Analysis and detection of malicious events in Network traffic

Tim Coleman colemat@colorado.edu
Anusha Basavaraja anba9017@colorado.edu
Christopher Bisbee chbi4702@colorado.edu

ABSTRACT

As IoT and network related devices keep increasing at a rapid rate, the attack surface also increases allowing malicious actors to gain access to systems and cause damage and/or extort money for personal gain.

This paper aims to understand a commonly known type of malware that is associated with botnet type of cybercrime. This type of malware normally operates as a command and control on the network of the infected devices where the infected device connects to the central control agent to receive instructions such as DDoS. This paper looks closely at the corresponding network traffic and uses data mining and machine learning techniques to try and predict whether a network flow is likely malicious or benign.

The data set CTU-13 was selected to use which contains 13 different netflow scenarios where each scenario represented a specific botnet network capture.

5 key features were selected from CTU-13, normalized and used to train 3 different machine learning classification models: Logistic Regression, Naive Bayesian and Random Forest. Each model was trained and tested using the same parameters and evaluated in the same manner using ROC (AUC) and AP plots as evaluation metrics. The Random Forest model scored the best with a result of AUC=0.84 and AP=0.63

INTRODUCTION

Cybersecurity is a large and growing area of research that spans a broad range of IT disciplines from securing and defending large corporate networks, computers, IoT devices, securing sensitive data, encryption best practices, access controls, user responsibility, etc., all of which are important to consider when implementing cybersecurity policy for an entity, such as corporation. However, even when following a comprehensive cybersecurity policy such as the NIST cybersecurity framework (https://www.nist.gov/cyberframework), mistakes happen and malicious events can occur which can be very costly in terms of money, time, and reputation. According to Douglas Thomas [3], it was estimated that US losses in 2016 ranged from $167.9 to $770.0 billion and have been growing ever since.

This paper investigates a specific type of cybercrime that is closely associated with malware that participates in distributed types of attacks such as DDoS. These types of malware usually involve a command and control connection across the network in which they are connected to and are often referred to as botnets which consists of a collection of infected hosts that are controlled by a central agent.

Using techniques from data mining and machine learning, the goal is to try and analyze the network traffic and understand to see if there are network features that can be extracted that correlate to known malicious patterns such as botnets. And then answer the question, can these features be used to train a machine learning model to try and predict whether a network flow is malicious or benign?

Malware can be thought of as any software that can be installed and can run on a host that is designed to inflict some form of damage such as disabling, deleting, encrypting, stealing, crypto mining, controlling, etc.. Malware comes in different forms and usually targeted for a specific purpose. Common types of malware include viruses, worms, trojan horses, and blended malware

that combine several forms into one.

This paper focuses on a type of malware associated with a botnet cybercrime, which essentially refers to a collection of network devices that can be controlled from a command control center and/or peer-to-peer and carry out specific actions such as engaging in a DDoS attack. The malware installed on the device itself can stem from a virus, worm or trojan horse, but they all perform a similar function in that it allows the attacker to control the device and turn these devices into what is commonly known as "zombies."

The goal of the paper is to gain a better understanding of the type of network connections commonly associated with botnet malware, and determine if there are features that can be extracted from the NetfFlows to train machine learning classification models to predict if a netflow is malicious or benign.

RELATED WORK

The paper *Cyber Attack Detection thanks to Machine Learning Algorithms* [1] mentions using NetFlow data that does not contain sensitive information and is widely used by network operators. With proper analysis methods, NetFlow data can be an abundant source of information for anomaly detection. One major drawback to NetFlow has to do with the volume of data generated, particularly for links with high load. This could have an effect on the accuracy of the results and the amount of processing to be done. A lot of relevant work has already been performed to evaluate and process NetFlow data to address this. One method is by looking at NetFlow sampling (Wagner, Francois, Engel, et al. 2011). It further explains about incorporating Machine Learning techniques in malware detection. Machine learning capabilities are utilized by many applications to solve network and security-related issues. It can help project traffic trends and spot anomalies in network behaviour. Large providers have embraced machine learning and incorporated tools and cloud-based intelligence systems to identify malicious and legitimate content, isolate infected hosts, and provide an overall view of the network's health. Stevanovic and Pedersen 2014 developed a flow-based botnet detection system using supervised machine learning. Santana, Suthaharan, and Mohanty 2018 explored a couple of Machine Learning models to characterize their capabilities, performance and limitations for botnet attacks. Machine learning has also been seen as a solution for evaluating NetFlows or IP-related data, where the main issue would be selecting parameters that could achieve high quality of results. There are multiple ML approaches mentioned in this paper such as Random Forest, Gradient-Boosted Trees, Classification voting based on Decision trees and NaiveBayes, Logistic regression.

