Mastery Extension: Malicious Network Traffic Detection with Neural Networks

by

C.J. DuHamel

Mastery Extension Project

CSC 321: Intro to Computer Security

Professor Bret Hartman

March 2025

1 Introduction

1.1 Introduction

Identifying malicious network traffic is paramount to ensuring a secure system, as no security is impenetrable. Implementing instant and accurate identification of unexpected network traffic can reduce the extent of zero-day attacks by large margins, as exploits can be identified instantly. Introducing a system like this into enterprise systems has the potential to identify and classify the type of attack, prompting immediate and potentially automated action, shutting down affected resources before they have the chance to affect critical resources, or at least before the extent of the damage becomes unrecoverable. In this paper, I will detail the process of designing, training, and testing a machine learning model that can identify malicious network traffic based on packet details and metadata, providing a machine the ability to analyze network traffic and predict the security risk that a given network packet presents.

1.2 Motivation

Network security can be difficult to monitor with traditional filtering and firewalls, as new attacks can be orchestrated that these filters cannot detect. Introducing machine learning allows these filtering mechanisms to learn more about the nature of network attacks and apply that to attacks that, although would bypass traditional filters, exhibit nuances that malicious traffic has. This provides some of the advantages of human interaction and auditing, but in real time with large volumes of network data.

2 Datasets

Multiple datasets were considered for this task but only two were chosen for exploration: The CIC IDS-2017 Dataset and the UNSW-NB15 Dataset. One was selected for the purposes of training the model, which is detailed below.

2.1 CIC IDS-2017 Dataset

The Canadian Institute of Cyber Security Intrusion Detection Evaluation Dataset (Sharafaldin et al., 2018) is a research focused dataset designed to provide developers of intrusion detection and intrusion prevention systems. It was developed by the Canadian Institute of Cybersecurity for research purposes, built based on data collected from a model enterprise system. This means that a network was set up to act like an enterprise system, including traffic modeling real network traffic to best provide a dataset that is similar to real world scenarios. It consists of several different CSV files categorized into the days and time intervals in which they were collected (e.g. Thursday Morning). Each day contains a different subcategory of network based attacks, including Denial of Service (DoS) attacks and Web Attacks. Each file consists of logged network traffic in the form of packet and flow information, detailed by 78 input features, along with a label for each entry as either "BENIGN" or one of the attack categories.

2.2 UNSW-NB15 Dataset

The University of New South Whales created the UNSW-NB15 Dataset (Moustafa et al., 2018) is also a research focused dataset, which utilizes a hybrid dataset construction approach, unlike the CIC IDS-2017 dataset, which is real network traffic but in an artificial system. This hybrid approach utilizes real network traffic as the normal "benign" traffic, and then from this synthesizes attack traffic to add to the dataset. This dataset is split into four files but does not contain separation of attack types. Each file contains collected network traffic with packet and flow details for each row, which is detailed with 49 different input features and a label: "BENIGN" or one of the attack types included in the dataset.

2.3 Comparison

Both datasets provide a great platform for building a machine learning model off. Both are improvements of previous datasets that had several issues such as very large class imbalances, improper collection techniques, and most importantly, out-of-date attack examples, resulting in analysis lacking modern contexts. However, I chose the CIC IDS-2017 dataset for this task for a few reasons. The data collection method from a synthetic enterprise system I believe is best for interpretability, both throughout the model training and during result analysis. Second, the dataset is presented in the form of days and time periods, each with their own set of attack types, allowing the model to focus on identifying primary features of a certain type of attack, which is what the benchmarks for the model are based off. Lastly, it offered less input features, which may be acceptable, however additional input features allows for further analysis and more robust models after proper feature selection and preprocessing.

3 Exploratory Data Analysis and Preprocessing

3.1 Initial Data Analysis

The data was downloaded from the Canadian Institute for Cybersecurity and loaded into a Pandas data frame. Based on a previous experiment, in an attempt to create a more well-rounded and robust model, I combined two of the time intervals ("Wednesday" and "Thursday Morning") to include both DoS Attacks and Web Attacks in the same set of training and testing data.

