Botnet Detection Using Machine Learning

Table of Contents

Table of Figures

Table of Tables

1) Overview:

1.1 Introduction

With the advancements of technology and the establishment of the Internet of Things (IoT), there has been an increase in one of the most dangerous forms of cybercrime attacks called botnet attacks. Botnet attacks use a group of infected internet-connected devices, such as computers, servers, or mobile devices, called botnets. This term originates from the words ‘robot’ and ‘network’, and is sometimes referred to as a ‘zombie’.

Cybercriminals hijack devices connected on the internet by infecting malware using different methods like popup ads or email links, with the user being unaware of these malicious attempts until it’s too late. The more devices infected, the larger the botnet becomes, giving the intruder more resources and functionality to perform the intended cyber-attacks.

1.2 Problem Statement

Botnets impose a very big threat to individuals and organizations as they are used to inaugurate different types of cyber-attacks such as phishing, denial-of-service (DOS) distributed denial-of-service (DDoS), or spamming. An intruder can use any security vulnerability as an access point for a botnet. Since every botnet is set up in a unique way, it is very hard to detect. Furthermore, intruders make sure that their victims are oblivious to the attacks.

To be able to detect and prevent botnets would definitely help make networks more secure and protect devices that are connected to the internet. This project aims to use machine-learning techniques in order to detect botnets by analyzing different features extracted to help prevent malicious activity.

1.3 Literature Review

Over the years, a lot of research was conducted to show the efficiency of different machine-learning approaches to detect botnets.

In centralized environments, Carl Livadas used supervised machine learning to detect IRC botnets and classified two beneficial features: ‘average bytes per packet and variance of bytes per packet’ [2] [3]. Other experiments used different methods with these features as well. However, since centralized botnets have a single host to manage the other bots, this became an easy target for failure. Thus, botnets that have a decentralized or distributed topology increasingly became formed.

In decentralized environments, Yen and Reiter’s detection approach focuses on P2P nodes. With the assumption of these nodes being classified as P2P when there were continuous outgoing flows or diverse networks, they used two steps to detect malicious activity: identification followed by detection. The identification step was to use ‘statistical features’ and ‘flow activity time’ [2]. The detection step was then implemented by using more features based on two further assumptions: ‘clients in the same botnet use the same protocol and network, and there is a significant overlap in peers that P2P bots connect to’ [2]. This experiment showed high detecting precision.

Taking another approach, Anchit Bijalwan [4] used three machine learning algorithms: KNN, SVM and Decision Tree. The testing accuracies of each algorithm used is summarized in the table below:

Metrics	**Decision tree**	**KNN**	**SVM**
Accuracy	93.7	94.65	75.99
Precision	92.09	95.0	81.07
Recall	93.48	95.0	76.05
F1 score	94.76	95.0	66.78

Table 1: Anchit Bijalwan's research results

Overall, the KNN algorithm had the highest percentages in all of the evaluation metrics and SVM had the lowest, making it not suitable for detecting botnets.

In the paper published by (Faek, Al-Fawa'reh and Al-Fayoumi) in 2021 [5], the authors proposed to use Netflow and Machine Learning algorithms to improve the detection process for intrusion attacks with a novel dataset. They found that Random Forests get the highest accuracy compared to other algorithms, and they highlighted the importance of attack detection systems in the security infrastructure. In their work, they used machine learning algorithms for the detection of botnets on network traffic. They proposed a lightweight model implemented with a number of algorithms and tested with a novel dataset.

The main hypothesis of this work is based on the proved fact that the use of Machine learning (ML) in the detection of Abnormal and their ability to determine anomalies over a period of time by revealing the irregular features in a data, are considered as one of the most important trend in the research in the network security field. ML can provide methods for botnet detection using different strategies able to detect patterns automatically to predict future trends in data.