Random Forest (RF) is a supervised machine learning algorithm that involves the use of multiple decision trees in order to perform classification and regression tasks (Ho 1995). The Random Forest algorithm is considered to be an ensemble machine learning algorithm as it involves the concept of majority voting of multiple trees. *Gradient boosting* is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It combines the elements of a loss function, weak learner, and an additive model. *Support Vector Machine* (SVM) is a supervised learning model used for regression and classification analysis. It is highly preferred for its high accuracy with less computation power and complexity. SVM is also used in computer security, where they are used for intrusion detection. *Logistic regression* is a supervised learning model that is used as a method for binary classification. The term itself is borrowed from Statistics. At the core of the method, it uses logistic functions, a sigmoid curve that is useful for a range of fields including neural networks. Logistic regression models the probability for classification problems with two possible

outcomes. Logistic regression can be used to identify network traffic as malicious or not.

The paper also mentions using the CTU-13 dataset, one of the datasets we are also using for our project. The paper provides more ideas on *Feature extraction* and *Feature selection*, the major steps in our project. According to the paper, a popular idea is to summarize data inside a time window. This is justified by the fact that botnets "*tend to have a temporal locality behavior*" (Garcia et al. 2014). Moreover, it enabled them to reduce the amount of data and to deliver a botnet detector which gives live results after each time window (in exchange for input data details). The extracted features try to represent the NetFlow communications inside a time window. To do so, the time window data is gathered by source addresses. There are 2 features extracted from each categorical NetFlow characteristics: the number of unique occurrences in the subgroup and the normalized subgroup entropy. As for the numerical NetFlow characteristics, 5 features are extracted: sum, mean, standard deviation, maximum and median. These extracted features enable training of different ML models. They have used *Filter* and *Wrapper* methods for Feature selection. The *Filter method* involves filtering features in order to select the best subset of features for training the model. The best features are selected based on scores of Pearson Correlation statistical test since the pre-processed data involves continuous data. The *wrapper method* involves testing different subsets of features on a training model in an iterative fashion in order to determine the best features based on the inferences made by the training model. During each iteration, features are either added to or removed from the subset which will then be used to train the model. This is repeated until no more improvement is observed.

The paper *Machine Learning-Based Early Detection of IoT Botnets Using Network-Edge Traffic* [2] presents a lightweight IoT botnet detection solution, EDIMA, which is designed to be deployed at the edge gateway installed in home networks and targets early detection of botnets prior to the launch of an attack. EDIMA includes a novel two-stage Machine Learning (ML)- based detector developed specifically for IoT bot detection at the edge gateway. The ML-based bot detector first employs ML algorithms for aggregate traffic classification and subsequently Autocorrelation Function (ACF)-based tests to detect individual bots. The EDIMA architecture also comprises a malware traffic database, a policy engine, a feature extractor and a traffic parser. The EDIMA is designed to have a modular architecture with following components: *Feature Extractor*, *ML-based Bot Detector*, *Traffic Parser/ Sub-sampler*, *Malware PCAP Database*, *ML Model Constructor* and *Policy Engine*. Each of these components play their roles in botnet detection. The Feature extraction ML-based bot detector and ML Model Constructor are of main points of interest to us. In a traffic session, they have extracted features from TCP packet headers only and not the payloads. This method works on unencrypted as well as encrypted traffic. Once the features are extracted, the ML model will be trained on it for the Classification task to identify whether the network traffic is *benign* or *malicious*. Once the aggregate traffic at an edge gateway has been classified as malicious, the second fine-grained stage of the ML-based bot detector attempts to detect the underlying bots by checking the ingress/egress traffic from each IoT device for the presence of bot-CnC communication patterns. The set of IoT devices is sorted according to their IP addresses. Performance evaluation results show that EDIMA achieves high bot scanning and bot-CnC traffic detection accuracies with very low false positive rates. The detection performance is also shown to be robust to an increase in the number of IoT devices connected to the edge gateway where EDIMA is deployed.

The paper *Intelligent Mirai Malware Detection for IoT Nodes* [6] develops and compares the performance of Random Forest (RF) and Artificial Neural Network (ANN) models trained using a merged dataset of benign and malicious network traffic data across seven IoT devices,

including commonly used devices such as indoor and outdoor security cameras, a Phillips Monitor, and a Samsung Webcam. The malicious traffic data contains Mirai malware which can be used to turn networked devices running Linux into bots for DDOS attacks. The features of the dataset used to train, cross-validate, and test each model are gathered from network traffic moving between source and destination hosts. This paper relies on another paper [7] for feature extraction to identify the most revealing of the 115 features in the dataset (N-BaIoT, which captures normal network traffic patterns) across five different time windows and uses Matlab to create the ANN model and Python to create the RF model. When cleaning the data, the authors note that the most difficult problem was aligning time windows across datasets during the merger to ensure that a proper comparison can be made between benign and malicious network traffic occurring concurrently. Also noteworthy was the authors' use of Microsoft Excel to import the dataset into Matlab and also "for modification of data and for initializing the features of the data." Excel is a fine tool for most purposes but is prone to introducing errors during data analysis so analysts and researchers must be especially careful to avoid or cross-check data that they work on in Excel to ensure that no errors or inconsistencies have been introduced. When comparing the RF and ANN models, the authors found that the ANN model with two hidden layers performed better than the RF model with an accuracy of 92.8%, a False Negative percentage of only 0.3%, and an F-1 score of 0.99.