3.2 Class Imbalances

Charting the distribution of the target variable (Fig 3.1) results in clearly imbalanced target classes. I attempt to address this in part by combining attack types into a 2 distinct target classes: “DoS” and “Web Attack”. In this process, I remove the “Heartbleed” attack types, as they make up only 11 rows of the dataset and doesn’t fit properly into either category. There is still a major class imbalance issue, so I also run Synthetic Minority Oversampling Technique (SMOTE) on the data, synthetically upsampling the minority classes to be equal with the majority class. The resulting class distribution is visible in Figure 3.2 (Fig. 3.2)

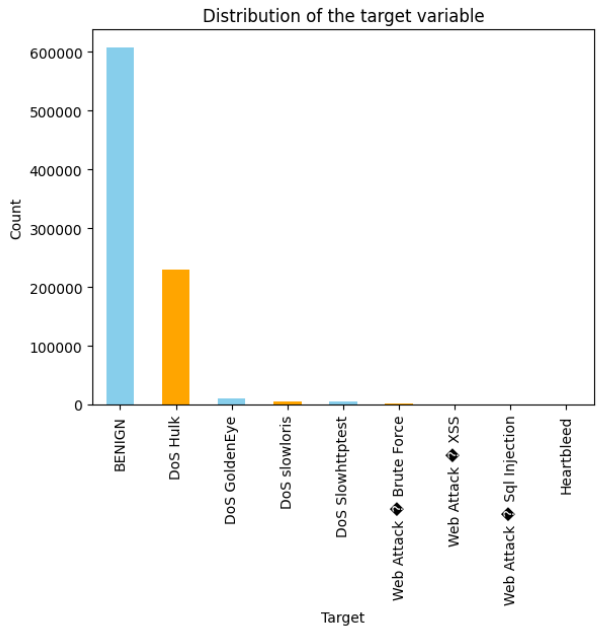


Fig 3.1

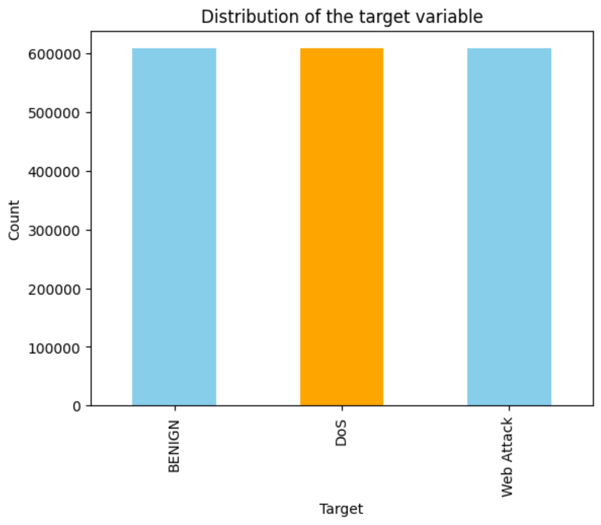


Fig 3.2

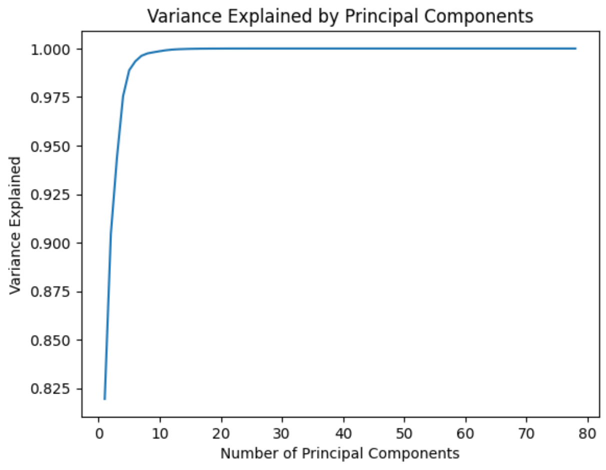
3.3 Feature Selection and Reduction

To begin the feature selection process, I run an ANOVA test on the data to extract and use SciKit Learn’s SelectKBest to get the 10 most important features in the dataset. It returned:

'Bwd Packet Length Max', ' Bwd Packet Length Mean',' Bwd Packet Length Std', ' Packet Length Std', ' PSH Flag Count',' Avg Bwd Segment Size', 'Init_Win_bytes_forward',' Init_Win_bytes_backward', 'Idle Mean', ' Idle Max'

Due to limited experiences in networking and network security, based on the results I decided to utilize Principal Component Analysis (PCA) to do feature reduction rather than feature selection. By taking Linear Combinations of the input features and identifying the necessary principal components, I can capture the most important data from the dataset in a significantly reduced number of features without further feature analysis.

