Applied AI for Botnet Detection

Nilay Anand, Trieu Vo

College of Professional Studies

EAI 6980: Integrated Experiential Capstone

Professor Mimoza Dimodugno

Abstract

Botnet detection is concerned with detecting hostile bots in a network. The goal is to recognize botnet behaviors such as unauthorized access, data theft, and distributed denial-of-service (DDoS) assaults using machine learning approaches. Deep learning models will typically train on enormous datasets of network traffic to learn the differentiating characteristics of botnet behavior and identify it from benign traffic. The problem is correctly identifying botnets while avoiding false positives and negatives, as well as keeping up with botnet creators' evolving techniques. This project uses the CTU-13 dataset gathered by CTU University, Czech Republic. We test different machine learning models in multiple scenarios to compare performance while improving accuracy for detecting bot behavior in a network. The algorithms under consideration are Random Forest, XG Boost, Light GBM and neural networks.

Introduction

According to [12], Botnets are considered one of the biggest threats to the internet. A botnet is a set of computers infected by bots. A bot is a piece of malicious software that gets orders from a master. This appellation "bot" comes from the old chat service Internet Relay Chat (IRC), where users could develop so-called "bots" that could keep channels alive, deliver funny lines on request, etc. The first botnets were directly built as IRC bots. Once bot malware runs on a computer, it has as much access to the computer's resources as its owner. Bots can then read and write files, execute programs, intercept keystrokes, access the camera, send emails, etc. Criminals need to be able to control their bots and give them orders. For this purpose, bots report to Command-and-Control servers (C&C or CC or C2). These CC servers are the weak point of the botnet: without them the bots are just useless drones. Cyber criminals have been developing more and more sophisticated ways for the bots to receive their orders. A network of bots can take one the following structures:

- Centralized - This is the most basic and earliest type. The algorithms send reports to a central computer on a regular basis. This was originally an IRC discussion area, but other protocols have since developed. The bottom line is the same: if the central computer goes down, the bots are rendered worthless. Defenders can close the entire botnet if they gain possession of the central server. Defenders can easily locate the central server by studying a bot or the data it transmits.
- Proxies - To make the task of finding the C&C server harder, bot creators started to include proxies in their architecture. The individual bots do not contact the C&C server directly, but intermediate machines that serve as relays, or proxies. These proxies can

either be servers operated by the botnet master, or infected machines themselves. There are several advantages to this architecture:

  - Defenders need to analyze a proxy to find the C&C server.
  - Adding more proxies is easy, making the infrastructure more resilient.
  - However, there is still a single point of failure in the form of the C&C server itself.

- Peer-to-peer - The final evolution of botnet architecture is the move for peer-to-peer Bots contact other bots, and not the C&C server. Information and control commands are propagated in the network from bot to bot. To maintain control of the botnet, its master only needs to be able to contact any infected machine. This makes the takedown of the whole botnet a very difficult task.

The impact of this global cyber threat has been significant. According to industry estimates, botnets have caused over $9 billion in losses to U.S. victims and over $110 billion in losses globally. Approximately 500 million computers are infected globally each year, translating into 18 victims per second [14].

Machine learning has shown some real promise to combat this problem. In this study, we test the viability of some algorithms that have shown good performance in other areas of study, namely, Random Forest, XG Boost and Light GBM. Furthermore, we also build and compare the performance of a neural network to identify features of a bot net. To achieve this, we will be using the CTU-13 dataset which is a well labeled data of packet captures of an infected network.

Literature Review

Introduction

Botnets pose a significant threat to the security of networks, computers, and the Internet as a whole. Detecting botnets is essential to prevent the spread of malware and prevent damage to network infrastructure. Traditional methods of botnet detection involve manually analyzing network traffic and system logs, which is time-consuming and not scalable. Machine learning (ML) has emerged as a promising solution to detect botnets automatically. This literature review will explore the different approaches and techniques used for botnet detection using machine learning.

Related Work

Several researchers have explored the use of machine learning for botnet detection. In a study by (Tobiyama et al., 2016), suggested a malware process detection technique for infection detection using two stage DNNs. Their plan identifies malware by using CNN to categorize feature images. The feature picture was produced by a behavioral language model built with RNN from extracted behavioral characteristics. They analyzed the validation results obtained under various circumstances and found that the best result AUC=0.96 was obtained when the feature image size was 3030.

In another study, (Larriva-Novo et al., 2020) tested the different optimizers used in neural networks to test for improvements in the model for cybersecurity applications. The optimizers were compared for a recurrent neural network, they tested RMSprop, SGD, Adam, Adagrad, Adadelta, Adamax and Nadam. Between the different runs they did not find significant change in accuracy which was between 98% and 99% for every iteration.

In more recent work, (Ahmed et al., 2022) investigated the improvements in botnet detection with a well-designed neural network. They compared the results with models trained by SVM, Naïve Bayes, Back Propagation and Decision Tree. The SVM achieved 99.5% accuracy, decision tree achieved 95.2%, NB achieved 98.5%, and backpropagation achieved 96.1%. Compared with previous techniques, the DNN Model proposed in this paper achieved 99.6% for training and testing and 99.2% prediction accuracy of 2000 records.

Conclusion

Botnet detection is a challenging task, and machine learning has emerged as a promising solution. Various machine learning algorithms and techniques, such as SVM, RF, CNN, K-means clustering, and RNN, have been used for botnet detection. These studies have shown that machine learning-based approaches can achieve high accuracy, precision, and recall in detecting botnets. However, selecting the appropriate dataset and algorithm, as well as optimizing the model's parameters, are crucial to achieving the best results. Future research in this area should focus on developing more robust and scalable machine learning-based solutions for botnet detection.

Methodology

Dataset

The models were trained and evaluated using a dataset labeled packet captures of botnet, background, and normal traffic from the CTU-13 Dataset. The dataset of botnet traffic is captured at CTU University, Czech Republic in 2011 and provided by the Stratosphere Lab. It is a capture of real botnet traffic mixed with normal traffic and background traffic for 7 different types of bots over 13 different sessions. The data is labeled as the type of traffic is known beforehand. The complete dataset is not available publicly due to privacy concern. The subset of data publicly available is 75 Gigabytes in size and has nearly 20 million rows and 15 features. Each of these features describes a different component of the captured packet [1].

Table 1

Features in the CTU-13 dataset and their meaning.

