Machine Learning Engineer Nanodegree

Capstone Project

Cody Durden
September 5, 2018

I. Definition

Project Overview

Intrusion Detection Systems (IDS) are a type of software that monitors for malicious activity or deviations from set policy. Typically, IDS's are hosted on a company's network and monitor traffic. This serves as a good basis for protecting against unwanted intruders in a company's network. But as all things evolve so does the way these nets of protection should protect. This is where this project takes off. The outline to is to see if machine learning can aid in the current detection of malicious traffic

There are generally two different ways an IDS protects the network; signature and anomaly detection. Signature detection uses known signatures in network traffic to identify malicious traffic. Anomaly detection searches for deviations in normal traffic to try and identify potential malicious traffic. As is usually the case both used in tangent can serve as more beneficial. But what if we add the power of machine learning to aid in these decisions. This is what I want to accomplish in this project.

Problem Statement

The problem with current IDS systems is that they are slow learners. There are 2 main types of IDS

- Signature Based Detection
- Anomaly Based Detection

Signature Based Detection is reliant on detecting predetermined signatures for detection. Essentially the signatures to look out for are added to the IDS and if these are seen in network traffic an alert is triggered and then investigation can ensue.

Anomaly Based Detection is more real time in nature but still suffers from slow learning. This class requires a baseline to check for these anomalies. So the flaws could be in an ill manufactured baseline or lots of false positives in the implementation of a new Anomaly Based Detection system.

The solution to this slow learning our IDS's suffer from is the use of state of the art machine learning algorithms. By applying machine learning to the problems the IDS can take real time data, compare against pretrained data and make a determination on the current status of the network traffic sampled.

Metrics

To determine the added value of using machine learning on network traffic I will use accuracy, precision, recall, and false alarm rate (False Positives) as performance metrics using a confusion matrix. From this matrix precision will be used to determine how often true attacks are detected.

Precision = TP / TP + FP

This is important to prevent the IDS from blowing to many whistles on traffic that is benign in nature.

Recall is another important metric I will be addressing.

Recall = TP / TP + FN

The importance of detecting malicious activity is at the heart of what an IDS is designed to do, and any traffic that goes thru and into the network unflagged is potential for disaster on the network.

Accuracy is important in telling us our true value; meaning we are measuring what we aim to measure

TP + TN / TP + TN + FP + FN

II. Analysis

Data Exploration

The data as a whole was very clean. This was because a tool called the CICFlowMeter was used. This tool takes incoming pcap traffic and generates a csv file with more than 80 features.

The following were the 80 features used in analysis:

Destination Port
Flow Duration
Total Fwd Packets
Total Backward Packets
Total Length of Fwd Packets
Total Length of Bwd Packets
Fwd Packet Length Max
Fwd Packet Length Min
Fwd Packet Length Mean
Fwd Packet Length Std
Bwd Packet Length Max
Bwd Packet Length Min
Bwd Packet Length Mean
Bwd Packet Length Std
Flow Bytes/s
Flow Packets/s
Flow IAT Mean
Flow IAT Std
Flow IAT Max
Flow IAT Min
Fwd IAT Total
Fwd IAT Mean
Fwd IAT Std
Fwd IAT Max
Fwd IAT Min
Bwd IAT Total
Bwd IAT Mean
Bwd IAT Std
Bwd IAT Max
Bwd IAT Min
Fwd PSH Flags