I perform PCA on the data and retrieve the analysis data. It is clear based on the graph and chart (Fig 3.3 and 3.4) of the number of principal components vs. the variance explained that the number of principal components necessary to include in the model can reduce the features to less than 10 from 78. To find the exact number, I calculate the cumulative sum of the variance explained array, and determine the number of components that are required to explain 99 percent of the variance, which resulted in 5 principle components. I select the top 5, and create a new table of those 5 components and the output labels.



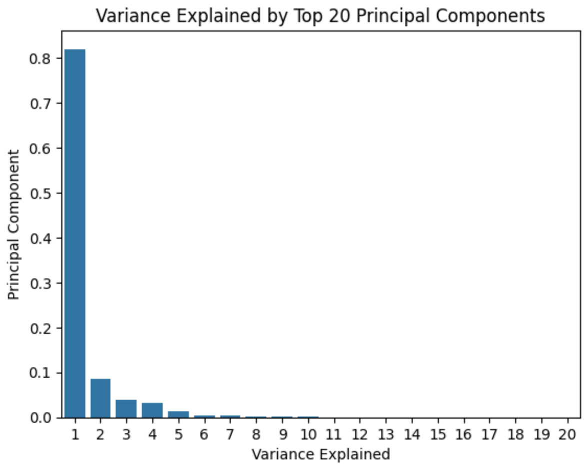


Fig 3.3

Fig 3.4

3.4 Data Preparation

To prepare the data for the model, I Label Encode the output labels to create an integer based output classification. I then scale the data using the SciKit Learn Standard Scaler since all data is numerical and to reduce the total numerical range.

Following this, I randomly split the data into training and testing sets into an 80/20 split. The 20 percent partition I split in half even further into a testing set and a validation set. This is to ensure there is always a generalization set after model fine tuning with the test set, ensuring that the generalization set is data that the model has not been specifically educated on or tuned for. This, theoretically, will provide an example of “real world data” that is introduced to the model when it is ready to be used. Following the split, the training set contained 1,823,202 rows and both the test and validation sets contained 182,321 and 182,320 rows respectively.

4 Model Architecture and Training

4.1 Random Forest Model: Architecture

To create a baseline model for testing and comparison, I utilized SciKit Learn’s Random Forest Model with its default parameters. I chose this model as a baseline because it is simple and interpretable. Using the default parameters should provide a baseline metric I can use to improve on with other models.

4.2 Random Forest Model: Results

The random forest model performed very well, with an accuracy score of 99.9% on the training set. On the test set, it had an accuracy score of 99.6% and an F1 score of 99.6, which shows that the model is not overfitting the training data. Testing on the validation set yields similar results, with an accuracy of 99.58%, also showing that the model is not overfitting the training data. However, this does not prove that the model is not overfitting this specific kind of traffic.

4.3 Neural Network: Architecture

To accomplish this classification task, I am using a Multi-Layer Perceptron (MLP) Model built using the nn.Sequential class offered by PyTorch. It contains 3 layers: 64 neurons, 128 neurons, and finally another 64 neuron layer. It was chosen in this manner to provide some variety to the hidden layers without making the network too complicated given the task and the performance of the Random Forest Model. It also employs powers of two to define the hidden layer sizes, as libraries are often optimized for values that are powers of 2. Each hidden layer is accompanied by a ReLU Activation layer to prepare input for the next layer. The output layer has 3 neurons to represent the three classes. Each output neuron produces a number representing the relative probability of being that particular class.

The model employs Cross Entropy Loss as its primary loss metric, which is very common amongst classification tasks. This is utilized by the ADAM Optimizer utilizing a learning rate of 0.001. To best accommodate GPU acceleration with Nvidia CUDA, set up locally, I employ a batch size of 1024, which was decided on based on training times with various batch sizes.