Feature	**Meaning**
StartTime	The start time of the capture
Dur	The duration of the capture
Proto	The protocol
SrcAddr	The source address (IP address)
Sport	The source port
Dir	The direction of
DstAddr	The destination address
Dport	The destination port
State	The state

sTos	The source type of service
dTos	The destination type of service
TotPkts	Total packets: the total number of packets that have been transmitted between two network devices, including both data and control packets
TotBytes	Total bytes: the total amount of data that has been transmitted between two network devices, including both data and control packets
SrcBytes	Source bytes: the amount of data that has been transmitted from the source device to the destination device, excluding any control packets
Label	The type of the capture (background, botnet and normal)

The data also consists of the byte code of the data sent within the packets that may be used to detect suspicious activity. There are several questions that we want to find answers to:

1. How can we categorize the botnet traffic and detect infected machines?
2. What are the most essential factors influencing classification?
3. What is the Effectiveness of model against a novel botnet not seen by the model?

The dataset has contained packet captures from multiple bots, each of which have a unique way of operation, the functioning of each is as following:

- Rbot - A series of backdoor Trojans called Backdoor: Win32/Rbot gives hackers access to affected machines. The Trojan connects to a certain IRC server and enters a particular channel after infecting a computer to accept commands from attackers. The Trojan can be instructed via commands to propagate to other systems by checking for network shares with weak passwords, taking advantage of Windows flaws, and using backdoor ports made available by other families of malicious software. The Trojan can also give

attackers access to additional backdoor tasks like initiating DoS attacks and obtaining system data from affected systems [5].

- Donbot - A botnet known for sending out spam emails is called DONBOT, also known as BUZUS or BACHSOY. To propagate malicious files, it also spammed shortened URLs using instant messaging services like Yahoo Messenger and MSN. Variants of DONBOT frequently enter systems as files left behind by other malware or as files mistakenly downloaded by users while accessing dangerous websites. When employed, DONBOT can compromise the security of the infected systems by acting as a proxy server. Once it establishes a connection to its C&C server, it can also take over the systems [6].
- Fast flux - a technique used by threat actors to evade detection and hosting malicious activities such as phishing, malware distribution, and command and control servers. The technique involves rapidly changing IP addresses for a single hostname, making it difficult to track the malicious domain. Several types of Fast Flux, include single flux, double flux, and triple flux, and explains how attackers use them. It also describes how Fast Flux is used in diverse types of cyberattacks, such as phishing and banking trojans, and how it can be used to hide the location of the attacker's command and control servers. Recommended ways to detect and mitigate Fast Flux attacks are using DNS sink holing, monitoring for DNS changes, and blocking connections to Fast Flux domains [7].
- Sogou – It is classified as a trojan. These types of malwares are commonly used for establishing remote access connections, capturing keyboard input, collecting system information, performing denial-of-service (DoS) attacks, and running/terminating processes [8].

- Qvod – It is classified as a Trojan downloader capable of downloading other malicious files or an updated version of itself [9]. It is known to be a part of QVOD Player which is a free Chinese media player that uses p2p protocols like BitTorrent to provide on demand video services [10].

Data Exploration

We used Google Drive to store the dataset and used Google Colab to work with it. We first worked with a botnet named Rbot in folder 10 to see how to preprocess the data and build a baseline model to predict the botnet. We saw the first observations to gain insight into the data set records. We then examined the shape of the data, which showed us almost 1.3 million observations and 15 features. We also used the whole dataset afterwards and we found out the shape of the whole dataset is almost 20 million observations and 15 features. We also examined the structure of the data set and found that there are 6 numerical and 9 categorical features.

Finally, we checked the target variable. The target variable contains data about the type of traffic being logged. It is classified into 52 different categories based on flow direction, protocol, and traffic type (Background, Botnet, Normal). Since the goal of the project is to identify botnet traffic, the labels are named after the type of traffic. We decided to convert them to binary values to simplify the problem: 0 for background and normal traffic, and 1 for botnet traffic.

Train-Test Split

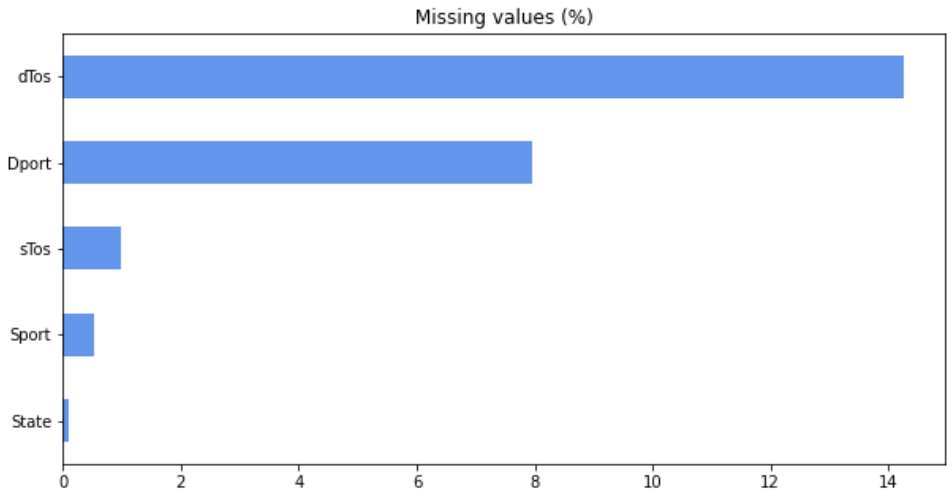
Because we want to make sure there is no data leakage when creating the model, we decided to split the dataset into training and test sets right at this step, before manipulating the data. We used stratified sampling to ensure that the training set and the test set have the same distribution on the target variable. The ratio of the training set to the test set is 8:2.

Data Cleaning

We found the missing values in some features and their percentages are as follows:

Figure 1

Percentage of missing values



We decided to replace the missing values with the most common values. We performed this step on the training and test datasets separately, so that they don't mix up values.

Data Transformation

We found out that there are 9 categorical features in the dataset. Among them, there are 6 features that we can convert them into numerical features. They are Proto, SrcAddr, DstAddr, Sport, DPort, Dir.

For the ‘SrcAddr’ and ‘DsrAddr’, they contains IP address and Mac address. We found out that most of the IP address are IPv4 addresses and just a small number of them are IPv6 addresses, which comes from the background traffic. So we decided to remove all records that have the IPv6 and Mac addresses. We also use a library called ipaddress to convert those addresses to numerical values.

Regarding ‘Sport’ and ‘Dport’, although they are numeric values, their types are object values. Some of them also contain hexadecimal values, so we created a function to convert them to numeric values.

Regarding 'Dir', because there were some errors in the flow direction, we used Regular Expression to find and modify them. We made the modification on the training set and the test set separately.