DATASET

The Stratosphere Laboratory is the provider and maintainer of the dataset that for this project: https://www.stratosphereips.org/

Dataset: *CTU-13*
https://www.stratosphereips.org/datasets-ctu13

The CTU-13 data set contains 13 different scenarios of classified data that contain different network captures where each scenario represents a start and end capture time. Each scenario captures a specific type of malware and labels them accordingly as benign or malicious. All the datasets are formatted using Cisco NetFlow v5.

NetFlow is a network protocol that was designed by Cisco in the mid 1990s to collect metadata from network traffic as it enters and exits network interfaces. This type of network data is used by many IT professionals instead of using raw network captures as the data is aggregated into more meaningful data chunks and is easier to analyze. Below is a reference format of NetFlow v5, which is the format used for this data set.

NetFlow v5 format:

Bytes	Contents	Description
0-3	srcaddr	Source IP address
4-7	dstaddr	Destination IP address
8-11	nexthop	IP address of next hop router
12-13	input	SNMP index of input interface
14-15	output	SNMP index of output interface
16-19	dPkts	Packets in the flow
20-23	dOctets	Total number of Layer 3 bytes in the packets of the flow
24-27	First	SysUptime at start of flow
28-31	Last	SysUptime at the time the last packet of the flow was received
32-33	srcport	TCP/UDP source port number or equivalent
34-35	dstport	TCP/UDP destination port number or equivalent
36	pad1	Unused (zero) bytes
37	tcp_flags	Cumulative OR of TCP flags
38	prot	IP protocol type (for example, TCP = 6; UDP = 17)
39	tos	IP type of service (ToS)
40-41	src_as	Autonomous system number of the source, either origin or peer
42-43	dst_as	Autonomous system number of the destination, either origin or peer

44	src_mask	Source address prefix mask bits
45	dst_mask	Destination address prefix mask bits
46-47	pad2	Unused (zero) bytes

https://www.cisco.com/c/en/us/td/docs/net_mgmt/NetFlow_collection_engine/3-6/user/guide/format.html

MAIN TECHNIQUES APPLIED

The main techniques applied can be broken out into 4 categories:

1 Data Preprocessing
2 Data Exploration and Visualization
3 Feature selection and extraction
4 Classification and machine learning

1) Data Preprocessing

The CTU-13 data set is formatted using NetFlow version 5 and also includes an additional column "Label" that contains information about the type of flow and if it was malicious or benign. However, as this label has high cardinality, another column was created called "botnet" that contained a boolean value to represent a binary classification to easily distinguish between malicious or benign traffic where botnet=1 and not botnet "benign" =0.

In addition to adding another column, data cleanup was needed for destination and source ports and IP addresses as invalid data existed in some columns such as NaN, hexadecimal values, and/or corrupted data that had alpha characters when numeric was expected causing errors when manipulating and casting to integers or floats. This was remedied by using built-in pandas regex to replace occurrences in columns with suitable replacement values. As the number of offending rows was minimal in each scenario (< 1% of the data), the offending rows were dropped vs replaced.

2) Data Exploration and Visualization

Out of 13 scenarios in CTU-13 dataset, each of them consists of millions of records (netflow data objects) and the percentage of botnet presence in each scenario is listed below:

Scenarios	***Percentage of Botnet***
Scenario - 1	1.451
Scenario - 2	1.158
Scenario - 3	0.569
Scenario - 4	0.231
Scenario - 5	0.694
Scenario - 6	0.828
Scenario - 7	0.056
Scenario - 8	0.207
Scenario - 9	8.862
Scenario - 10	8.120
Scenario - 11	7.612
Scenario - 12	0.666
Scenario - 13	2.078

Table: The presence of percentage of Botnet traffic in each scenario in CTU-13 Dataset

Based on the table above, scenarios 9, 10 and 11 contain the highest proportion of Botnet traffic in their netflow data.

Out of 17 attributes belonging to different attribute types, we have selected a few of them for plotting different graphs for better understanding and visualizing the CTU-13 dataset. A few of the attributes of the CTU-13 dataset were selected to highlight the characteristics of the data set.

Dur : Duration of the communication in seconds.

Where min = 0sec, max = 3600sec

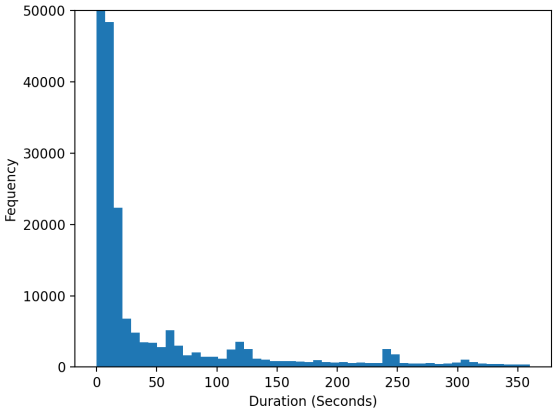


Figure: Duration of Communications

Proto: Transport Protocol Used. There are 15 protocols used in total.