4.4 Neural Network: Results

The model was trained three separate times: with 10 epochs, with 20 epochs, and with 30 epochs. 10 epochs yielded the lowest accuracy at 90.6%. Training 20 epochs yielded a higher accuracy at 92.2%, while 30 increased slightly to 92.4%. The training accuracies for each iteration are within 0.1% of their testing accuracies, implying that the model is not overfitting on the training set.

Running the best model (30 epochs) on the validation set results in an accuracy of 92.5 percent, reinforcing the idea that the model is not overfitting. However, this again does not mean that the model is not overfitting on the specific kind of traffic that is being used to train, test, and validate the model.

5 Conclusion

5.1 Discussion

Unfortunately, due to the scope of the project, I am unable to gather further results from different model parameters or test my models on generalization sets from other synthesized or real enterprise networks. However, some parts of the model performance and architecture can be improved based on the results of this project alone.

The Random Forest Model outperformed the Neural Network by a large margin, which was unexpected due to the complexity of the data. However, it is possible that the data is simply not as complex as it seems. Rather the dataset may be nearly linearly separable, giving simpler models the advantage, including Linear Regression models. I attempted to overcome this challenged when I was faced with this problem early in the project, using only the DoS dataset. To create a more realistic and more robust model, I added the Web Attack dataset, however, this did not make much of a difference. It is possible that there are distinct characteristic features of both attacks present in the dataset (e.g. oddly large packet sizes for DoS attacks) for all examples of the given attack, allowing the network to easily distinguish between attacks and benign traffic.

Despite the Random Forest model outperforming the Neural Network, I believe that the Neural Network is a more robust model with a higher chance of success on a generalization/validation set outside of the current dataset (i.e. real enterprise network traffic). If this dataset is truly near linearly separable, it is more likely that the neural network was able to capture more defining nuances of each attack than the Random Forest Model. Modern DoS and Web Attacks employ strategies to bypass traditional network firewalls and filters, which could be difficult for the Random Forest model to handle, whereas the Neural Network captures some of the semantic features and relationships between attack types.

Based on Analysis of the principle components returned by PCA, it seems that it is easy to identify features that define a given attack, therefore, it is further evidence that the attacks present in this dataset are too obvious to provide quality training data for an Intrusion Detection Model.

5.2 Future Research

The first step following this research is to collect a generalization set of real enterprise network traffic and test my models on it, providing a interpretable and concrete accuracy of model performance. This will determine if the dataset is appropriate for the task of identifying malicious traffic. More exploratory data analysis may also utilized to identify linear separability before any further training on the data is done.

The neural network architecture could also be improved, as variations of the current architecture have not been thoroughly tested. It is possible that expanding the network can provide the model the ability to capture more smaller relationships between attacks, allowing it to identify more subtle forms.

Lastly, I would like to explore the sequential nature of the network traffic. Employing an RNN/sequential based strategy such as using GRU or Transformer architecture. Network attacks may take place over a period of time, including multiple network entry points and repeated suspicious activity in the network. Analyzing and utilizing past network traffic in the classification of new traffic in real time can provide information to the model that my current strategy would never be able to accomplish.

5.3 Conclusion

Machine Learning provides opportunities for computers to accomplish similar tasks as human auditors, allowing systems to “self-audit” themselves without the need for human interaction. This in my opinion is not a replacement for humans, rather it allows computers to do what humans can do but around the clock, while humans still provide the much-needed auditing and analysis at the same regular intervals. Identifying attacks based on characteristics rather than comparing them to existing resources of known attacks can enable systems to alert or even prevent 0-days much sooner than before, as the system can identify characteristics of an attack by its training, rather than by reference.

6 Citations

Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”, 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018.

Moustafa, Nour, and Jill Slay. "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset." Information Security Journal: A Global Perspective (2016): 1-14.

Moustafa, Nour, et al. "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks." IEEE Transactions on Big Data (2017).

Moustafa, Nour, et al. "Big data analytics for intrusion detection system: statistical decision-making using finite dirichlet mixture models." Data Analytics and Decision Support for Cybersecurity. Springer, Cham, 2017. 127-156.

Sarhan, Mohanad, Siamak Layeghy, Nour Moustafa, and Marius Portmann. NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. In Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings (p. 117). Springer Nature.