0.01388
15	0.01449	0.01445	0.01432	0.01426	0.01404
20	0.01429	0.01430	0.01445	0.01443	0.01441

6) EXPERIMENTS AND RESULTS

For both models, various evaluation metrics and confusion matrices were tabulated for a vivid comparison and performance analysis.

- True Positive: Anomalies identified as anomalies
- False Positive: Benign identified as anomalies
- True Negative: Benign identified as benign
- False Negative: Anomalies identified as benign
- Precision: Positive predictive value given as
  PPV = TP/(TP+FP)
- Recall: True Positive Rate given as
  TPR = TP/(TP+FN)

TABLE III: Training dataset for random forest classifier

Matrics	Trees = 10	Trees = 20	Trees = 30
True positive	421618	422831	423353
False positive	1336	1082	1072
True Negative	2121538	2122005	2121922
False Negative	2193	730	332
Precision	0.996	0.997	0.997

TABLE IV: Testing dataset for random forest classifier

Matrics	Trees = 10	Trees = 20	Trees = 30

True positive	44808	45231	45094
False positive	1057	1082	1082
True Negative	234949	234674	234804
False Negative	2152	1979	1986
Precision	0.976	0.976	0.976

TABLE V: Training dataset for KNN

Matrics	K = 10	K = 15	K = 20
True positive	400101	400178	397671
False positive	6176	8368	7127
True Negative	2116869	2114570	2115719
False Negative	23539	23569	26168
Precision	0.976	0.979	0.982

TABLE VI: Testing dataset for KNN

Matrics	K = 10	K = 15	K = 20
True positive	44223	44084	43841
False positive	933	1091	981
True Negative	234902	234851	235053
False Negative	2908	2940	3091
Precision	0.979	0.975	0.978
Recall	0.938	0.937	0.934
Mean square error	0.013	0.014	0.014
Execution time (sec)	33.69	38.05	41.8

For random forest classifier, high precision and recall were observed in the training dataset which might point to the model being over fitted. However, it can also be seen that the evaluation metrics are very high for the testing datasets as well which shows that the results are both highly useful and complete. While considering the testing dataset, among different models used, the one with 30 numbers of trees proved to be slightly better in terms of accuracy, precision, and recall. However, the execution time nearly tripled when compared to the one with 10 trees in the forest. Hence, performance-wise, the one with 10 trees can be considered more useful and relevant.

For KNN classifier, the one with K=10 proved to be superior while considering precision, recall, accuracy and execution time. Hence this model was used as the ideal one for the KNN classifier.

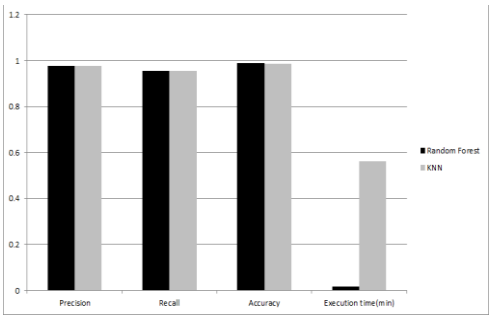

Fig. III: Bar Diagram for Model Comparison

But when the final selected models were analyzed, random forest classifier matched the KNN classifier in precision, showed better accuracy and recall and definitely outmatched in execution time. Hence, considering the analysis and evaluation upon testing dataset, random forest proved to be superior. The system was then subjected to various attack tools available. The performance of the system was analyzed by using a local host device on the monitored network as the victim the first time and testing on example.com the other time. The results were averaged from 5 tests for each case. The performance of this system with those tools is given below:

1) *Slowloris*: Python package for Slowloris was download from pip and installed on a machine in the network. Then Slowloris command was called upon the target victim and the network was monitored for 10 minutes. During this attack the results from the system was observed as:

TABLE VII: Performance analysis for slowloris

Analysis		With local victim host	With example.com
RF Classifier	Total flows predicted as anomaly	103	231
	Total flows involving victim's IP	426	791

2) *Hulk:* Python program for hulk was used to test the target victim and the network was monitored for 10 minutes. The results were as follows:

TABLE VIII: Performance analysis for Hulk

Analysis		With local victim host	With example.com
RF Classifier	Total flows predicted as anomaly	98	233
	Total flows involving victim's IP	392	703
KNN Classifier	Total flows predicted as anomaly	125	357
	Total flows involving victim's IP	401	799

3) Http Flooding: Apache Benchmark was used to flood the target victim with loads of request and the results were obtained as follows:

TABLE IX: Performance analysis for HTTP Flooding

Analysis		With local victim host	With example.com
RF Classifier	Total flows predicted as anomaly	101	132
	Total flows involving victim's IP	547	566
KNN Classifier	Total flows predicted as anomaly	201	238
	Total flows involving victim's IP	643	627

4) *Goldeneye*: Lastly, goldeneye was used to test the network and the results were analyzed for 10 minutes:

TABLE X: Performance analysis for GOLDENEYE

Analysis		With local victim host	With example.com
RF Classifier	Total flows predicted as anomaly	86	201
	Total flows involving victim's IP	354	821
KNN Classifier	Total flows predicted as anomaly	129	368
	Total flows involving victim's IP	377	857

As we can see in all the cases KNN classifier has shown better results than random forest classifier as it has predicted most anomalies. The weaker performance of random forest classifier might be because of it being over fitted and thus memorized the noise in the dataset. But the downfall with the KNN classifier is that it takes much more time for execution and thus random forest might overtake it when higher traffic load is involved.

7) CONCLUSION

A random forest classifier and a KNN classifier model was trained with the CICIDS2017 dataset and tested this system against various attack tools. The system not only had excellent evaluation metrics with testing datasets but also showcased good performance against real-life attacks.

The network infrastructure of the system can be varying and the point of implementation must be strategic to sniff out all the packets in the network to be monitored. However, the generic implementation remains the same as per given above. Based on the application area, need, and complexity, the system should be specialized to monitor all the relevant aspects of the network.

This system can be thus useful for network analysts to monitor and secure a network and also to study the behavior of the traffic flows. During the research, the crucial aspects involved in the intrusion attacks and various measures that can be taken upon those attacks were understood. The strength and usefulness of machine learning in network security and the ways it can support against the ever-growing threats of cyber- attacks were also recognized.

References:

[1] B.Pfahringer, "Winning the KDD99 Classification Cup: Bagged Boosting," in SIGKDD Explorations, 2000.

[2] Q. Yang, Li, F., "Support Vector Machine for Intrusion Detection Based on LSI Feature Selection," Intelligent Control and Automation, WCICA, 2006.

[3] J. Ma and S. Perkins, "Online novelty detection on temporal sequences" ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), Washington, DC, Aug. 2003.

[4] A. Ihler, J. Hutchins, and P. Smyth, "Adaptive event detection with time- varying Poisson processes" ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD), Philadelphia, PA, Aug. 2006.

[5] S. A. Hofmeyr, S. Forrest, "Architecture for an artificial immune system" IEEE Trans. on Evolutionary Computation, vol. 8, N4, 2000, pp. 443-473

[6] S. Rawat, K. Pujari, and V. P. Gulati, "On the Use of Singular Value Decomposition for a Fast Intrusion Detection System", Views On Designing Complex Architectures (VODCA-04), 2004.

[7] Y.H. Liao, V. R. Vemuri, "Use of K-Nearest Neighbor classifier for intrusion detection", Computers & Security, 21(5), pp 439448, 2002.

[8] Y. H. Liao, V. R. Vemuri, "Using Text Categorization Techniques for Intrusion Detection", Proceedings of the 11th USENIX Security Symposium, pp.51-59, August, 2002.

[9] T. Liu, Z. Chen, B.Y. Zhang et al "Improving Text Classification using Local Latent Semantic Indexing", ICDM2004, pp. 162-169, 2004.

[10] S. Mukkamala, G. I. Janoski, and A. H. Sung, "Intrusion Detection Using Support Vector Machines", Proceedings of the High Performance Computing Symposium (HPC 2002), pp. 178-183, San Diego, 2002.