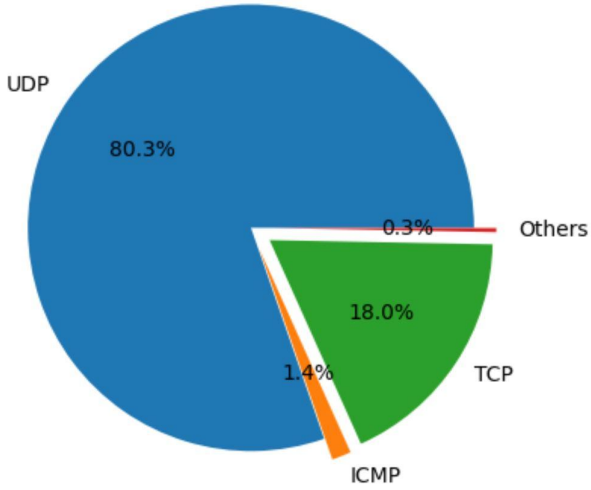


Figure: Distribution of Transport Protocols

Dport: Destination port. There are a total of 73,786 ports. It is a range.

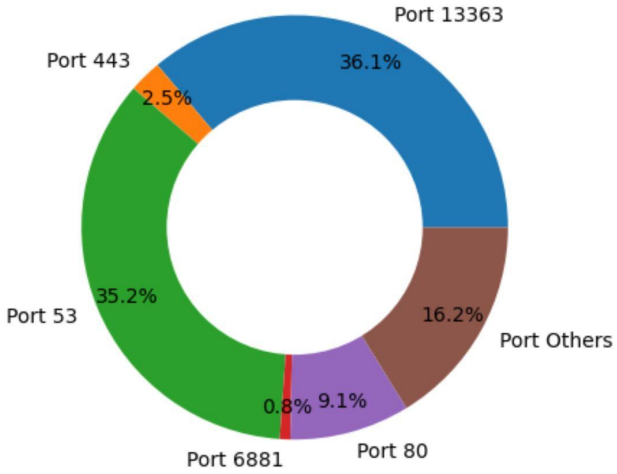


Figure: Distribution of Destination Ports

TotPkts: Total number of transaction packets.

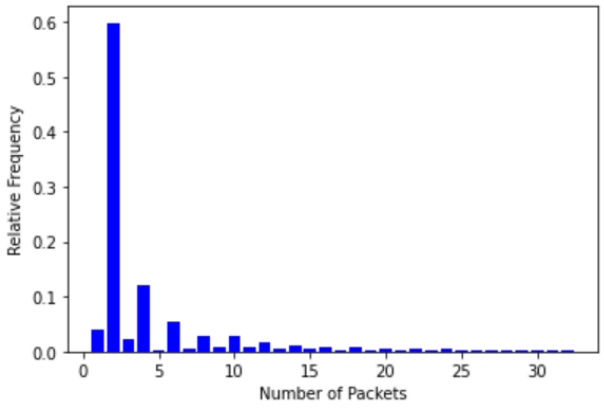


Figure: Most frequent number of packets

Other attributes in the dataset are:
StartTime: Initiation of packet transfer
SrcAddr: IP address of the source
Sport: Source port
Dir: Direction of the flow
DstAddr: IP address of the destination
State: Transaction state
sTos: Source TOS byte value
dTos: Destination TOS byte value
TotBytes: Total number of transaction Bytes
SrcBytes: Total number of transaction Bytes from the Source

Label: Label made of "flow=" followed by a short description of netflow type
Botnet: Indicates the presence or absence of botnet traffic in the netflow (absence = 0 and present = 1). We have considered 'Botnet' as our class label for training our model.

3) Feature Selection and Extraction

While the data visualizations plots were helpful in determining frequency, value counts and distributions, it wasn't as helpful to determine how important a feature is relative to containing a botnet flow. Understanding how features may be correlated to a botnet is critical to selecting the best features to use as input to machine learning classifiers. This analysis used a couple of methods to determine and rank features that had higher correlations to containing botnet traffic and extracted these features with the highest scores.

For this analysis we used Python based scipy packages containing correlation functions such as *pointbseirialr* and *chi-squared*, which are used to correlate a continuous variable vs a categorical variable and categorical vs categorical, respectively. These tools were used for this project to determine if a feature is independent or correlated to botnet traffic, and provided a good indicator if a feature is worth considering as input for building a machine learning model.