In the data set, there are 2 categorical variables with low cardinality 'Proto' and 'Dir'. We decided to convert these 2 variables numeric variables using One Hot Encoding. First we use OneHotEncoder library to fit and transform these 2 variables on the training set. We then applied that encoder to the test set to ensure that the unique variables in the training set would be present in the test set. We want to make the distributions of the training and test sets similar, so that the model will be able to accurately predict the test set.

Data Visualization

After we transformed the data, we visualized the data to find useful information for feature selection. We use Histogram and Boxplot to see the distribution of the data. We see that most data has values around zero, and some variables have a lot of outliers. From there, we decided to use the RobustScaler library to solve them. This library is very useful for dealing with outliers, and it is also great for scaling data.

Figure 2

Histogram of features

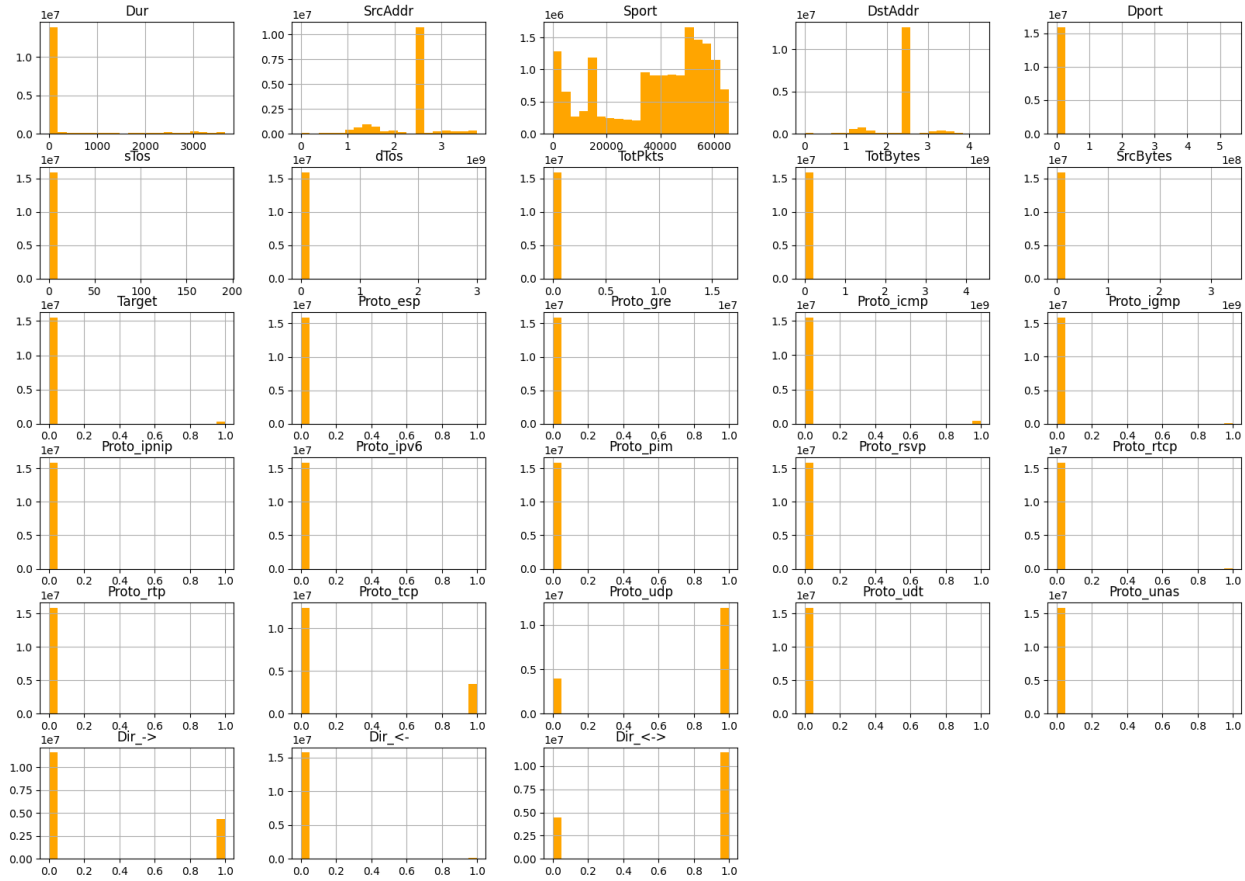
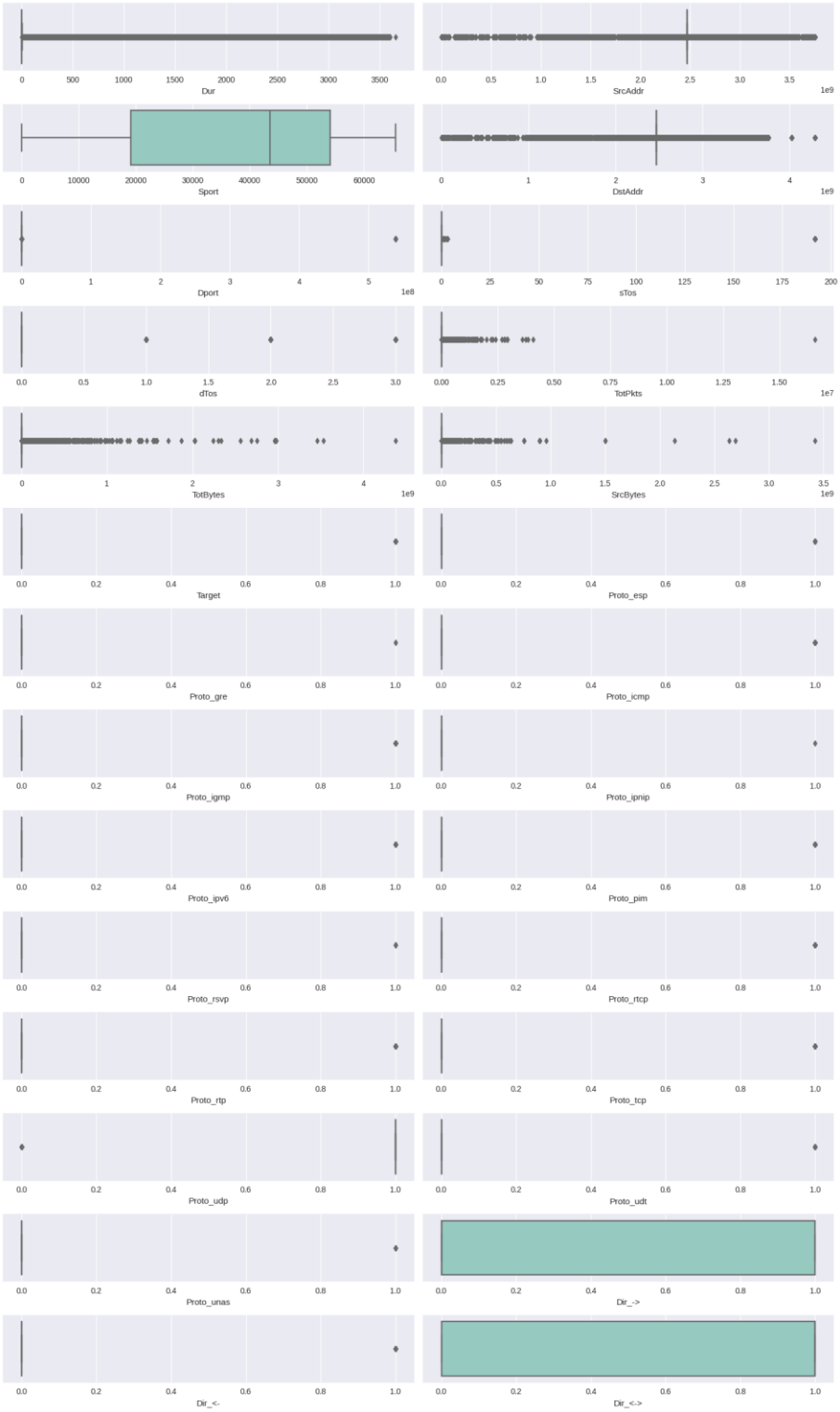