The researchers had used a dataset that targeted the following features: generality, reality, representation. The dataset consists of a two-step process: training and testing, where the model was trained on 80% of the data, and tested it on the remaining 20%. The data were labeled based on the type of attack (Meris, Rbot, Virut, NSIS, Menti, Zero access, Weasel, Smoke Bot, Sogou, Murlo and so on) after removing the missing values. Two methods were used throughout the experiments: Decision Tree and Random Forest.

The results of this research showed that Random Forests achieved the highest accuracy, totaling 97.9% when they used 25 features, and 99.1% when using both 76 and 69 features. Decision Tree algorithm achieved similarly high accuracy, while Quadratic Discriminant Analysis had lower accuracy, and Gaussian NB had the least accuracy of the studied algorithms.

1.4 Dataset

The dataset used in this project is called the ‘BoT-IoT’ dataset designed by the UNSW Canberra university, where they already split it into training and testing sets with the following features:

Feature	**Description**
pkSeqID	Row Identifier
Stime	Record start time
Flgs	Flow state flags seen in transactions
flgs_number	Numerical representation of feature flags
Proto	Textual representation of transaction protocols present in network flow
proto_number	Numerical representation of feature proto
Saddr	Source IP address
Sport	Source port number
Daddr	Destination IP address
Dport	Destination port number
Pkts	Total count of packets in transaction
Bytes	Totan number of bytes in transaction
State	Transaction state
state_number	Numerical representation of feature state
Ltime	Record last time
Seq	Argus sequence number
Dur	Record total duration
Mean	Average duration of aggregated records
Stddev	Standard deviation of aggregated records
Sum	Total duration of aggregated records
Min	Minimum duration of aggregated records
Max	Maximum duration of aggregated records
Spkts	Source-to-destination packet count
Dpkts	Destination-to-source packet count
Sbytes	Source-to-destination byte count
Dbytes	Destination-to-source byte count
Rate	Total packets per second in transaction
Srate	Source-to-destination packets per second
Drate	Destination-to-source packets per second
TnBPSrcIP	Total Number of bytes per source IP
TnBPDstIP	Total Number of bytes per Destination IP.
TnP_PSrcIP	Total Number of packets per source IP.
TnP_PDstIP	Total Number of packets per Destination IP.
TnP_PerProto	Total Number of packets per protocol.
TnP_Per_Dport	Total Number of packets per dport
AR_P_Proto_P_SrcIP	Average rate per protocol per Source IP. (calculated by pkts/dur)
AR_P_Proto_P_DstIP	Average rate per protocol per Destination IP.
N_IN_Conn_P_SrcIP	Number of inbound connections per source IP.
N_IN_Conn_P_DstIP	Number of inbound connections per destination IP.
AR_P_Proto_P_Sport	Average rate per protocol per sport
AR_P_Proto_P_Dport	Average rate per protocol per dport
Pkts_P_State_P_Protocol_P_DestIP	Number of packets grouped by state of flows and protocols per destination IP.
Pkts_P_State_P_Protocol_P_SrcIP	Number of packets grouped by state of flows and protocols per source IP.
Attack	Class label: 0 for Normal traffic, 1 for Attack Traffic
Category	Traffic category
Subcategory	Traffic subcategory

Table 2: Dataset features and descriptions

We visualized the attacks that are classified into categories to analyze the different percentages of each attack used in the training and testing dataset.

Training Dataset:

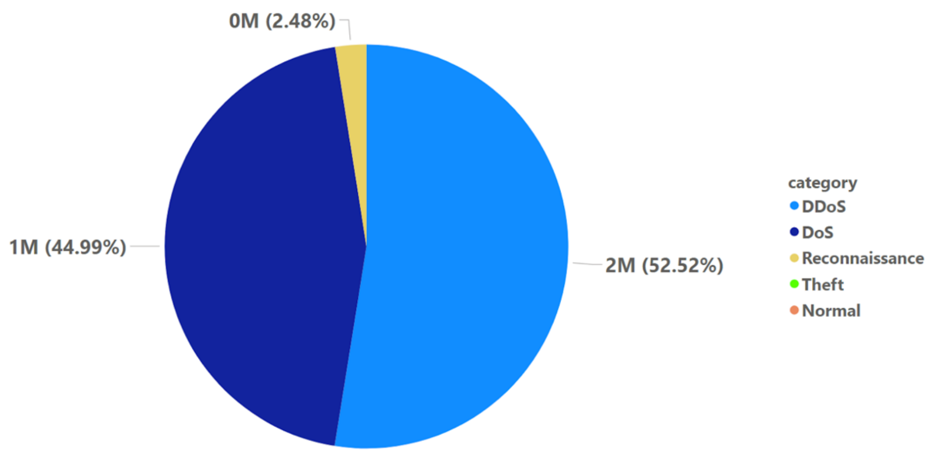

Figure 1: Pie chart of training dataset

Shown above is the relationship between category and attack.

DDoS and DoS attacks were accounted for the most from the sum of the attacks with a combined percentage of 97.51%. Also, the number of attacks in the entire training dataset is approximately equal to 2.93 million rows.

Testing Dataset

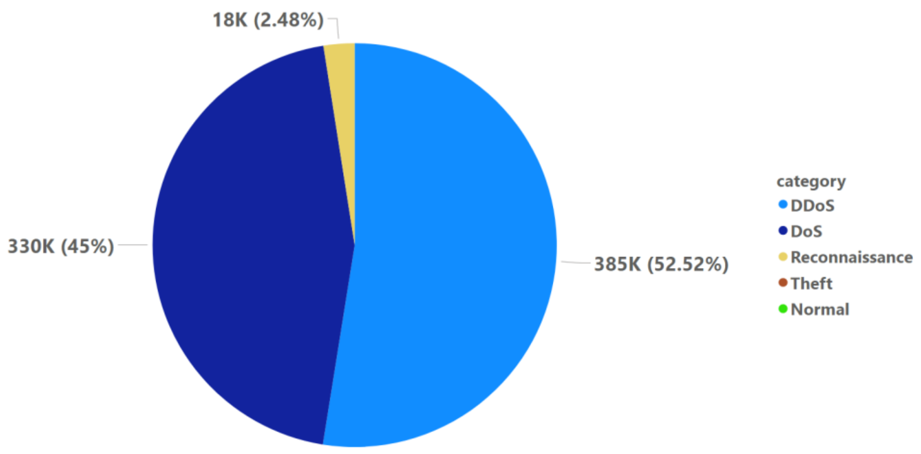

Figure 2: Pie chart of testing dataset

Shown above is the relationship between category, subcategory and attack.

DDoS and DoS attacks were accounted for the most from the sum of the attacks with a combined percentage of 97.52%. Also, the number of attacks in the entire testing dataset is approximately equal to 0.07 million rows.

2) Methodology:

As the data is already split into training and testing sets, we started to study the features to familiarize ourselves with them. Moreover, the source of our dataset extracted the best features to use in the ML model. Firstly, we dropped all rows that contain any null value.

Next, to transform the data into a better range (0-1), we encoded the numerical data using a MinMaxScaler and the categorical data using an hotEncoder.

Before we began modeling, we applied some methods to clean our dataset and prepare it for the modeling phase where we tried two ways for splitting the dataset to training and testing sets:

1. **Whole dataset**: Using the already existing training and testing sets.
2. **Undersampled dataset:** Using cross validation to train and test the dataset.

We built a model that predicted the category of the attack using the category feature which contains 5 types of attacks: DoS, DDoS, theft, Reconnaissance and Normal. The algorithm used in this project for both approaches was the K-Nearest Neighbor (KNN) algorithm.

2.1 Whole dataset

2.1.1 Pre-processing

Since the dataset as a whole was already fit to be trained, there was no need for extra pre-processing steps.

2.1.2 Training and testing

We trained the data on different k-values of the KNN and tested the predicted data each time by calculating the accuracy, recall, precision, and f1 score. In all of the cases we achieved an accuracy of rounded to 0.999.

