ZKG INTERNATIONAL ISSN: 2366-1313

PREDICTION OF BOTNET ATTACKS FOR IoT DEVICES USING MACHINE LEARNING TECHNIQUES

[1]Dr. A. PRANAYANATH REDDY, [2]A. GANESH, [3]D. RENUKA SINDHU,

[4]G. MEGHANA

[1](Assistant Professor) ,**CSE.** Teegala Krishna Reddy Engineering College Hyderabad

[234]B,tech scholar ,**CSE.** Teegala Krishna Reddy Engineering College Hyderabad

ABSTRACT

There are an increasing number of Internet of Things (IoT) devices connected to the network these days, and due to the advancement in technology, the security threads and cyberattacks, such as botnets, are emerging and evolving rapidly with high-risk attacks. These attacks disrupt the IoT transition by disrupting networks and services for IoT devices. Many recent studies have proposed ML and DL techniques for detecting and classifying botnet attacks in the IoT environment. This study proposes machine learning methods for classifying binary classes i.e., Benign, or TCP attacks. A complete machine learning pipeline is proposed, including exploratory data analysis, which provides detailed insights into the data, followed by preprocessing. During this process, the data passes through several fundamental steps. A random forest, k-nearest neighbor, support vector machines, and a logistic regression model are proposed, trained, tested, and evaluated on the dataset. In addition to model accuracy, F1-score, recall, and precision are also considered.

1.INTRODUCTION

1.1 Problem Statement

The traditional attack detection systems cannot be competently relocated in the IoT environments because of the different nature of such devices, and the diverse architecture of the underlying network methodologies with the conventional network. Additionally, the possible attacks can be distinct from the attacks that are found on traditional network

ISSN: 2366-1313

devices. The heavyweight encryption methods cannot be deployed on these resource-constrained devices. On the other side, IoT devices become very cheap to set up for personal usage, like in small businesses and smart home appliances. The attackers were launching the attacks on the victim nodes after infecting the botnets on these devices. They can also circumvent formal rule-based detection systems. Although the machine learning-based system can detect the variances of the many kinds of attacks, new distinct kinds of attacks can be launched sometimes. Additionally, the complex processing of ML classifiers is a challenge to implementing the lightweight attack detection system on resource constrained devices.

1.2 DESCRIPTION

The general idea of the Internet of Things (IoT) is to allow for communication between human-to-thing or thing-to-thing(s). Things denote sensors or devices, whilst a human or an object is an entity that can request or deliver a service [1]. The interconnection among the entities is always complex. IoT is broadly acceptable and implemented in various domains, such as healthcare, smart home, and agriculture. However, IoT has resource constraints and heterogeneous environments, such as low computational power and memory. These constraints create problems in providing and implementing a security solution in IoT devices.

These constraints further escalate the existing challenges for the IoT environment. Therefore, various kinds of attacks are possible due to the vulnerability of IoT devices. IoT-based botnet attacks are one of the most popular, spread faster, and create more impact than other attacks. In recent years, several works have been conducted to detect and avoid this kind of attack [2]–[3] by using novel approaches. Hence, a plethora of relevant models, methods, etc. have been introduced over the past few years, with quite a reasonable number of studies reported in the research domain. 2 Many studies are trying to protect against these botnet attacks on the IoT environment. However, there are many gaps still existing to develop an effective detection mechanism. An intrusion detection system (IDS) is one of the efficient ways to deal with attacks. However, traditional IDSs are often not able to be deployed for the IoT environments due to the resource constraint problem of these devices. The complex cryptographic

ISSN: 2366-1313

mechanisms cannot be embedded in many IoT devices either for the same reason.

There are mainly two kinds of IDSs: the anomaly and misuse approaches. The misuse-based, also called the signature-based, approach, is based on the attacks' signatures, and they can also be found in most public IDSs, specifically Suricata [4]. Formally, the attacker can easily circumvent the signature-based approaches, and these mechanisms cannot guarantee to detect the unknown attacks and the variances of known attacks. The anomalybased systems are based on normal data and can support to identify the unknown attacks. However, the different nature of IoT devices is being faced with the difficulty of collecting common normal data. Machine learning-based detection can guarantee detection of not only the known attacks and their variances. Therefore, we proposed a machine learning-based botnet attack detection architecture. We also adopted a feature selection method to reduce the demand for processing resources for performing the detection system on resource-constrained devices. The experiment results indicate that the detection accuracy of our proposed system is high enough to detect botnet attacks. Moreover, it can support the extension for detecting new distinct kinds of attacks.