Figure 3

Boxplots of features



We also visualized some botnet properties to understand botnet behavior. Based on these visualizations, we found that most of the botnet traffic has very short duration, they usually use udp, tcp and icmp protocols. In addition, they also use two-way flow, or only one way to send information.

Figure 4

Duration of botnet traffic

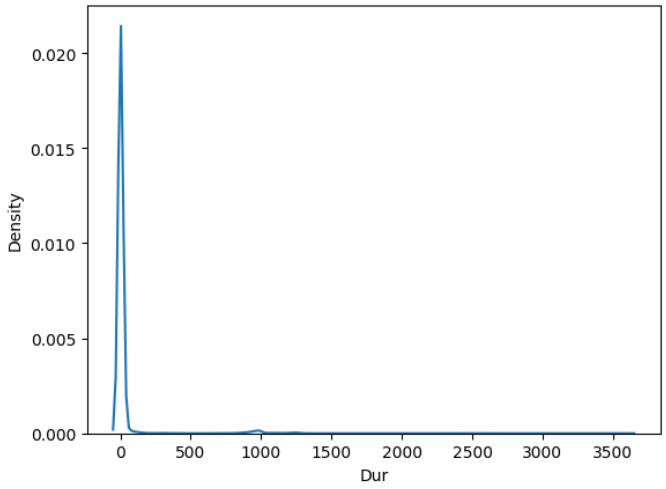


Figure 5

Most popular protocols of botnet traffic

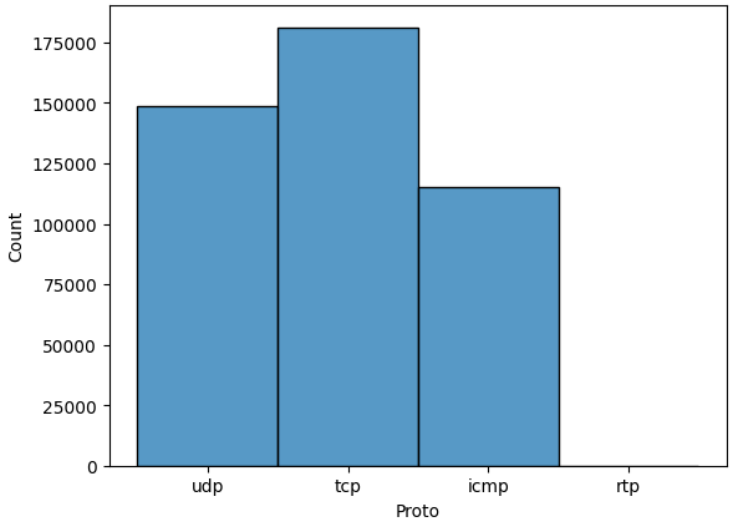
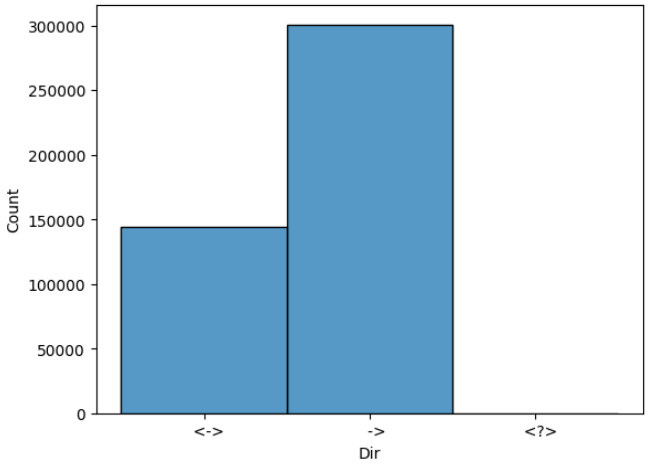


Figure 6

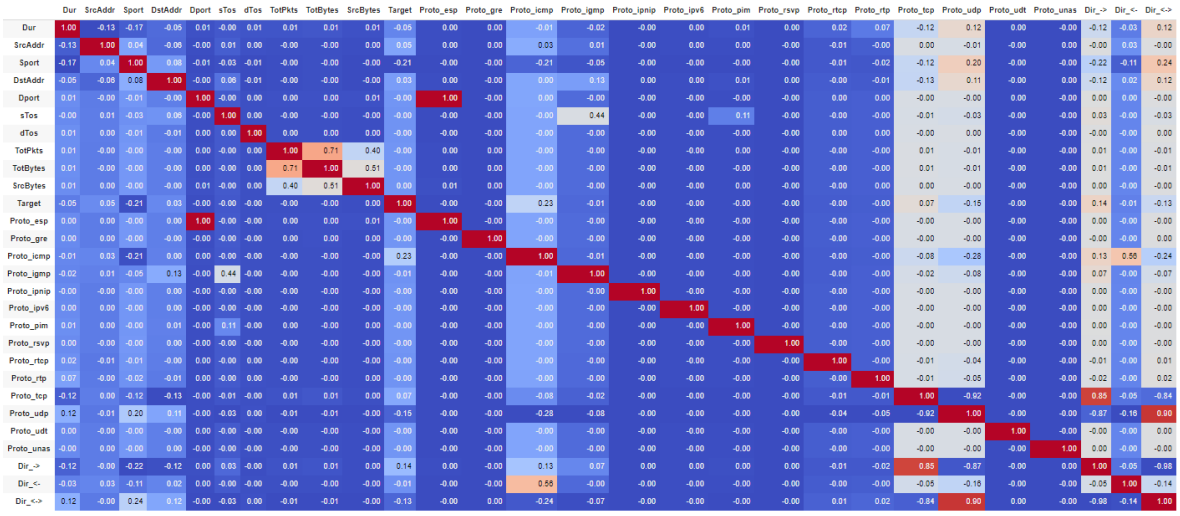
Most popular directions of botnet traffic