Since we are predicting not just a single value rather, 5 values representing different categories; it is recommended that we use average values for the evaluation metrics. It would not be fair if we only looked at the accuracy alone since we have 5 different categories. So, an accuracy alone would not give a full picture of how well the model behaved. In order to have a deeper insight we would have to look at the macro average and weighted average metrics. The macro average calculates the unweighted average values of evaluation metrics without taking into account the weight of each category in the data. However, the weighted average takes into account the weight of each category to give a more reliable metric.

	Macro average			**Weighted average**		
K-value	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
5	0.82064	0.94363	0.84819	0.99984	0.99945	0.99962
15	0.84551	0.94247	0.86322	0.99985	0.99958	0.99970
25	0.87485	0.93501	0.89234	0.99987	0.99978	0.99982
35	0.89087	0.95551	0.91095	0.99990	0.99982	0.99985

Table 3: Evaluation metrics whole dataset

2.2 Under sampled dataset

From the exploration phase, we found out that the dataset is unbalanced, the class True (meaning there is an attack) covers 99.9% of the records and the class False (meaning there is no attack) covers 0.01% of the records when analyzing the attack column.

Due to this imbalance, we decided to under sample the dataset by extracting a subset with an equal number of attacks labeled records. As the not attacked labeled records are very limited in number, we extracted the subset from both the training and testing sets. Then we selected all records that have the value false in column attack (which is equal to 477) and put it together with the same number of records that have the value true in column attack in one data frame, which we will use as an input to our model (954 records). We also shuffled the resulting data frame to make sure that the records are randomized.

As for the training and testing, we implemented 5-fold cross validation and we experimented with a range of k values (between 3 and 50) to find the most suitable k for our model. After running the proposed model, we got the best f1 score which is equal to 99.621% when k is equal to 7. However, it’s important to mention that the scores in general are very close in value.

	Macro average	**Weighted average**

K-value	Precision	Recall	F1-score	Precision	Recall	F1-score
5	0.99649	0.96225	0.97798	0.99499	0.99494	0.99485
7	0.99764	0.96326	0.97907	0.99624	0.99621	0.99612
9	0.99609	0.96123	0.97726	0.99378	0.99368	0.99360

Table 4: Evaluation metrics under sampled dataset

4) Conclusion:

We proved that the usage of machine learning techniques is helpful in the detection of the attacks that may occur on the network. In this work, we proposed a model that is able to predict the occurrence of an attack based on the behavior of the traffic. In addition, our model is able to identify the type of the attack from a list of possible attacks: DoS, DDoS, theft, and Reconnaissance. The results of our work showed that the KNN model is good for this type of problem. It gives an accuracy above 99% when using the whole dataset and also when we under sampled the dataset.

Despite those high numbers of accuracies and weighted averages, we genuinely believe that the model is overfitted and further testing should be made especially using different algorithms and with much more data filtering that we used here.

5) References:

Dataset: https://research.unsw.edu.au/projects/bot-iot-dataset

[1] Hanna, K. T., Lutkevich, B., & Wright, R. (2021, March 30). *What is botnet?* Search Security. Retrieved June 5, 2022, from https://www.techtarget.com/searchsecurity/definition/botnet

[2] E. Biglar Beigi, H. Hadian Jazi, N. Stakhanova and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," 2014 IEEE Conference on Communications and Network Security, 2014, pp. 247-255, doi: 10.1109/CNS.2014.6997492.

[3] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, “Using machine learning techniques to identify botnet traffic,” in *Local Computer Net-works, Proceedings 2006 31st IEEE Conference on*. IEEE, 2006, pp. [33] 967–974.

[4] Bijalwan, Anchit. (2020). Botnet Forensic Analysis Using Machine Learning. Security and Communication Networks. 2020. 1-9. 10.1155/2020/9302318.

[5] Faek, Rana & Al-Fawa'reh, Mohammad & Al-Fayoumi, Mustafa. (2021). Exposing Bot Attacks Using Machine Learning and Flow Level Analysis. 99-106. 10.1145/3460620.3460739.