2.LITERATURE SURVEY

Soe et al. [5] adopted a lightweight detection system with a high performance. The overall detection performance achieves around 99% for the botnet attack detection using three different ML algorithms, including artificial neural network (ANN), J48 decision tree, and Naïve Bayes. The experiment result indicated that the proposed architecture can effectively detect botnet-based attacks, and can be extended with corresponding sub-engines for new kinds of attacks. Ali et al. [6] outlined the existing proposed contributions, datasets utilized, network forensic methods utilized, and research focus of the primary selected studies.

The demographic characteristics of primary studies were also outlined. The result of this review revealed that research in this domain is gaining momentum, particularly in the last 3 years (2018-2020). Nine key contributions were also identified, with Evaluation, System, and Model being the most conducted. Irfan et al. [7] classified the incoming data in the IoT, as containing malware or not. In this research, this work

samples the dataset because the datasets contain an imbalance class. After that, this work classified the sample using Random Forest. This work used Naive Bayes, K-Nearest Neighbor, and Decision Tree too as a comparison. The dataset that has been used in this research is from the UCI Machine Learning Depository's Website. The dataset showed the data traffic from the IoT Device in a normal condition and attacked by Mirai or Bashlite. Shah et al. [8] presented a concept called 'login puzzle' to prevent the capture of IoT devices on a large scale. The login puzzle is a variant of the client puzzle, which presents a puzzle to the remote device during the login process to prevent unrestricted login attempts. Login puzzle is a set of multiple mini puzzles with a variable complexity, which the remote device is required to solve before logging into any IoT device. Every unsuccessful login attempt increases the complexity of solving the login puzzle for the next attempt. This paper introduced a novel mechanism to change the complexity of the puzzle after every unsuccessful login attempt. If each IoT device had used a login puzzle, the Mirai attack would have required almost two months to acquire devices, while it acquired them in 20 hours. Tzagkarakis et al. [9] presented an IoT botnet attack detection method based on a sparsity representation framework using a reconstruction error thresholding rule for identifying malicious 4 network traffic at the IoT edge coming from compromised IoT devices.

The botnet attack detection is performed based on small-sized benign IoT network traffic data, and thus we have no prior knowledge about malicious IoT traffic data. We present our results on a real IoT-based network dataset and show the efficacy of the proposed technique against a reconstruction error-based autoencoder approach. Meidan et al. [10] proposed a novel network-based anomaly detection method for the IoT called N-BaIoT that extracts behavior snapshots of the network and uses deep autoencoders to detect anomalous network traffic from compromised IoT devices. To evaluate the method, this work infected nine commercial IoT devices in our lab with two widely known IoT-based botnets, Mirai and BASHLITE.

The evaluation results demonstrated the ability of the proposed method to detect the attacks accurately and instantly as they were being launched from the compromised IoT devices that were part of a botnet. Popoola et

ISSN: 2366-1313

al. [11] proposed the federated DL (FDL) method for zero-day botnet attack detection to avoid data privacy leakage in IoT-edge devices. In this method, an optimal deep neural network (DNN) architecture is employed for network traffic classification. A model parameter server remotely coordinates the independent training of the DNN models in multiple IoT-edge devices, while the federated averaging (FedAvg) algorithm is used to aggregate local model updates. A global DNN model is produced after several communication rounds between the model parameter server and the IoT-edge devices.

The zero-day botnet attack scenarios in IoT-edge devices are simulated with the Bot-IoT and N-BaIoT data sets. Hussain et al. [12] produced a generic scanning and DDoS attack dataset by generating 33 types of scans and 60 types of DDoS attacks. In addition, this work partially integrated the scan and DDoS attack samples from three publicly available datasets for maximum attack coverage to better train the machine learning algorithms. Afterward, this work proposed a two-fold machine learning approach to prevent and detect IoT botnet attacks.