For example, determining the correlation between the total number of packets vs whether it's a botnet yields the following results:

Total Bytes vs Botnet:
PointbiserialrResult(correlation=-0.0011626170736417556, pvalue=0.0930014441334596)

Using the results above, we can determine that with a pvalue > 0.05, it is determined to *not* be statistically significant so Total Bytes was not selected for feature selection.

Another example is comparing the netflow duration to the botnet, with the idea that botnets are typically associated with quick connections. Using the *PointbiserialrResult* function, we can determine the following:

Duration vs Botnet:
PointbiserialrResult(correlation=-0.10172736084672689, pvalue=0.0)

As this has a p value < 0.05, this is significant and thus was selected a feature to extract and use for the machine learning classifiers.

Below are the correlation results when comparing the continuous variables vs the binary botnet categorical values for CTU scenario 9 data set.

CTU Scenario 9		
Total flows	2087508	
Botnet flows	184987	
Percentage Botnet	8.86%	
	Botnet	
PointbiserialrResult	p value	correlation
Duration (time sec)	*0.000*	*-0.102*
Total Bytes	*0.093*	*-0.001*
Total Packets	*0.004*	*-0.002*
Source Bytes	*0.084*	*-0.001*

From the result above, Duration and Total Packets were selected as features to be selected.

Another correlation technique is the chi-squared test which can be used to measure correlation between two categorical values. For example, it can be used to correlate between destination port and botnet as destination port is a category and contains a range of ports from 1-65535. If a particular port is often associated with a botnet, there should be an expected correlation between these features. Using pandas we can construct a contingency table known as crosstable

and then use this as input to the scipy.stats.chi2_contingency method to compute chi-squared and p value.

To compute the correlation between destination port vs botnet, we arrive at the following.

Full range of ports 1-65535
(chi-square=772594.3099241449, pvalue=0.0, df=41695)

From the example above, the number of degrees of freedom equals 41695, and with an expected critical value based on significance level of .05, we can compute the critical value, which yields 42171.124. Since our chi-squared value (*772594.30)* is higher, we can reject the null hypothesis and this may not be a good candidate for feature selection based on this outcome, or further data reduction and binning the data may be required to yield a most positive result, which is what we ended up doing for Destination Port and reduced the high cardinality down to the top 10 ports.

In the end, the final selected features to use for testing and training the machine learning classification were as follows:

Feature	**Type**	**Extraction Process**
Duration	Continuous float	Used as-is
Protocol	Categorical varchar	categories maped to int {'Proto': {'tcp': 1, 'udp': 2, 'icmp': 3, 'other:' 0}}
Destination Port	Categorical Int (1-65535)	categories maped to int {'Dport': {'dns': 1, 'nonres1': 2, 'http': 3, 'https': 4, 'ssh': 5, 'bittorrent1': 6, 'rdp': 7, 'dnap': 8, 'ntp': 9, 'snmp': 10 }}
Total Packets	Continuous int	Used as-is
Destination IP Address	Categorical IPv4 2^32	Replaced with .value_counts()
Botnet	Boolean	Botnet training and testing label

4) Classification and Machine Learning

Once the features were extracted, Logistic Regression, Random Forest and Naive Bayesian classifiers were trained and tested using the same technique for each scenario and a combined scenario.

Logistic Regression is a simple statistical Machine Learning model used for binary classification. The netflow data is represented in the form of a set of features after the feature extraction process. Then we score each netflow data based on the features using a *sigmoid function* also called *logistic function*.

Random Forest is based on decision trees and the model builds many trees all of which act independent of each other but are used in the final consensus voting to determine the outcome.

Naive Bayesian is based on Bayes' theorem and assumes each feature is independent of each other and uses a probability calculation based on statistics to score each attribute to determine the likelihood of an outcome for a given tuple.

Each model was trained and tested using even and odd rows, where the training set consists of even rows and the test set consists of odd rows. Since the botnet traffic in each scenario is not randomly distributed throughout the data and can be concentrated within select time periods, using an even / odd approach provides a better chance that the model can learn from botnet flows.

Each of the 13 CTU scenarios was run for each model as well as a combined run of all scenarios 1-13. For each scenario and model the Receiver Operating Characteristic (ROC) and Precision-Recall plots and scores are shown in results to compare the model performances.

The Receiver Operating Characteristic (ROC) ROC curve compares the relationship between True Positive Rate (x-axis) and False Positive Rate (y-axis). The purpose of the plot is to help visualize all the classification thresholds and determine the optimal threshold that maximizes the detection of true positives while minimizing the number of false positives. Generally speaking, lower thresholds results in classifying more items as positive but also increases the number of false positives, and conversely, increasing the threshold reduces the number of false positives but can lead to more false negatives. While different applications will have unique requirements for detecting botnet traffic, it is of the authors' opinion that it makes sense to have a higher threshold to reduce the number of false positives as this can overwhelm many cybersecurity experts attempting to detect malicious traffic, especially if a model is running in near real time.