We also used the correlation matrix to visualize the linear relationship between the variables. As a result, we were able to remove highly correlated variables, making the learning model faster and more accurate. We also noted to keep the variables that are highly correlated with the target variable, to help the model see clearly which characteristics help to best predict the botnet.

Figure 7

Linear Correlation Matrix



Feature Selection

After careful consideration, we have decided to remove the following variables: 'Proto_esp', 'TotPkts', 'Dir_<->', 'Dir_->'. These are variables that are highly correlated with each other, so they are not necessary as inputs to the model. At this step, the data contains about 15 million observations and 24 features.

Imbalanced data

Because the data contains a large amount of background traffic and only a small amount of botnet traffic, there is a huge disparity between the amounts of the two values in the target variable. If this problem is not solved before training the model, the model will be biased towards the larger volume traffic and will predict most of the future traffic to be background traffic. So, we used the RandomUnderSampler library to adjust the count of two values in the target variable. We have randomly taken the number of two traffics so that they are equal. As such, we had 355709 observations for background traffic and 355709 observations for botnet traffic.

Feature Scaling

Finally, to make the values inside the variables not differ too far, we use RobustScaler to rescale the data. This library is also very useful for dealing with outliers, making the model work more accurately.

After completing all the preparation and data manipulation steps, we had the final training set with about 700,000 observations and 23 features. We next fed these data into machine learning models and trained them, then tested them on new botnet datasets.

Machine Learning models

We decided to use well-known machine learning models and scored well in data science competitions. Random forests, XGBoost, LightGBM, and artificial neural networks are all commonly used machine learning models in many applications. Each of these models has its own strengths and weaknesses, but in general they are well suited for this botnet problem.

The models were trained and evaluated using common metrics such as accuracy, precision, recall, and F1 scores. The models were also evaluated on a new botnet dataset that the models had never seen before to gauge how well the model performed against unknown malware.

We used 3 main strategies to build and evaluate these models:

- Train & Test on several datasets of 1 botnet (Rbot), then Test again on a brand new dataset of the same botnet (Rbot). This strategy was the first step we took to build the basic models. It saved us time and computing power, and we were able to tweak the data preprocessing steps to improve the model.
- Train & Test on 13 datasets of 7 botnets. After getting the basic models and adjusting the data preprocessing, we ran our code over the entire data and looked at the model results again. This step took quite a while, but as the data became more numerous and varied, it helped us see the capabilities of the models more accurately.
- Train & Test on 12 datasets of 6 botnets, then Test again on an unknown new botnet. This strategy helped us test our trained models against a botnet dataset it had never seen before, a whole new type of botnet. We wanted to see if our model

would work well in the real world, where countless new botnets pop up and attack users' computers.

Random Forest

Random Forest is an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random forests correct for decision trees' habit of overfitting to their training set.

We built and trained the model based on the strategies listed above. The final parameters that we got after fine tuning the model are listed below:

{'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'max_features': 'sqrt', 'max_leaf_nodes': None, 'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_jobs': None, 'oob_score': False, 'random_state': 42, 'verbose': 0, 'warm_start': False}

Figure 8

Results of the Random Forest model on 12 datasets of 6 botnets

```
TEST RESULTS:

Classification Report:
              precision    recall  f1-score   support

           0     1.0000    1.0000    1.0000   3881234
           1     0.9989    1.0000    0.9994     88927

    accuracy                         1.0000   3970161
   macro avg     0.9995    1.0000    0.9997   3970161
weighted avg     1.0000    1.0000    1.0000   3970161


ROC AUC Score: 0.9999707650313148

Confusion Matrix:
[[3881138      96]
 [      3   88924]]
```

The Random Forest model gave very high results on both the training set and the test set. It correctly predicted nearly 4 million background traffic and nearly 100,000 botnet traffic. However, the model also omitted 3 botnet traffic, but with an F1-score of around 0.99 this model worked very well.

Figure 9

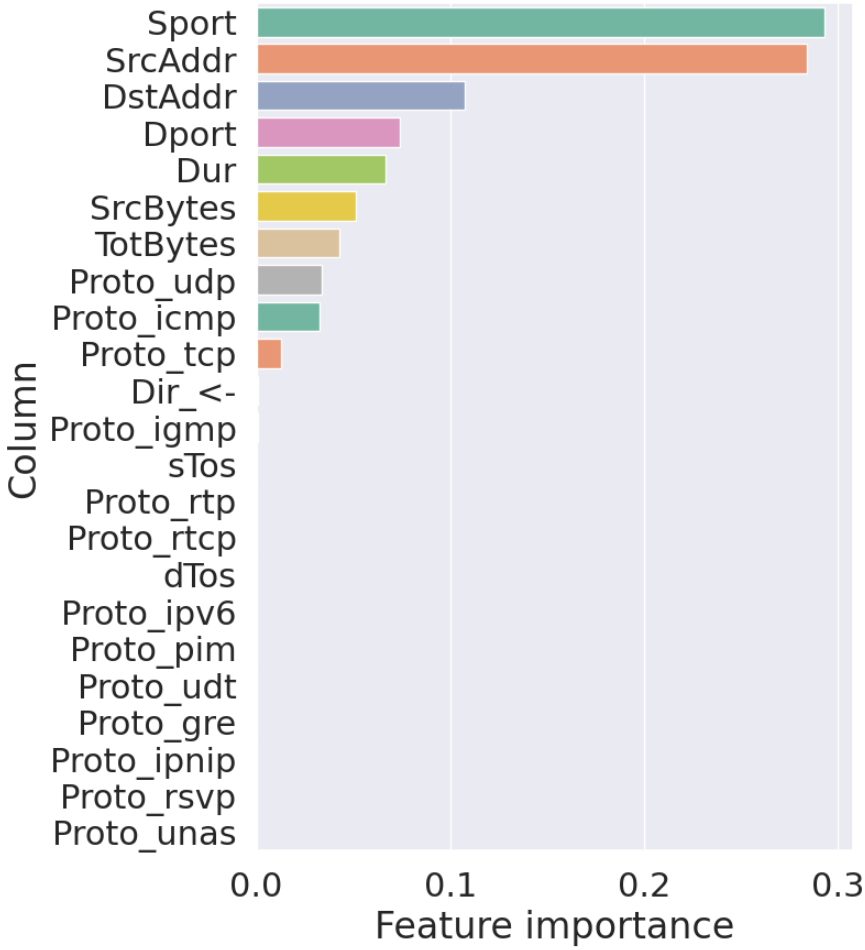
Results of the Random Forest model on a new dataset of an unknown botnet

```
TEST RESULTS:

Classification Report:
              precision    recall  f1-score   support

           0     1.0000    0.9999    0.9999    113941
           1     0.8400    1.0000    0.9130        63

    accuracy                         0.9999    114004
   macro avg     0.9200    0.9999    0.9565    114004
weighted avg     0.9999    0.9999    0.9999    114004


ROC AUC Score: 0.9999473411677974

Confusion Matrix:
[[113929     12]
 [     0     63]]
```