In the first fold, this work trained a state-of-theart deep learning model, i.e., ResNet-18 to detect the scanning activity in the premature attack stage to prevent IoT botnet attacks. While, in the second fold, this work trained another ResNet-18 model for DDoS attack identification to detect IoT botnet attacks. Abu et al. [13] proposed an ensemble learning model for botnet attack detection in IoT networks called ELBA-IoT that profiles behavior features of IoT networks and uses ensemble learning to identify anomalous network traffic from compromised IoT devices. In addition, this IoT-based botnet detection approach characterizes the evaluation of three different machine 5 learning techniques that belong to decision tree techniques (AdaBoosted, RUSBoosted, and bagged). To evaluate ELBA-IoT, we used the N-BaIoT-2021 dataset, which comprises records of both normal IoT network traffic and botnet attack traffic of infected IoT devices. Alharbi et al. [14] proposed a Gaussian distribution used in the population initialization. Furthermore, the local search mechanism was followed by the Gaussian density function and localglobal best function to achieve better exploration during each generation. Enhanced BA was further

ISSN: 2366-1313

employed for neural network hyper parameter tuning and weight optimization to classify ten different botnet attacks with an additional benign target class.

The proposed LGBA-NN algorithm was tested on an N-BaIoT data set with extensive real traffic data with benign and malicious target classes. The performance of LGBA-NN was compared with several recent advanced approaches such as weight optimization using Particle Swarm Optimization (PSO-NN) and BA-NN. Ahmed et al. [15] proposed a model for detecting botnets using deep learning to identify zero-day botnet attacks in real-time. The proposed model is trained and evaluated on a CTU-13 dataset with multiple neural network designs and hidden layers. Results demonstrated that the deeplearning artificial neural network model can accurately and efficiently identify botnets.

RANDOM FOREST ALGORITHM

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting.

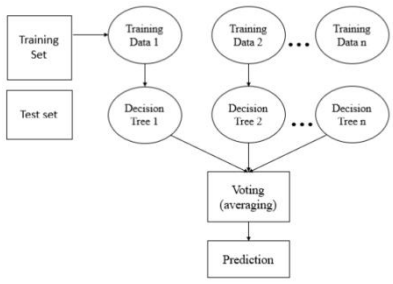


Fig. 2.1: Random Forest algorithm.

Random Forest algorithm Step 1: In Random Forest n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

ISSN: 2366-1313

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and Regression respectively. Important Features of Random Forest

• Diversity- Not all attributes/variables/features are considered while making an individual tree, each tree is different.

• Immune to the curse of dimensionality- Since each tree does not consider all the features, the feature space is reduced.

• Parallelization tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.

• Train-Test split- In a random forest we do not have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.

• Stability- Stability arises because the result is based on majority voting/ averaging. 7 Assumptions for Random Forest Since the random forest combines multiple trees to predict the class of the dataset, some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output.

Therefore, below are two assumptions for a better Random Forest classifier:

• There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.

• The predictions from each tree must have very low correlations. Below are some points that explain why we should use the Random Forest algorithm

• It takes less training time as compared to other algorithms.

• It predicts output with high accuracy, even for the large dataset it runs efficiently.

• It can also maintain accuracy when a large proportion of data is missing.

Types of Ensembles Before understanding the workings of the random forest, we must investigate the ensemble technique. Ensemble simply means combining multiple models. Thus, a collection of models is used to make predictions rather than an individual model. The ensemble uses two types of methods: Bagging– It creates a different training subset from sample training data with replacement & the final output is based on majority voting—for example, Random

ISSN: 2366-1313

Forest. Bagging, also known as Bootstrap Aggregation is the ensemble technique used by random forest. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation.

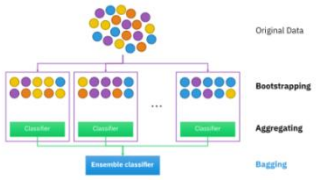


Fig. 2.2: RF classifier analysis.

Boosting– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy—for example, ADA BOOST, and XG BOOST.