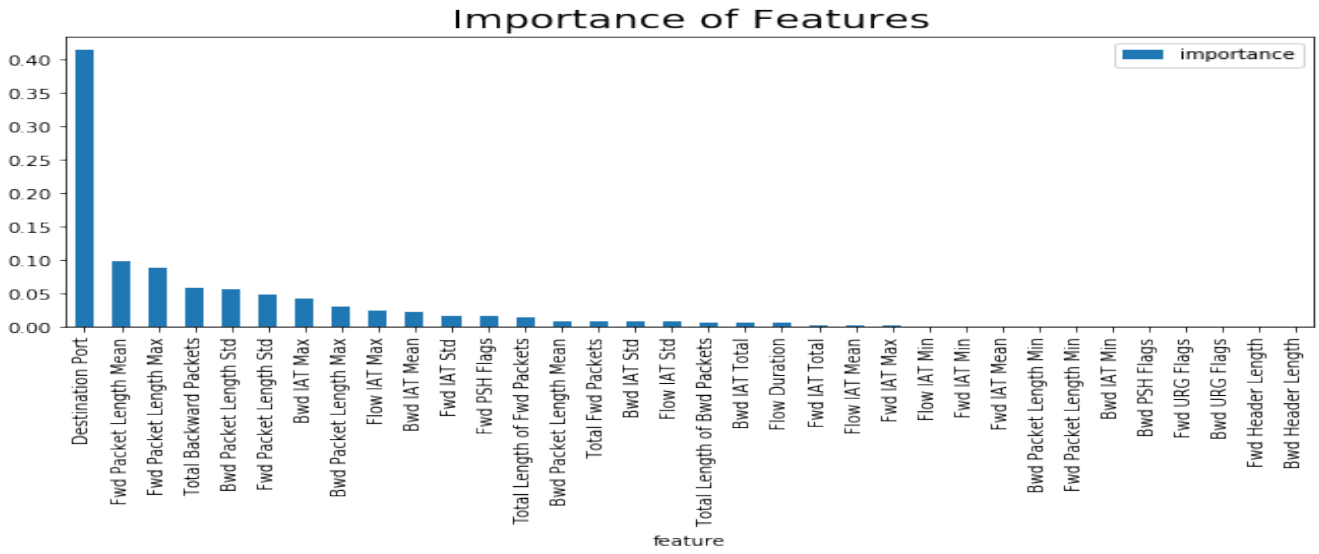
Bwd PSH Flags
Fwd URG Flags
Bwd URG Flags
Fwd Header Length
Bwd Header Length
Fwd Packets/s
Bwd Packets/s
Min Packet Length
Max Packet Length
Packet Length Mean
Packet Length Std
Packet Length Variance
FIN Flag Count
SYN Flag Count
RST Flag Count
PSH Flag Count
ACK Flag Count
URG Flag Count
CWE Flag Count
ECE Flag Count
Down/Up Ratio
Average Packet Size
Avg Fwd Segment Size
Avg Bwd Segment Size
Fwd Header Length.1
Fwd Avg Bytes/Bulk
Fwd Avg Packets/Bulk
Fwd Avg Bulk Rate
Bwd Avg Bytes/Bulk
Bwd Avg Packets/Bulk
Bwd Avg Bulk Rate
Subflow Fwd Packets
Subflow Fwd Bytes

Subflow Bwd Packets
Subflow Bwd Bytes
Init_Win_bytes_forward
Init_Win_bytes_backward
act_data_pkt_fwd
min_seg_size_forward
Active Mean
Active Std
Active Max
Active Min
Idle Mean
Idle Std
Idle Max
Idle Min
Label

In this dataset outliers were not sought after. The reason being that outliers could be the sole reason one array would be different from another.

Exploratory Visualization

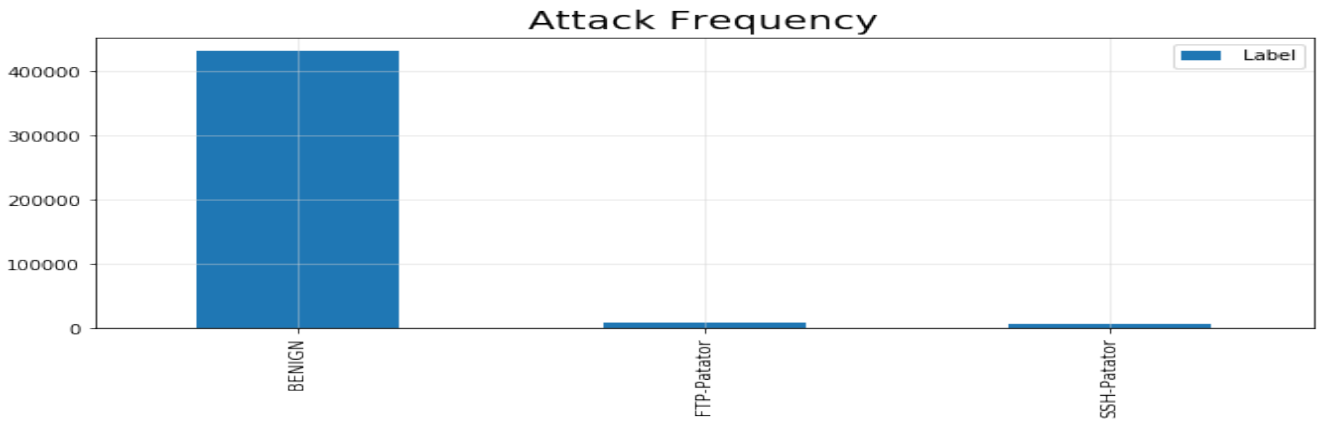
The below shows the ranking of features based on importance. These values were obtained using a Random Forrest Classifier.