The Area Under the Curve (AUC) measures the total area under the ROC curve. A higher AUC score generally implies a better performing model. For example, a perfect classifier would score a 1.0 while the worst score would score a 0.0. While the AUC is a good evaluation metric, it doesn't tell the full picture especially when data is imbalanced such as the CTU data set. As most of these scenarios have less than 5% of botnet traffic, the data is very imbalanced. To combat this, another evaluation plot that is commonly used compares Precision vs Recall where Precision is plotted along the y-axis and Recall along the x-axis. Precision is the ratio of correctly predicted positives to the total number of predicted positives. Recall is the ratio of the predicted positives over all the positives. The Precision Recall curve illustrates the tradeoffs at different thresholds but shows how well the model is at predicting botnets flows. A higher precision threshold will likely result in a lower recall which reduces the number of false positives but increases the number of false negatives. Just as illustrated by the ROC, the threshold is dependent on the type of data and objective, but for detection of botnets, generally you would favor a higher recall to avoid false positives.

KEY RESULTS

Here we compare the performance of Random Forest, Naive Bayesian and Linear Regression Models. Each model is trained and tested using even and odd rows, where the training set consists of even rows and the test set consists of odd rows.

The plots below contain the ROC and Recall - Precision plots for each CTU scenario and all scenarios (1-13) ran together using the even / odd split approach for Naive Bayesian, Random Forest and Logistic Regression as well as summary table for all scores.

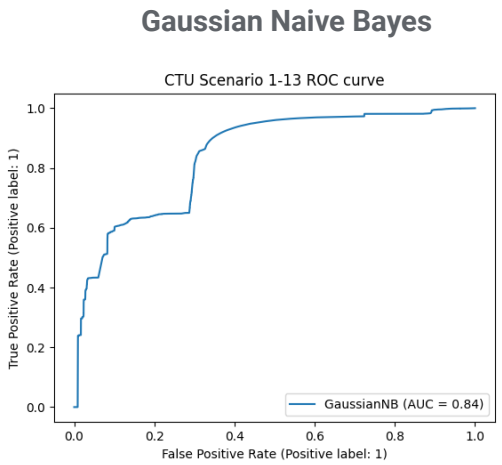


Figure : *ROC - Gaussian Naive Bayes for combined scenarios 1-13 even(train) / odd(test)*

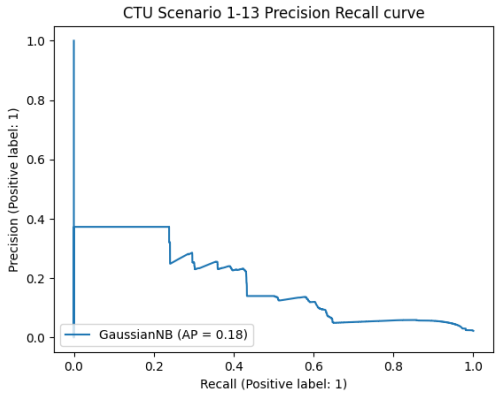


Figure : *Average Precision - Gaussian Naive Bayes for combined scenarios 1-13 even(train) / odd(test)*

Random Forest

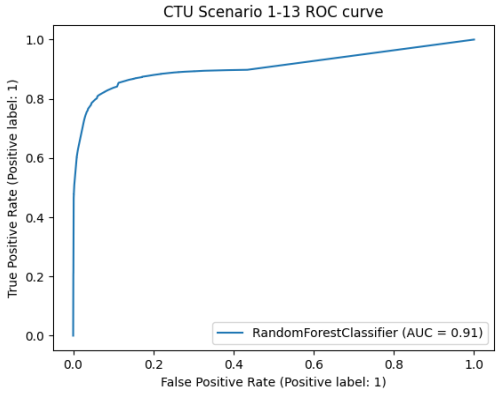


Figure : *ROC - Random Forest for combined scenarios 1-13 even(train) / odd(test)*

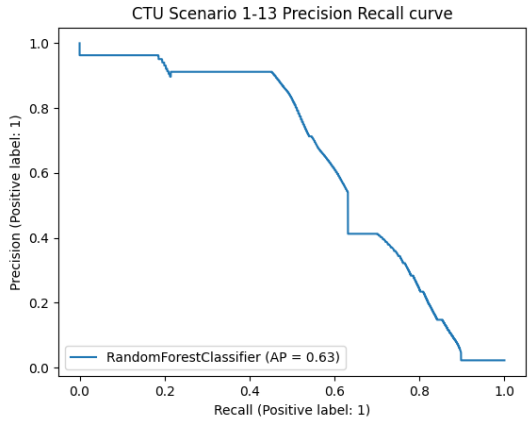


Figure : *Average Precision - Random Forest for combined scenarios 1-13 even(train) / odd(test)*