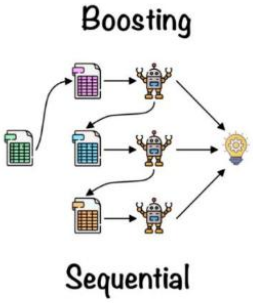


Fig. 2.3: Boosting RF classifier.

Applications of Random Forest: There are mainly four sectors where Random Forest is mostly used:

• **Banking:** The banking sector mostly uses this algorithm for the identification of loan risk.

• **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.

• **Land Use:** We can identify the areas of similar land use by this algorithm.

• **Marketing:** Marketing trends can be identified using this algorithm.

Disadvantages of random forest

• Increased accuracy requires more trees.

More trees slow down the model.

• Can't describe relationships within data.

K-Nearest Neighbor (KNN)

ISSN: 2366-1313

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on the Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most like the available categories. It stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well-suited category by using the K- NN algorithm. It can be used for Regression as well as for Classification but mostly it is used for Classification problems. It is a non-parametric algorithm, which means it does not make any assumptions on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much like the new data.

WHY DO WE NEED A K-NN ALGORITHM?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories? To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

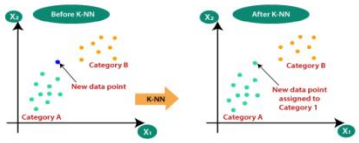


Fig. 2.4: KNN on the dataset.

How does K-NN work? The K-NN working can be explained based on the below algorithm:

Step 1: Select the number K of the neighbors.

Step 2: Calculate the Euclidean distance of K number of neighbors.

Step 3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of neighbors is maximum.

ISSN: 2366-1313

Step 6: The model is ready. Suppose we have a new data point, and we need to put it in the required category. Consider the below image:

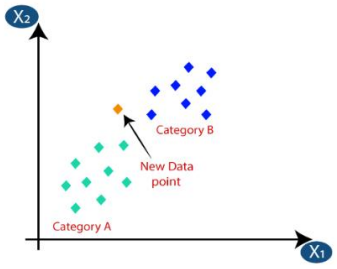


Fig. 2.5: Considering new data point.

Firstly, we will choose the number of neighbors, so we will choose the k=5. Next, we will calculate the Euclidean distance between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:

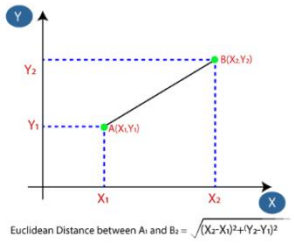


Fig. 2.6: Measuring of Euclidean distance.

By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:

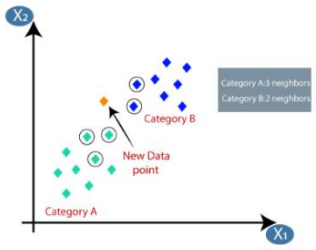


Fig. 2.7: Assigning data point to category A.

As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm? Below are some points to remember while selecting the value of K in the K-NN algorithm:

There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5. A very low value for K such as K=1 12 or K=2, can be noisy and lead to the effects of outliers in the model. Large values for K are good, but it may find some difficulties.

Disadvantages of the KNN Algorithm

Always needs to determine the value of K which may be complex sometimes. The computation cost is high because of calculating the distance between the data points for all the training samples.

Support Vector Machine Algorithm

ISSN: 2366-1313

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence algorithm is termed a Support Vector Machine. Consider the below diagram in which two different categories are classified using a decision boundary or hyper plane:

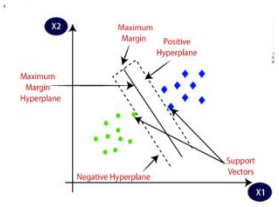


Fig. 2.8: Analysis of SVM.

Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, if we want a model that can accurately identify whether it is a cat or dog, such a model can be created by using the SVM 13 algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we will test it with this strange creature. So as the support vector creates a decision boundary between these two data (cat and dog) and chooses extreme cases (support vectors), it will see the extreme case of cat and dog. Based on the support vectors, it will classify it as a cat. Consider the below diagram:

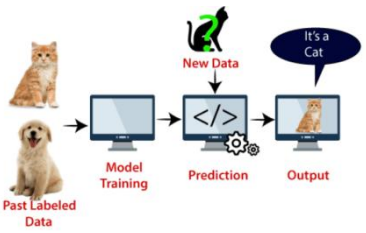


Fig. 2.9: Basic classification using SVM.

