INDIAN INSTITUTE OF TECHNOLOGY KANPUR

CS 658A Semester 2021–2022-II

Snort Based Intrusion Detection ft. ML Models

Team - CICADA3301

Members
Archi Gupta
Roll-21111014

Hirak Mondal
Roll-20111022

Professor:
Dr. Sandeep Shukla
Department of Computer Science

Manjyot Singh Nanra
Roll-21111038

Neeraj Chouhan
Roll-21111044

Sharanya Saha
Roll-21111056

Keywords - Snort, Malware Detection, Intrusion Detection, Machine Learning

1 SUMMARY

1.1 Problem Statement

Integrating machine learning models with SNORT for improvised and efficient intrusion detection

1.2 Goals

- Implementing different ML classification algorithms for intrusion detection using SNORT
- Demonstration and evaluation of the live working of our modified NIDS by performing attacks from Kali Linux Machine to Ubuntu Machine
- Integrating ML model with SNORT for Intrusion Detection

1.3 Data Sources

- NSL-KDD(1)
- Data set contains 43 features per record,
  - 41 of the features referring to the traffic input itself
  - last two are labels (whether it is a normal or attack) and Score (the severity of the traffic
- Why NSL-KDD?
  - After pre-processing logs from SNORT traffic, we get 6 overlapping features between the SNORT Traffic and NSL-KDD Dataset
  - Features are: (Duration, Protocol, Src_bytes, Dst_bytes, Count, Srv_Count) Count and Srv_count are time based features derived from the Snort Traffic.

1.4 Goals Achieved

- Implemented different ML classification algorithms for intrusion detection using SNORT
- Demonstrated and evaluation of the live working of our modified NIDS by performing attacks from Kali Linux Machine to Ubuntu Machine
- Integrated ML model with SNORT for Intrusion Detection

2 Short Explanation of the Problem

Snort IDS is an open-source network security tool. It can search and match rules with network traffic data in order to detect attacks and generate an alert. One of the greatest challenges of today's rule-based network intrusion detection system (NIDS) is the largest value of its false-positive rate which makes the rule-based NIDS system unreliable. Our main goal is to deliver an intrusion detection system with lower false-positive rate.