To test the effectiveness of the model in practice, we tested it against a new dataset of a botnet that it had not previously encountered. We were very satisfied with the results as the model correctly predicted over 100,000 instances of background traffic and identified 63 instances of botnet traffic. Additionally, the model did not miss any botnets when assuming that some background traffic was botnet traffic, which helped to protect users' computers more effectively.

Figure 10

Feature Importance of the Random Forest model



The most important features for botnet prediction were extracted from the Random Forest model. According to the model, Sport, SrcAddr, DstAddr, and Dur were the most crucial features for identifying botnet traffic. These features were practical because botnets typically spread from specific addresses and have a relatively short duration.

XGBoost

XGBoost (Extreme Gradient Boosting) is a powerful gradient boosting algorithm that uses a gradient boosting framework to enhance the performance of decision trees. It is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. XGBoost is known for its high accuracy and speed and is commonly used in various applications such as financial analysis, speech recognition, and image classification.

We built and trained the model based on the strategies listed above. The final parameters that we got after fine tuning the model are listed below:

{base_score=None, booster=None, callbacks=None,colsample_bylevel=None,
colsample_bynode=None,colsample_bytree=None,
early_stopping_rounds=None,enable_categorical=False, eval_metric=None,
feature_types=None,gamma=None, gpu_id=None, grow_policy=None,
importance_type=None,interaction_constraints=None, learning_rate=None,
max_bin=None,max_cat_threshold=None, max_cat_to_onehot=None,max_delta_step=None,
max_depth=None, max_leaves=None,min_child_weight=None, missing=nan,
monotone_constraints=None,n_estimators=100, n_jobs=None,
num_parallel_tree=None,predictor=None, random_state=42}

Figure 11

Results of the XGBoost model on 12 datasets of 6 botnets

```
TEST RESULTS:

Classification Report:
              precision    recall  f1-score   support

           0     1.0000    1.0000    1.0000   3881234
           1     0.9992    1.0000    0.9996     88927

    accuracy                         1.0000   3970161
   macro avg     0.9996    1.0000    0.9998   3970161
weighted avg     1.0000    1.0000    1.0000   3970161


ROC AUC Score: 0.9999849731851218

Confusion Matrix:
[[3881161      73]
 [      1   88926]]
```

The XGBoost model achieved outstanding results in both the training and test sets. It accurately predicted almost 4 million instances of background traffic and nearly 100,000 instances of botnet traffic. Although the model omitted 1 botnet traffic instances, its high F1-score of approximately 0.99 indicated that it was highly effective.

Figure 12

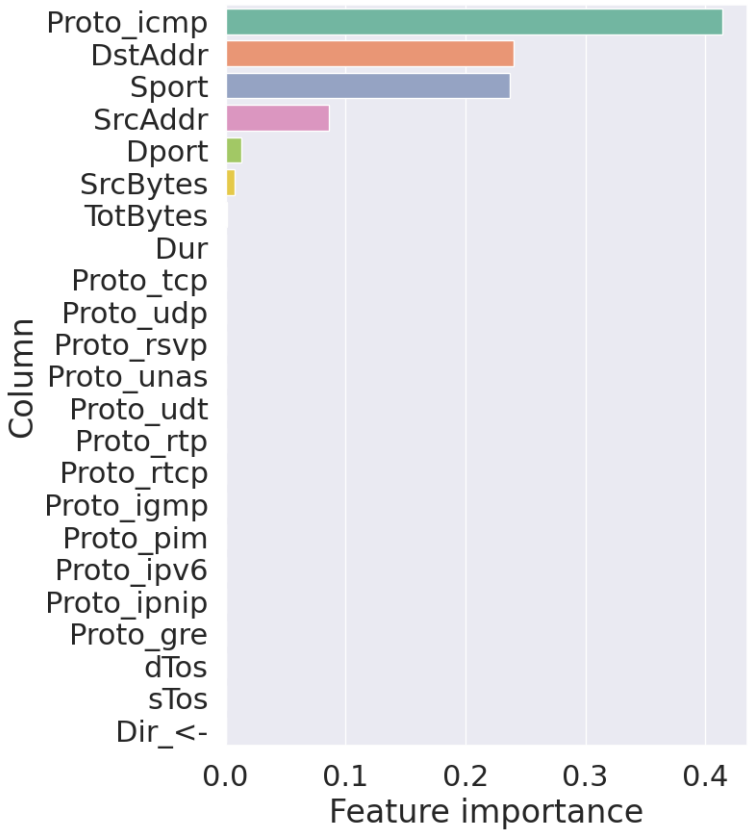
Results of the XGBoost model on a new dataset of an unknown botnet

```
TEST RESULTS:

Classification Report:
              precision    recall  f1-score   support

           0     1.0000    1.0000    1.0000    113941
           1     0.9844    1.0000    0.9921        63

    accuracy                         1.0000    114004
   macro avg     0.9922    1.0000    0.9961    114004
weighted avg     1.0000    1.0000    1.0000    114004


ROC AUC Score: 0.9999956117639831

Confusion Matrix:
[[113940      1]
 [     0     63]]
```

To assess the model's performance in a practical setting, we tested it on a new dataset containing botnet traffic that it had not previously encountered. The results were impressive, with the model correctly identifying over 100,000 instances of background traffic and identifying 63 instances of botnet traffic. Additionally, the model did not miss any botnets when considering some background traffic as botnet traffic, improving the protection of users' computers.