Types of SVM: SVM can be of two types

Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

Non-linear SVM: Non-linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as

ISSN: 2366-1313

non-linear data, and the classifier used is called a Non-linear SVM classifier.

SVM Working

Linear SVM: The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair (x1, x2) of coordinates in either green or blue. Consider the below image:

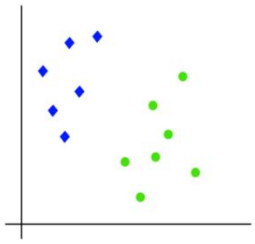


Fig. 2.10: Linear SVM.

So, as it is 2-d space by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:

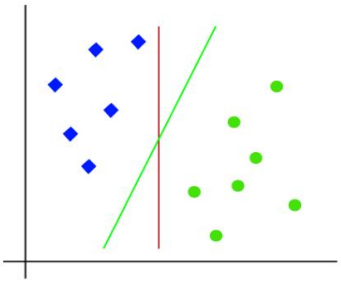


Fig. 2.11: Test-Vector in SVM.

Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called a hyperplane. The SVM algorithm finds the closest point of the lines from both classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as margin. The goal of SVM is to maximize this margin. The hyperplane with the maximum margin is called the optimal hyperplane.

Non-Linear SVM: If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:

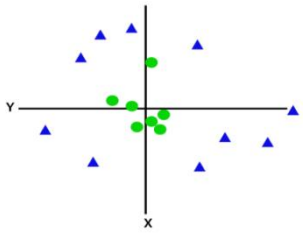


Fig. 2.13: Non-Linear SVM.

So, to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third-dimension z. It can be calculated as: z=x2 +y2 By adding the third dimension, the sample space will become as below image:

ISSN: 2366-1313

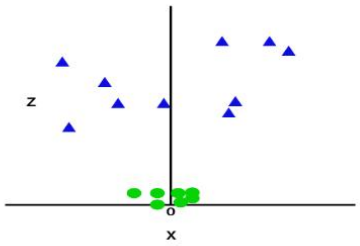


Fig. 2.14: Non-Linear SVM data separation.

Advantages of SVM

Support vector machine works comparably well when there is an understandable margin of dissociation between classes. It is more productive in high-dimensional spaces. It is effective in instances where the number of dimensions is larger than the number of specimens. Support vector machine is comparably memory systematic. Support Vector Machine (SVM) is a powerful supervised machine learning algorithm with several advantages. Some of the main advantages of SVM include: Handling high-dimensional data: SVMs are effective in handling high-dimensional data, which is common in many applications such as image and text classification.

3.SYSTEM DESIGN

3.1 System Flow

The proposed system architecture leverages machine learning techniques to detect and mitigate botnet attacks in IoT environments. At its core, the architecture comprises several key components, including data preprocessing modules for cleaning and formatting raw data, a feature selection mechanism to reduce computational demand, and a variety of machine learning classifiers such as K-Nearest Neighbor, Logistic Regression, Random Forest, Support Vector Machine, and MLP Classifier. These classifiers collectively analyze the preprocessed data to identify patterns indicative of botnet activity. Additionally, the architecture incorporates attack label modules to categorize instances as malicious or non-malicious, facilitating accurate detection. Overall, this modular and adaptable architecture offers robust protection against evolving botnet threats in resource-constrained IoT devices.

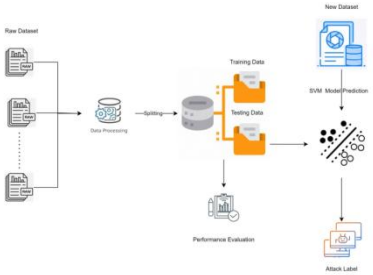
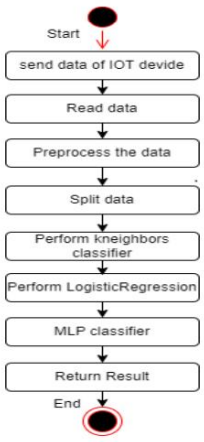


Fig. 3.1 System Flow

3.2 ACTIVITY DIAGRAM

ISSN: 2366-1313

The activity diagram is another important diagram in UML to describe the dynamic aspects of the system.