Below we see the distribution of labels for a dataset

Label	
BENIGN	432074
FTP-Patator	7938
SSH-Patator	5897

Below is a visual of the dataset makeup



Algorithms and Techniques

A series of supervised machine learning algorithms were used to improve the identification of malicious traffic.

- KNeighborsClassifier
- Decision Tree Classifier
- BernoulliNB
- LogisticRegression
- AdaBoostClassifier

These models were trained on the datasets provided by the University of New Brunswick (Canadian Institute for Cybersecurity). By using the labeled dataset, I was able to train the algorithms and gather the metrics needed for assessing the model. During repeated trials the default were tweaked to seek improvement.

Fortunately, most of the data that comes in as network traffic can be for the most part easily transformed into numerical data. But consequently, the range of such numerical data was quite large. To overcome this I transformed the dataset using a Standard Scaler. By doing this the values were held within a range.

Feature selection was also an important step before performing any training. A Random Forest Classifier was used to extract the most import features. A score was given to these features and the result being that the top 10 features were used as input to the training algorithms. Principal Component Analysis (PCA) was also used to reduce the dimensionality of this dataset. Being the dataset included 80 features; reducing this selection aided in the speed of the training algorithm. Both methods were used in determining the best choice. PCA was ultimately used in training.

Benchmark

For this dataset I could not find a benchmark to compare against. This being so I can use my own training results as a benchmark. In exploration for finding a benchmark I was attempting to find an IDS for installation (Bro or Snort). Neither was

cooperating during install so I have not been able to get this as the baseline that I would like.

III. Methodology

Data Preprocessing

Exploration

To get the data in shape for processing I first had to clean it up for input into my algorithms. The dataset was wholistically in good shape. The task I have at hand was to find the best features to use. During the exploratory phase of finding these features I used a Random Forrest Classifier.

Firstly the data had to be broken up into features and labels. This was accomplished by adding the 'Label' column into it own data structure and dropping 'Label' from the features. To get the data in shape to feed into the Random Forrest Classifier I used a MixMaxScaler. This gave me a resulting range of 0-1. I also needed to quantify the labels of the dataset. To do this I used one hot encoding. After encoding the labels I turned this set into a numpy array using the MinMaxScaler. These features and labels were then fed into the Random Forest Classifier.

Machine Learning

Getting the data into shape for a machine learning algorithm as above the data needed to be broken up into features and labels. The 'Label column was added as a separate data structure and 'Label' was dropped from the features. The data was then split into training and testing sets using sklearn's train_test_split. Next the training set was transformed using a Standard Scaler followed by using a PCA transformation. Lastly these were fed into an algorithm of choice.

Implementation

The ultimate goal was to pass this dataset thru machine learning algorithms and obtain a better result than the baseline IDS. To do this the above preprocessing steps were taken in order to gather the top 10 features that would be used in

training. Following preprocessing of the dataset I was now ready to feed the data into the algorithms.

The following algorithms were chosen for training

- KNeighborsClassifier
- Decision Tree Classifier
- BernoulliNB
- LogisticRegression
- AdaBoostClassifier

The decision to train multiple algorithms was made because the amount of processing time was minimal compared to the overarching goal to retain the optimal model.

The dataset did not need a lot of massaging to get into a stable state for processing. The initial conversion from .pcap was already performed and the resulting .csv file was clean.

Refinement

The initial model used a Random Forest Classifier to choose the best features for the model. This seemed to work well during initial training. But I kept thinking that I should choose another method for feature selection. Following this thought process I chose to use Principal Component Analysis (PCA) for feature selection. To get the data in shape for using PCA a transformation using a StandardScaler was used. Afterwards the training data was transformed using pca.transform.

When applying PCA to the data I used .95 for the number of components parameter. This allowed the PCA function to choose the minimum number of principal components where 95% of the overall variance is retained.

pca = PCA(.95)

The resulting number of components was 21 for the Tuesday-WorkingHours.pcap_ISCX.csv dataset.

Improvements were initially made to the Decision Tree Classifier. This algorithm showed to have higher overall accuracy and precision for tested datasets. By applying PCA and training on the Decision Tree Classifier algorithm I was able to get a score of 0.9994. Running sklearn classification reported 1’s across the board (precision, recall, f1-score).