3 Solution Strategy and Architecture

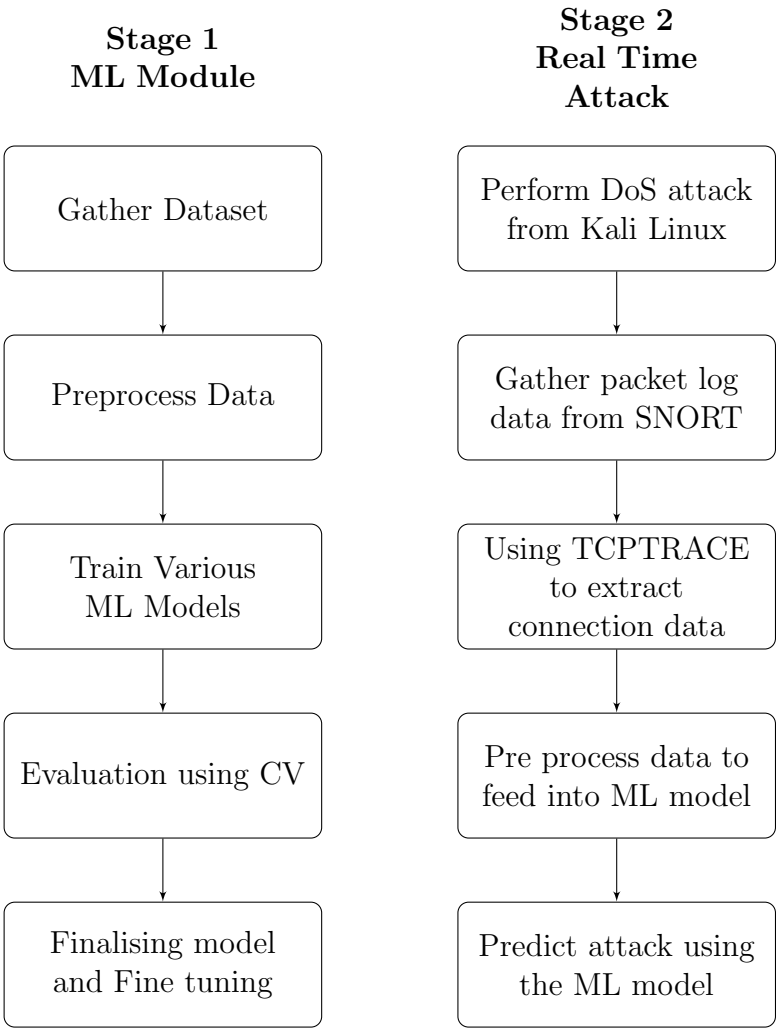

Figure 1: Steps

To avoid large values of false-positive rate we plan to deploy ML algorithms that can help in improving the current rule-based NIDS. We have used different supervised ML-algorithms to classify a connection as a 'normal' connection or an 'attack'. We train the algorithms on publicly available dataset NSL-KDD and perform cross-validation to check which algorithm gives the lowest false alarm rate and the best accuracy.

We pick the algorithm that gives best performance on the desired metrics. We use this algorithm to predict real time attacks. Real time attacks are DOS attack performed from Kali Linux machine to Ubuntu machine, where snort is installed. Snort will capture the packet log and we extract connection information and use this to perform prediction.

4 Experimental Setup

4.1 Simulating DoS Attack

- Software Used
  - VMware Workstation 16 Player(2)
  - Ubuntu-20.04.4(3)
  - Kali-Linux-2022.1(4)
  - Metasploit(5)
  - SNORT(6)
- Performing the attack

  We have performed the Denial of Service (DoS) Attack in an virtual environment. We have had 2 virtual machines, one with Kali Linux installed (the attacker) and the other one with Ubuntu (the target). At first we have searched for an open port using **nmap** in the target machine and when we have found an open port we started TCP SynFlood attack using **metasploit** by setting the RHOST to the IP address of the target machine and the RPORT to the open port of the target machine. We have observed a huge surge in the incoming traffic followed by massive utilization of CPU of the target machine as depicted in Figure 2. We can see that the usage of CPU1 is 78.8% and CPU3 is 86.4% which is a clear indication of exploiting system resources using DoS. Moreover we can see a sudden increase in network traffic which is also a result of this attack.

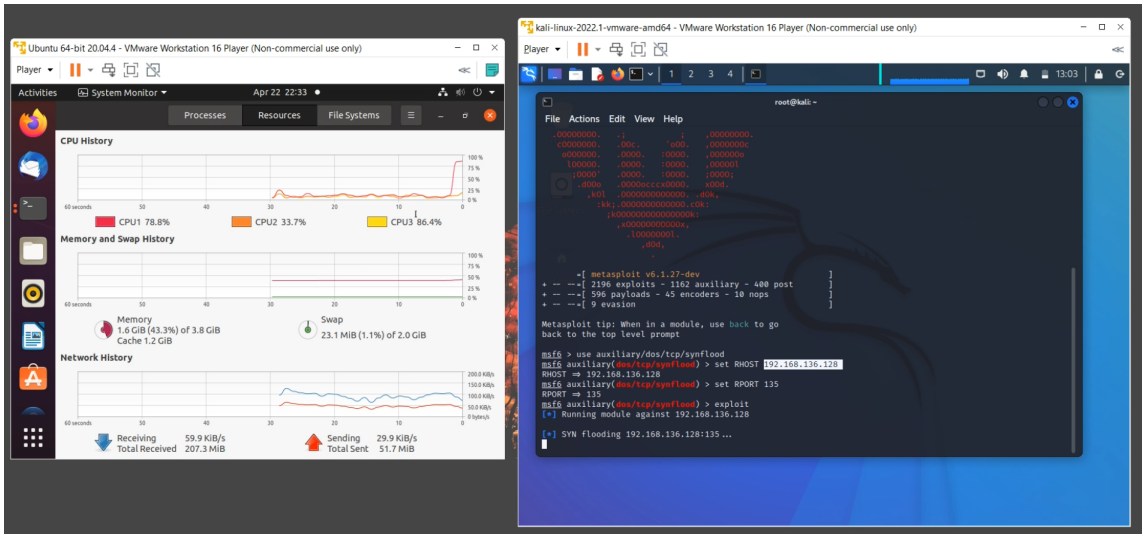

Figure 2: System Logs during DoS

4.2 Sniffing packets using SNORT

While performing the DoS attack we have taken the help of SNORT to capture the packets. But for our machine learning model we needed connections instead of packets, so we converted the packets to connections using **tcptrace**.