4.OUTPUT SCREENS

4.1 Loading Dataset

Loading the dataset is the initial step in our analysis pipeline, which is crucial for preparing the raw data for subsequent processing. We meticulously import the datasets, "junk.csv" and "dataset_description.txt," containing essential information about network activities and their corresponding labels. This process involves verifying data integrity, handling missing values, and encoding categorical variables as necessary. By ensuring the cleanliness and completeness of our dataset, we lay a solid foundation for accurate and reliable analysis, paving the way for effective detection of botnet attacks in IoT environments.

	MI_dir_L5_weight	MI_dir_L5_mean	MI_dir_L5_variance	MI_dir_L3_weight	MI_dir_L3_mean	M
0	1.000000	60.0	0.0	1.000000	60.0	
1	1.000000	60.0	0.0	1.000000	60.0	
2	1.000000	60.0	0.0	1.000000	60.0	
3	1.000000	590.0	0.0	1.000000	590.0	
4	1.927179	590.0	0.0	1.955648	590.0	

5 rows × 115 columns

Figure 4.1.1 Non-Malicious Dataset

	MI_dir_L5_weight	MI_dir_L5_mean	MI_dir_L5_variance	MI_dir_L3_weight	MI_dir_L3_mean	M
0	1.000000	98.000000	0.000000	1.000000	98.000000	
1	1.029191	98.000000	0.000000	1.119992	98.000000	
2	1.077270	68.295282	68.180692	1.236877	72.128396	
3	2.038100	71.094319	42.860737	2.209694	72.975393	
4	3.000117	72.062842	30.450564	3.184892	73.297101	

5 rows × 115 columns

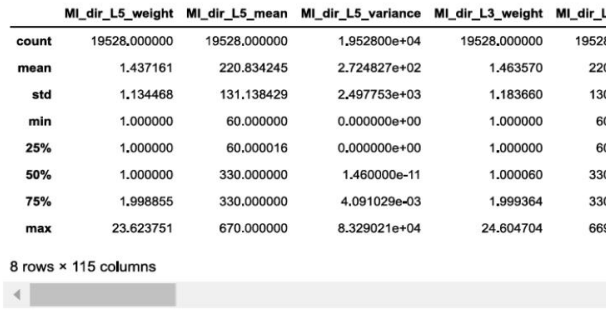
Figure 4.1.2 Malicious Dataset

4.2 Description of Dataset

The dataset comprises two key components: "junk.csv" and "dataset_description.txt." "junk.csv" contains extensive records detailing network activities, each accompanied by 116 features, facilitating comprehensive analysis. Meanwhile, "dataset_description.txt" provides vital contextual information essential for understanding the dataset's structure, format, and variables. This combined dataset, comprising 50426 records, serves as a foundational resource for our research on

ISSN: 2366-1313

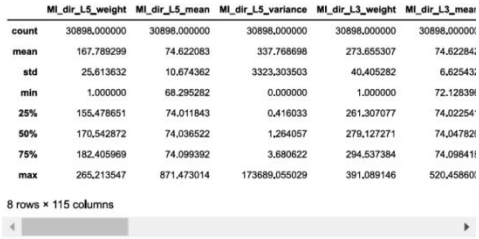
detecting IoT botnet attacks using machine learning techniques.