Figure 13

Feature Importance of the XGBoost model



The XGBoost model identified Proto_icmp, DstAddr, Sport, and SrcAddr as the most critical features for botnet prediction. These features were practical and could potentially aid in developing effective network security solutions.

LightGBM

LightGBM (Light Gradient Boosting Machine) is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be highly efficient in terms of memory usage and training speed, making it well-suited for large datasets. LightGBM uses a technique called gradient-based one-side sampling (GOSS) to perform feature selection, which helps to improve accuracy while reducing training time.

Figure 14

Results of the LightGBM model on 12 datasets of 6 botnets

```
TEST RESULTS:

Classification Report:
              precision    recall  f1-score   support

           0     1.0000    1.0000    1.0000   3881234
           1     0.9982    1.0000    0.9991     88927

    accuracy                         1.0000   3970161
   macro avg     0.9991    1.0000    0.9995   3970161
weighted avg     1.0000    1.0000    1.0000   3970161


ROC AUC Score: 0.9999788726987345

Confusion Matrix:
[[3881070     164]
 [      0   88927]]
```

The LightGBM model performed exceptionally well on both the training and test sets, accurately identifying almost 4 million instances of background traffic and nearly 100,000 instances of botnet traffic. It outperformed the other models with the lowest false-negative rate, ensuring that no single botnet traffic was ignored or confused.

Figure 15

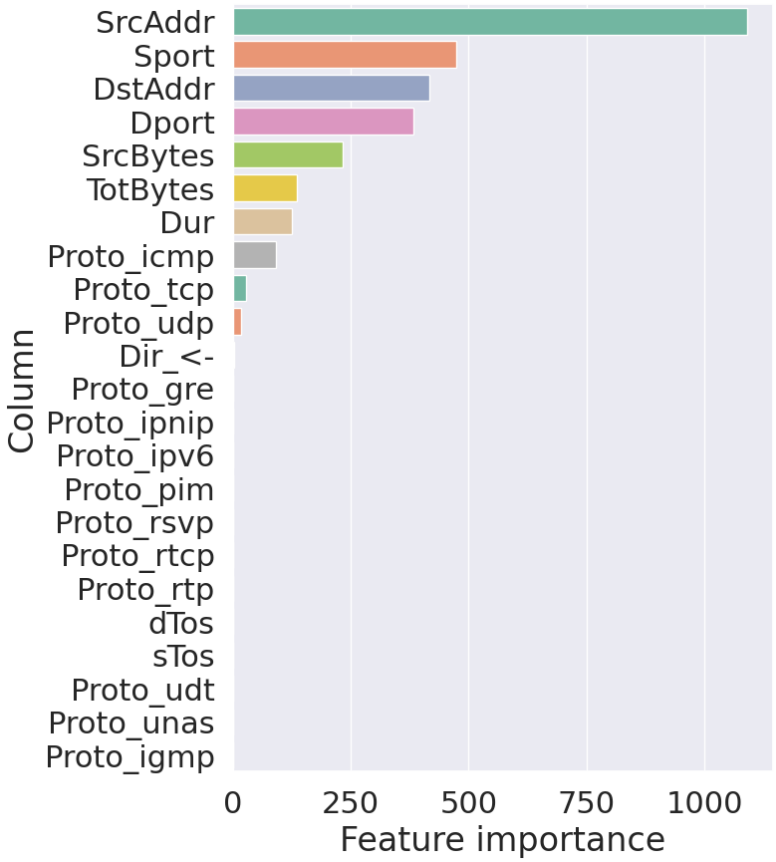
Results of the LightGBM model on a new dataset of an unknown botnet

```
TEST RESULTS:

Classification Report:
              precision    recall  f1-score   support

           0     1.0000    0.9999    1.0000    113941
           1     0.8630    1.0000    0.9265        63

    accuracy                         0.9999    114004
   macro avg     0.9315    1.0000    0.9632    114004
weighted avg     0.9999    0.9999    0.9999    114004


ROC AUC Score: 0.9999561176398312

Confusion Matrix:
[[113931     10]
 [     0     63]]
```

To further validate its effectiveness, the model was tested against a new dataset of botnet traffic that it had not encountered before, where it successfully identified over 100,000 instances of background traffic and 63 instances of botnet traffic. The model's capability to recognize botnet traffic accurately while minimizing false negatives is crucial in providing better protection for users' computers.

Figure 16

Feature Importance of the LightGBM model



The most important features for botnet prediction were extracted from the LightGBM model. According to the model, SrcAddr, Sport, DstAddr, Dport, ScrBytes, Dur were identified as the most crucial features for identifying botnet traffic. These features were practical because we needed to know the source and destination addresses of botnet traffic, and these traffics also had relatively small packet sizes and relatively short durations. These features were extremely useful for botnet detection in practice.

Artificial Neural Network

Artificial neural networks (ANNs) are a set of algorithms modeled after the human brain that are used for machine learning and artificial intelligence applications. ANNs are composed of interconnected processing nodes that simulate the behavior of neurons in the brain. ANNs can be used for a wide variety of tasks, including image and speech recognition, natural language processing, and predictive analytics. They have become increasingly popular in recent years due to their ability to handle complex, high-dimensional data and produce accurate predictions.

Figure 17

Neural Network Architecture

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 10)                240

 dense_1 (Dense)             (None, 10)                110

 dense_2 (Dense)             (None, 1)                 11

=================================================================
Total params: 361
Trainable params: 361
Non-trainable params: 0
_________________________________________________________________
```

As, seen in Figure 17, we chose a simple architecture for the neural network due to computional constraints. The neural network consists of a Input layer, 2 dense hidden layers and an output layer and gives us the proprability of the packet representing normal/background traffic or botnet traffic. Similar to the testing techniques used in previous models, we used a 80-20 split of 12 capture scenarios to train the model and test it on a set of data generated by a network, the model is not familiar with. We trained the model for 50 epochs initially but saw no

improvements in the model beyond 20 epochs. For the final experiments, we limited the training to 20 epochs.

Figure 18

Results of the Neural Network model on 12 datasets of 6 botnets

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

           0     0.9948    0.8335    0.9070   3881234
           1     0.1001    0.8086    0.1782     88927

    accuracy                         0.8329   3970161
   macro avg     0.5474    0.8210    0.5426   3970161
weighted avg     0.9747    0.8329    0.8907   3970161


ROC AUC Score: 0.8210383504960316

Confusion Matrix:
[[3234978  646256]
 [  17022   71905]]
```

The result of testing on 12 scenarios can be seen in Figure 18, the performance of the model is acceptable but not anywhere close to the previous machine learning methods we tested. The model is only able to achieve 83% accuracy on a test set with AUC 0.82. Furthermore, testing it on unknown data, the model performance is even poorer. The accuracy hovers around 84% and AUC is lowered to 0.62. The results can be surmised from Figure 19. Although the model performance is not as good as conventional techniques, other architectures and model tuning have shown better results in studies.