IV. Results

Model Evaluation and Validation

The final model chosen was to use PCA followed by a Decision Tree Classifier. The initial feature set included 69 distinct features. PCA was used with n_component variable of 0.95. This allowed the PCA function to choose the minimum number of principal components where 95% of the overall variance is retained.

The Decision Tree Classifier was then run with the pca transformed training input. The classifier was given both a criterion='entropy' and random_state=0 to check for any improvements or inferior results. These variables seems to neither add nor detract from the results.

The final model seemed very stable in its prediction abilities. No matter the dataset given the model performed very very close to perfect prediction abilities. A total of 7 datasets were given to the model and all had a classification report of perfect precision, recall, and f1-score.

```
Classification report:
              precision    recall  f1-score   support

      BENIGN       1.00      1.00      1.00    345663
FTP-Patator        1.00      1.00      1.00      6364
SSH-Patator        1.00      1.00      1.00      4700

avg / total        1.00      1.00      1.00    356727
```

Justification

As I did not find a benchmark to compare my results against I cannot come to a satisfactory answer as to whether improvements were significant. This method for classifying the data did hold well. The Classifier performed superbly on all datasets I had to throw at it. Overall I believe this would be an improvement to a current IDS setup.

V. Conclusion

Free-Form Visualization

This visualization illustrates the numerous features included in this dataset. Selection of the most important features is key to prediction viability. The order depicted here is ordered by importance after running thru Random Forrest Classifier.

	Feature	*Importance*
0	Destination Port	0.344
9	Fwd Packet Length Std	0.155
6	Fwd Packet Length Max	0.097
4	Total Length of Fwd Packets	0.058
3	Total Backward Packets	0.049
8	Fwd Packet Length Mean	0.044
19	Fwd IAT Mean	0.041
12	Bwd Packet Length Mean	0.031
13	Bwd Packet Length Std	0.026
28	Fwd PSH Flags	0.025
22	Fwd IAT Min	0.022
26	Bwd IAT Max	0.017
5	Total Length of Bwd Packets	0.013
14	Flow IAT Mean	0.010
24	Bwd IAT Mean	0.009
18	Fwd IAT Total	0.009
25	Bwd IAT Std	0.009
23	Bwd IAT Total	0.008
10	Bwd Packet Length Max	0.007
16	Flow IAT Max	0.007
2	Total Fwd Packets	0.007
21	Fwd IAT Max	0.004
27	Bwd IAT Min	0.003
1	Flow Duration	0.003
17	Flow IAT Min	0.001
20	Fwd IAT Std	0.001
15	Flow IAT Std	0.001
11	Bwd Packet Length Min	0.000
7	Fwd Packet Length Min	0.000
29	Bwd PSH Flags	0.000
30	Fwd URG Flags	0.000
31	Bwd URG Flags	0.000
32	Fwd Header Length	0.000
33	Bwd Header Length	0.000

Reflection

The problem with current day IDS is that the threat advances faster than the detection. And this is just the natural order of things; people create threats and then people develop systems to mitigate these threats. The whole system is predicated on detecting previously known or thought about threats. So what makes one IDS better than the other is the ability to learn faster. This is why I think machine learning implemented into IDS is crucial in keeping up with the fast paced threats.

To summarize the wholistic nature of the solution I propose we should follow the below process

1) Collect incoming PCAP traffic
2) Convert traffic to csv (for machine learning)
3) Choose features based on relevance (from past experience and currently streaming network traffic)
4) Send csv file with relevant features into supervised machine learning algorithm
5) Also send csv file with relevant features into unsupervised machine learning algorithm to pick up untrained labels
6) Using both supervised and unsupervised machine learning algorithms will create a real time trainable IDS

I believe the above is possible in a well-maintained security setting. The maintainer of the IDS would need to be familiar with Machine Learning to get the most out of the output.

Improvement

Improvements to this design would be to add additional datasets for testing or to put this design on the network in step with an IDS and try to create a working model of the overall design.

The benchmark currently is an IDS with no supplemented machine learning. In the future creating a benchmark with machine learning would be a better starting point to seek accelerated improvement.

For improvement in design I would suggest having a background program that continuously analyzes and updates labels including both machine learning and deep learning. This in turn is fed back into the streaming selection of input from the network for determination of traffic nature.