	MI_dir_L5_weight	MI_dir_L5_mean	MI_dir_L5_variance	MI_dir_L3_weight	MI_dir_L
count	19528.000000	19528.000000	1.952800e+04	19528.000000	1952
mean	1.437161	220.834245	2.724827e+02	1.463570	22(
std	1.134468	131.138429	2.497753e+03	1.183660	13(
min	1.000000	60.000000	0.000000e+00	1.000000	6(
25%	1.000000	60.000016	0.000000e+00	1.000000	6(
50%	1.000000	330.000000	1.460000e-11	1.000060	33(
75%	1.998855	330.000000	4.091029e-03	1.999364	33(
max	23.623751	670.000000	8.329021e+04	24.604704	66

8 rows × 115 columns

Figure 4.2.1 Description of Non-Malicious Dataset



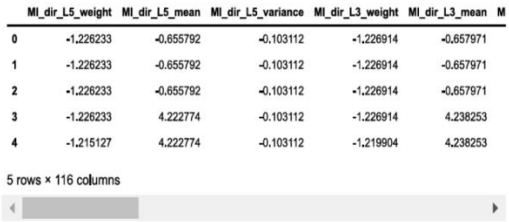
	MI_dir_L5_weight	MI_dir_L5_mean	MI_dir_L5_variance	MI_dir_L3_weight	MI_dir_L3_mea
count	30898.000000	30898.000000	30898.000000	30898.000000	30898.00000
mean	167.789299	74.622083	337.768698	273.655307	74.62284
std	25.613632	10.674362	3323.303503	40.405282	6.62543
min	1.000000	68.295282	0.000000	1.000000	72.12839
25%	155.478651	74.011843	0.416033	261.307077	74.02254
50%	170.542872	74.036522	1.264057	279.127271	74.04782
75%	182.405969	74.099392	3.680622	294.537384	74.09841
max	265.213547	871.473014	173689.055029	391.089146	520.45860

8 rows × 115 columns

Figure 4.2.2 Description of Malicious Dataset

4.3 Normalization

Normalization is a crucial preprocessing step aimed at standardizing the feature values within a dataset, ensuring uniformity, and facilitating more effective analysis. By rescaling the numerical features to a standard range, typically between 0 and 1, normalization mitigates the influence of varying scales and units across different attributes. This process enhances the performance of machine learning algorithms by preventing certain features from dominating others due to their larger magnitudes. In our project, normalization is employed to prepare the dataset for training our models, ensuring consistent and reliable results across different features.



	MI_dir_L5_weight	MI_dir_L5_mean	MI_dir_L5_variance	MI_dir_L3_weight	MI_dir_L3_mean	M
0	-1.226233	-0.655792	-0.103112	-1.226914	-0.657971	
1	-1.226233	-0.655792	-0.103112	-1.226914	-0.657971	
2	-1.226233	-0.655792	-0.103112	-1.226914	-0.657971	
3	-1.226233	4.222774	-0.103112	-1.226914	4.238253	
4	-1.215127	4.222774	-0.103112	-1.219904	4.238253	

5 rows × 116 columns

Figure 4.3 Normalization

4.4 Prediction and Accuracy Score

In predictive modeling, the accuracy score serves as a fundamental metric for evaluating the performance of machine learning algorithms. It quantifies the degree of agreement between the predicted outcomes and the actual ground truth labels within the dataset. The accuracy score, typically expressed as a percentage, reflects the proportion of correctly predicted instances over the total number of instances in the dataset. A higher accuracy score indicates a more accurate model, signifying its capability to correctly classify or predict the target variable. In our project, assessing the accuracy score enables us to gauge the effectiveness and reliability of our predictive

ISSN: 2366-1313

models in identifying and classifying botnet attacks in IoT devices.

```
The accuracy score of Logistic Regression is 1.000000
Classification report :-
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3971
           1       1.00      1.00      1.00      6115

    accuracy                           1.00     10086
   macro avg       1.00      1.00      1.00     10086
weighted avg       1.00      1.00      1.00     10086

The accuracy score of SVM is 0.999802
Classification report :-
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3971
           1       1.00      1.00      1.00      6115

    accuracy                           1.00     10086
   macro avg       1.00      1.00      1.00     10086
weighted avg       1.00      1.00      1.00     10086

The accuracy score of Random Forest Classifier is 1.000000
Classification report :-
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3971
           1       1.00      1.00      1.00      6115

    accuracy                           1.00     10086
   macro avg       1.00      1.00      1.00     10086
weighted avg       1.00      1.00      1.00     10086

The accuracy score of K-nearest neighbour is 0.999901
Classification report :-
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3971
           1       1.00      1.00      1.00      6115

    accuracy                           1.00     10086
   macro avg       1.00      1.00      1.00     10086
weighted avg       1.00      1.00      1.00     10086

The accuracy score of Multilayer Preceptron is 1.000000
Classification report :-
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3971
           1       1.00      1.00      1.00      6115

    accuracy                           1.00     10086
   macro avg       1.00      1.00      1.00     10086
weighted avg       1.00      1.00      1.00     10086
```