Logistic Regression

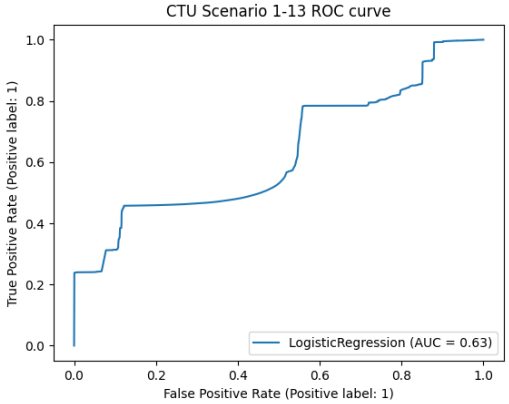


Figure : *ROC - Linear Regression for combined scenarios 1-13 even(train) / odd(test)*

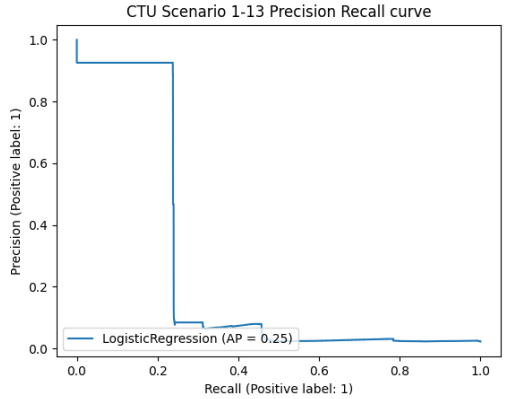


Figure : *Average Precision - Linear Regression for combined scenarios 1-13 even(train) / odd(test)*

Area Under Curve (AUC) Score			
Scenario	**Gaussian Naive Bayes**	**Random Forest**	**Linear Regression**
CTU-1	0.83	0.94	0.72
CTU-2	0.89	0.94	0.94
CTU-3	1.00	1.00	0.96
CTU-4	0.96	0.96	0.91
CTU-5	0.86	0.92	0.87
CTU-6	0.99	0.99	0.97
CTU-7	0.74	0.70	0.67

CTU-8	0.93	0.97	0.91
CTU-9	0.75	0.93	0.56
CTU-10	1.00	1.00	1.00
CTU-11	1.00	1.00	1.00
CTU-12	0.74	0.75	0.70
CTU-13	0.87	0.94	0.81
CTU All 1-13	0.79	0.84	0.50

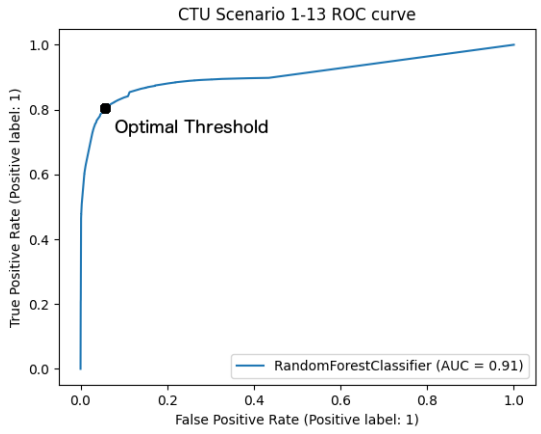
Table: *Area Under Curve (AUC) for scenarios 1-13 and combined 1-13.*

Average Precision (AP) Score			
Scenario	**Gaussian Naive Bayes**	**Random Forest**	**Linear Regression**
CTU-1	0.58	0.87	0.57
CTU-2	0.06	0.81	0.14
CTU-3	0.97	1.00	0.91
CTU-4	0.12	0.48	0.05
CTU-5	0.04	0.73	0.04
CTU-6	0.49	0.87	0.50
CTU-7	0.00	0.24	0.00
CTU-8	0.04	0.44	0.14
CTU-9	0.27	0.69	0.28
CTU-10	0.99	1.00	0.99
CTU-11	0.99	1.00	1.00
CTU-12	0.02	0.80	0.02
CTU-13	0.17	0.83	0.50
CTU All 1-13	0.18	0.63	0.25

Table: *Average Precision (AP) for scenarios 1-13 and combined 1-13.*

Overall, the Random Forest performed the best when using ROC and Precision-Recall as evaluation criteria. It is worth noting that while the Random Forest model did perform the best, in selected scenarios such as 3,10,11 all the three models - Random Forest, Naive Bayseian and Logistic Regression scored well for AUC and AP scores, and interestingly, these scenarios also contain the same virus Rbot. So while Random Forest did have the best overall score, using a Naive Bayseian classifier could make sense to use for certain types of viruses such as Rbot. However, if the goal is to apply a model that can generalize to multiple types of scenarios (viruses), from this analysis using the Random Forest would lend itself to the best results.

While Random Forest is the best scoring model, to use this model a user would need to select the best threshold to avoid too many false positives while having the largest amount of true positives detected. To determine that threshold, you can use the ROC curve and identify the best ratio which is usually when the curve begins to bend and flatten around on the x-axis represented by the block dot shown below.