Figure 19

Results of the Neural Network model on a new dataset of an unknown botnet

```
3563/3563 [==============================] - 3s 935us/step
Classification Report:
              precision    recall  f1-score   support

           0     0.9996    0.8397    0.9127    113941
           1     0.0014    0.4127    0.0028        63

    accuracy                         0.8394    114004
   macro avg     0.5005    0.6262    0.4578    114004
weighted avg     0.9991    0.8394    0.9122    114004


ROC AUC Score: 0.626189299028751

Confusion Matrix:
[[95674 18267]
 [   37    26]]
```

Conclusion and Future work

In this project, we employed the CTU-13 Dataset to detect botnet traffic. The dataset contained nearly 20 million rows of data and 15 features, which is a comprehensive resource to analyze and detect botnets. To ensure the data was ready for machine learning models, we conducted various data science steps, including data cleaning, transformation, visualization, feature selection, under sampling techniques, and feature scaling. These measures were necessary to ensure the accuracy, reliability, and effectiveness of the machine learning models we built. We developed four models to detect the presence of botnets in the network traffic data: Random Forest, XGBoost, LightGBM, and Neural Network. Although the conventional machine learning models performed similarly in terms of accuracy, LightGBM had the smallest false-negative rate on both the training and test sets, indicating that it was better than the other models at detecting botnets. Additionally, we found that the most important features for prediction, such as address, port, duration, and packet size, were practical and could be easily monitored in a network, making LightGBM a practical solution for detecting botnets in a network environment.

Despite being powerful models for handling complex data, the Neural Networks architecture we used was not the best fit for our specific problem, as was unable to achieve accuracy rates higher than 89%. Our team plans to create a pipeline and deploy the LightGBM model in production moving forward. We will also verify false positives and retrain the models as necessary to maintain accuracy. Our work highlights the importance of using comprehensive datasets, proper data preprocessing techniques, and choosing the right machine learning models to solve complex problems in network security. By conducting these measures, we were able to accurately detect botnet traffic and prevent damage to network infrastructure.

References

[1] García, S., Grill, M., Stiborek, J., & Zunino, A. (2014). An empirical comparison of botnet detection methods. *Computers & Security*, *45*, 100–123. https://doi.org/10.1016/j.cose.2014.05.011

[2] Khan, R. U., Zhang, X., Kumar, R., Sharif, A., Golilarz, N. A., & Alazab, M. (2019, June 11) *An Adaptive Multi-Layer Botnet Detection Technique Using Machine Learning Classifiers*. MDPI. Retrieved January 25, 2023, from https://www.mdpi.com/2076-3417/9/11/2375

[3] Sarker, I.H. Deep Cybersecurity: A Comprehensive Overview from Neural Network and Deep Learning Perspective. *SN COMPUT. SCI.* **2**, 154 (2021). https://doi.org/10.1007/s42979-021-00535-6

[4] García, S., Grill, M., Stiborek, J., & Zunino, A. (2014). An empirical comparison of botnet detection methods. Computers & Security, 45, 100–123. https://doi.org/10.1016/j.cose.2014.05.011

[5] Corporation, M. (2005, March 21). *Backdoor:Win32/Rbot threat description - Microsoft Security Intelligence*. Backdoor:Win32/Rbot Threat Description - Microsoft Security Intelligence. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Backdoor:Win32/Rbot

[6] *DONBOT - Threat Encyclopedia*. (2012, October 9). DONBOT - Threat Encyclopedia. https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/DONBOT

[7] Rebekah Houser, Daiping Liu, J. S., Szurdi, J., Houser, R., & Liu, D. (2021, March 2). *Fast Flux 101: How Cybercriminals Improve the Resilience of Their Infrastructure to Evade Detection and Law Enforcement Takedowns*. Unit 42. https://unit42.paloaltonetworks.com/fast-flux-101/

[8] *Fortiguard*. (n.d.). FortiGuard. https://fortiguard.com/encyclopedia/virus/7360473

[9] *Fortiguard*. (n.d.). FortiGuard. https://fortiguard.com/encyclopedia/virus/3475312/w32-qvod-jn-tr-dldr

[10] *Fortiguard*. (n.d.). FortiGuard. https://fortiguard.com/appcontrol/16832

[11] Alani, M. (2021, December 14). *Handling IP Addresses in Machine Learning Datasets – Mohammed M. Alani*. https://www.mohammedalani.com/tutorials/handling-ip-addresses-in-machine-learning-datasets/

[12] *Botnets*. (2016, February 17). ENISA. https://www.enisa.europa.eu/topics/incident-response/glossary/botnets

[13] Bederna, Z., & Szádeczky, T. (2021). Effects of botnets – a human-organisational approach. *Security and Defence Quarterly*, *35*(3), 25–44. https://doi.org/10.35467/sdq/138588

[14] *Taking down botnets*. (2014, July 15). Federal Bureau of Investigation. https://www.fbi.gov/news/testimony/taking-down-botnets

[15] *IPv4 - Packet Structure*. (n.d.). Tutorialspoint.com. Retrieved March 29, 2023, from https://www.tutorialspoint.com/ipv4/ipv4_packet_structure.htm

[16] Tobiyama, S., Yamaguchi, Y., Shimada, H., Ikuse, T., & Yagi, T. (2016). Malware detection with deep neural network using process behavior. *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*.

[17] Larriva-Novo, X. A., Vega-Barbas, M., Villagra, V. A., & Sanz Rodrigo, M. (2020). Evaluation of cybersecurity data set characteristics for their applicability to neural networks algorithms detecting cybersecurity anomalies. *IEEE Access: Practical Innovations, Open Solutions*, *8*, 9005–9014. https://doi.org/10.1109/access.2019.2963407

[18] Ahmed, A. A., Jabbar, W. A., Sadiq, A. S., & Patel, H. (2022). Deep learning-based classification model for botnet attack detection. *Journal of Ambient Intelligence and Humanized Computing*, *13*(7), 3457–3466. https://doi.org/10.1007/s12652-020-01848-9