4.3 Dataset

We settled on NSL-KDD for model training after experimenting with a few datasets. The six overlapping features with the snort logs were one of the motivating factors for choosing NSL-KDD over any other dataset. It contains 43 features per record, 41 of which describes the traffic input. The last two features are labels, one of which specifies the type of attack and another tells about the severity of the attack.
The dataset mainly consists of four types of attacks:

- Denial of Service (DoS)
- Remote to Local (R2L)
- User to Root (U2R)
- Probing

4.4 Dataset Preprocessing

4.4.1 NSL-KDD Dataset

We processed NSL-KDD dataset by eliminating feature which were not relevant in detecting attacks and features that didn't overlap with features we get from Snort Logs. Directly overlapping features with snort logs are:

- Duration
- Protocol
- Src_bytes
- Dst_bytes

We further derived two additional time based features count and Srv_count from the snort logs. Thus, we used total 6 features from NSL-KDD dataset for training our ML Model. We preprocessed the dataset by verifying presence of missing values, feature scaling and normalizating the dataset. The category labels were handled with label encoder. To perform binary classification, we replaced all the attack labels with 'attack', thus forming two classes **attack** and **normal**.

4.4.2 Snort Logs

Coming to processing of snort logs, we convert the snort logs obtained after performing DOS attack to csv file format so that it can be used by ML algorithm. This is done by extracting connection based information from snort logs using "tcptrace" command on linux machine. The output of tcptrace is preprocessed using python scripts to extract relevant information from each connection and add them to csv file.

4.5 Evaluation Metric

We have used two evaluation metric to determine the best suitable ML-algorithm for our problem-statement.

- **True Positive (TP):** A true positive is an outcome where the model correctly predicts the positive class.
- **False Negative (FN):** A false negative is an outcome where the model incorrectly predicts the negative class.
- **True Negative (TN):** A true negative is an outcome where the model correctly predicts the negative class.

- **False Positive (FN):** A false positive is an outcome where the model incorrectly predicts the positive class.

4.5.1 False Alarm Rate

False Alarm rate is calculated as the number of all incorrect predictions divided by the number of true positives. It is given by the relation:

$$False\ Alarm\ Rate = \frac{FP + FN}{TP} \tag{1}$$

4.5.2 Accuracy

The accuracy is the proportion of the total number of predictions that were correct. It is given by the relation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

4.6 Machine Learning Algorithms

As we had well-labeled training data we have used a few supervised ML algorithms for classification.

4.6.1 Decision Tree Classifier

Decision tree is a Supervised learning technique that can be used for classification. It is a tree-structured classifier, where internal nodes represent the features of the dataset, branches represent the decision rules and each leaf node represents the outcome.

4.6.2 Random Forest Classifier

As the name implies, a random forest is made up of a huge number of individual decision trees that work together as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

4.6.3 K-Nearest Neighbor Algorithm

KNN is a non-parametric and lazy learning algorithm. K is the number of nearest neighbors, which is a core deciding factor. It assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

4.6.4 Logistic Regression

It is a classification technique borrowed by machine learning from the field of statistics. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as "1". It becomes a classification technique with the addition of a decision threshold.

4.6.5 Stochastic Gradient Descent Classifier

Stochastic gradient descent is an optimization algorithm often used in machine learning applications to find the model parameters that correspond to the best fit between predicted and actual outputs. It's an inexact but powerful technique widely used in machine learning applications. Combined with back-propagation, it's dominant in neural network training applications.

4.6.6 AdaBoost Classifier

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

4.6.7 Gradient Boosting Classifier

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are mostly used when doing gradient boosting.

4.6.8 Histogram-Based Gradient Boosting Classifier

Histogram-based gradient boosting is a technique for training faster decision trees used in the gradient boosting ensemble. This estimator is much faster than GradientBoostingClassifier for big datasets.

4.6.9 Multilayer Perceptron Classifier

Multilayer Perceptron is a Neural Network that learns the relationship between linear and non-linear data. It has input and output layers, and one or more hidden layers with many neurons stacked together. And while in the Perceptron the neuron must have an activation function that imposes a threshold, like ReLU or sigmoid, neurons in a Multilayer Perceptron can use any arbitrary activation function.

4.6.10 Gaussian Naive Bias

Naive Bayes is a basic but effective probabilistic classification model in machine learning that draws influence from Bayes Theorem. Gaussian naive bias is an extension to that idea when dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution.