We generated a *HeatMap* for the results from the Random Forest model using the readings from the confusion *matrix*. Here, we used the combined dataset including all the scenarios from 1 to 13 combined in producing the Heatmap.

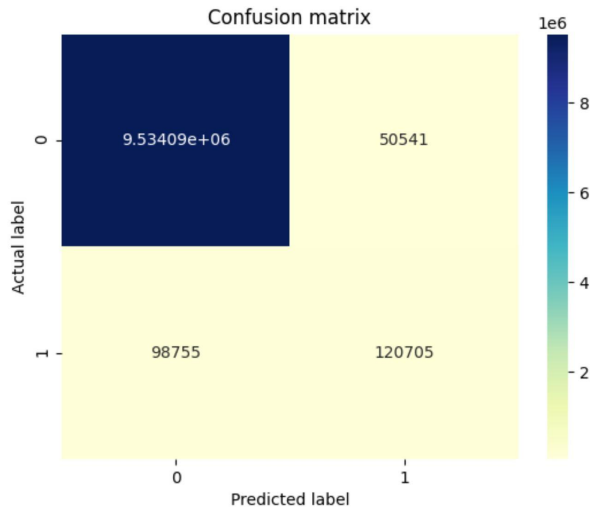


Figure: *Heatmap using the Confusion Matrix from Random Forest Model (1 to 13 scenarios combined)*

Interestingly, and worth highlighting different metrics to understand model performance, is that accuracy, although a common tool to gauge model performance, is not always the best approach especially for imbalanced data sets such as CTU-13. Even though the models Gaussian Naive Bayes and Logistic Regression perform really well in specific individual scenarios, we cannot generalize these models to the entire dataset.

APPLICATIONS AND FUTURE WORK

A lot was learned from working on this project. By using a real data set with data mining tools such as Python Pandas, Numpy and Sklearn to preprocess, explore, extract and apply machine learning techniques, all of these tools taught us invaluable insights that can be applied to other areas of data mining research. In particular, working with pandas was a great experience as some of our members previously worked with numpy, and while that Python library is powerful, pandas expands on this library and makes it even easier to use. This will be a favorite tool many of us will choose for data processing, exploration, and visualization.

The knowledge gained on understanding what features should be extracted as input to machine learning classifiers was also invaluable. As we learned, not all features are worth considering, and applying correlation techniques to determine and quantify features of high value is essential in building a good feature set to use with a machine learning classifier. Another thing is how categorical data is translated into machine readable format using id mapping or using one-hot encoding techniques is an area we didn't fully understand until working through this project.

Finally, understanding the ROC and AP curves was essential to understand how different thresholding yields different confusion matrices and hence different results for presion, recall, sensitivity and specificity, and how visualizing these as represented by ROC and AP curves was essential in helping to determine which model performed the best.

For future work it would be nice to run more model training and testing scenarios. However, as this data set contained millions of records, and the processing time (especially combined scenarios) was very lengthy using local computing resources such as a laptop, running on cloud computing resources would have been desirable to iterate faster on different test and train strategies such as k-fold cross validation. Also, iterating on different features as input would be very interesting to see how that would change the model results.

REFERENCES

Our github repo:
https://github.com/anushab97/Data-Mining-CSCI-5502-Project-Group-04.git

Video link to our presentation:
https://drive.google.com/file/d/1Z0qSlBBUOn6Tv8CTZsrZKdnh6DjVUf2-/view?usp=sharing

Research Papers:

[1] Cyber Attack Detection thanks to Machine Learning Algorithms: https://arxiv.org/pdf/2001.06309.pdf

[2] Machine Learning-Based Early Detection of IoT Botnets Using Network-Edge Traffic: https://arxiv.org/pdf/2010.11453.pd

[3] D. Thomas. *Cybercrime Losses An Examination of U.S. Manufacturing and the Total Economy* NIST, 2020. https://doi.org/10.6028/NIST.AMS.100-32

[4] Sebastian Garcia, Martin Grill, Jan Stiborek and Alejandro Zunino *An empirical comparison of botnet detection methods*. Computers and Security Journal, Elsevier. 2014. Vol 45, pp 100-123. http://dx.doi.org/10.1016/j.cose.2014.05.011

[5] Sebastian Garcia, Agustin Parmisano, & Maria Jose Erquiaga. *IoT-23: A labeled dataset with malicious and benign IoT network traffic* 2020 (Version 1.0.0) [Data set]. Zenodo. http://doi.org/10.5281/zenodo.4743746

[6] Tarun Ganesh Palla & Shahab Tayeb. *Intelligent Mirai Malware Detection for IoT Nodes.* MDPI AI/ML Techniques for Intelligent IoT Systems, 2021. https://www.mdpi.com/2079-9292/10/11/1241/htm

[7] Davis, A.; Gill, S.; Wong, R.; Tayeb, S. *Feature Selection for Deep Neural Networks in Cyber Security Applications*. In Proceedings of the 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), Vancouver, BC, Canada, 21–24 April 2020; pp. 1–7.