5 Experimental Results

We have performed 7 fold cross-validation to determine the best fit model for our problem statement. The few algorithms with which we were able to achieve good enough average cross-validation accuracy and low false alarm rates are :

- Decision Tree Classifier
- Random Forest Classifier
- Gradient Boosting Classifier
- Histogram based gradient Boosting Classifier
- K-Nearest Neighbour

Classification Model	Cross validation accuracy	False Alarm Rate
Random Forest Classifier	98.3	0.0321
AdaBoost Classifier	96.0	0.0791
MLP Classifier	91.8	0.1747
Gradient Boosting Classifier	97.9	0.0397
Logistic Regression	68.7	18.0774
SGD	36.1	22.3831
Decision Tree	98.2	0.0329
KNN Classifier	98.1	0.0350
Histogram Based Gradient Boosting Classifier	98.2	0.0334
Gaussian Naive Bias	53.4	0.8844

Figure 3: Average 7 Cross-validation results for different ML Algorithms

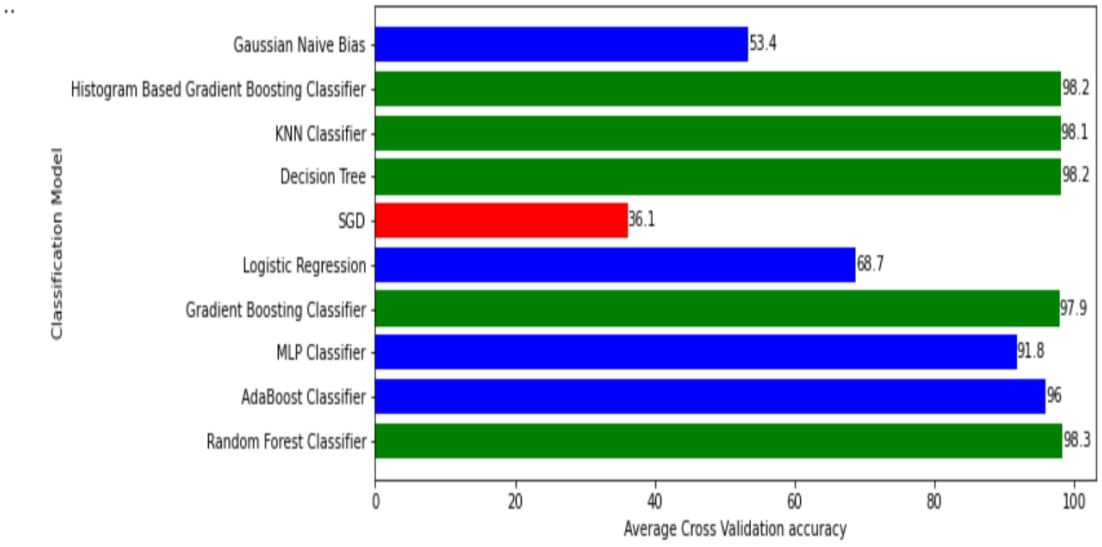

Figure 4: Average 7 fold Cross-validation accuracy for different ML Algorithms on NSL-KDD

After finding the best ML model, we used the same for making predictions on our simulated attacks. We performed DoS using TCP syn-flood and collected the logs using snort. We used the processed logs to make predictions using Random Forest Classifier.

Type of Attack	Total Number of connections	Number of connections classified as Attack
DoS	5746	5714
Non-Attack	11	0

6 Conclusion

Thus, we trained Machine learning model on NSL-KDD dataset for network intrusion detection to improve upon the existing Snort NIDS. We performed real time DoS attack using TCP Syn-Flood from Kali Linux machine to Ubuntu machine where snort was installed, and get packet logs of the attack. We performed pre-processing on these logs and then used ML model to predict whether an attack occurred. Our ML model was successfully able to identify attacks and normal traffic.

ML model is giving good accuracy but there is scope for fine tuning ML model to give better accuracy, also there is scope to detect other attacks which were not covered in NSL-KDD dataset. Further, on an ambitious leap there's a scope to integrate ML model with existing SNORT NIDS open source project to make it more robust.

References

[1] "Search unb." [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html

[2] "Download vmware workstation player," Apr 2022. [Online]. Available: https://www.vmware.com/in/products/workstation-player/workstation-player-evaluation.html

[3] "Download ubuntu desktop: Download." [Online]. Available: https://ubuntu.com/download/desktop

[4] "Kali linux," Jan 2022. [Online]. Available: https://www.kali.org/get-kali/

[5] "Penetration testing software, pen testing security." [Online]. Available: https://www.metasploit.com/

[6] "Snort 3 is available!" [Online]. Available: https://snort.org/