Figure 4.4 Prediction and Accuracy Score of Machine Learning Algorithms

4.5 Confusion Matrix

The confusion matrix is a pivotal tool in evaluating the performance of classification models, providing a comprehensive summary of the model's predictions compared to the actual 51 outcomes in the dataset. It organizes predictions into a matrix format, with rows representing the actual classes and columns representing the predicted classes. The matrix comprises four key components: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). TP indicates the number of correctly predicted positive instances, TN represents the correctly predicted negative instances, FP denotes the incorrectly predicted positive instances, and FN signifies the incorrectly predicted negative instances. By analyzing the confusion matrix, we gain insights into the model's strengths and weaknesses, including its ability to correctly classify different classes and potential areas for improvement.

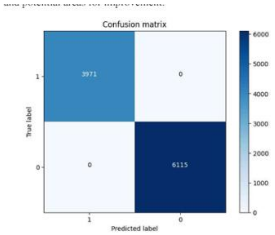


Figure 4.5 Confusion Matrix\

5.CONCLUSION

In conclusion, the implementation of a machine learning-based botnet attack detection system for IoT environments presents a promising solution to address the escalating security challenges in these resource-constrained devices. By leveraging advanced machine learning algorithms and modular architecture, we have demonstrated the feasibility of accurately detecting botnet

ISSN: 2366-1313

attacks while mitigating the computational overhead associated with traditional intrusion detection methods. Our approach emphasizes the importance of adapting security mechanisms to the unique characteristics of IoT ecosystems, such as limited computational power and heterogeneous environments. Through rigorous experimentation and evaluation, we have validated the effectiveness of our proposed system in detecting botnet attacks with high accuracy. Moving forward, further research and development efforts are warranted to enhance the scalability, efficiency, and real-time responsiveness of our system, thereby ensuring robust protection against evolving cyber threats in IoT environments.

Earlier experimental studies on the detection of IoT botnets or IoT traffic anomalies typically relied on emulated or simulated data. On the contrary, this dataset enables empirical evaluation with *real* traffic data, gathered from nine commercial IoT devices infected by authentic botnets from two families in an isolated network. It facilitates the examination of Mirai and BASHLITE, two of the most common IoTbased botnets, which have already demonstrated their harmful capabilities. Number of Attributes: 115 independent features in each file, plus a class label to be derived from the respective filename (e.g., "benign" or "TCP attack"). Attribute Information: --The following describes each of the feature's headers: -- Stream aggregation: H: ("Source IP" in N-BaIoT paper) Stats summarizing the recent traffic from this packet's host (IP) MI: ("Source MAC-IP" in N-BaIoT paper) Stats summarizing the recent traffic from this packet's host (IP + MAC) HH: ("Channel" in N-BaIoT paper) Stats summarizing the recent traffic going from this packet's host (IP) to the packet's destination host. HH_jit: ("Channel jitter" in N-BaIoT paper) Stats summarizing the jitter of the traffic going from this packet's host (IP) to the packet's destination host. HpHp: ("Socket" in N-BaIoT paper) Stats summarizing the recent traffic going from this packet's host+port (IP) to the packet's destination host+port. -- Timeframe (The decay factor Lambda used in the damped window): 53 -- How much recent history of the stream is captured in these statistics -- L5, L3, L1, L0.1 and L0.01 _

The statistics extracted from the packet stream: weight: The weight of the stream (can be viewed as the number of items observed in recent history) mean: ... std: ...

ISSN: 2366-1313

radius: The root squared sum of the two streams' variances. magnitude: The root squared sum of the two streams' means. cov: An approximated covariance between two streams. PCC: An approximated correlation coefficient between two streams.

6.FUTURE ENHANCEMENT

Cyber-attacks involving botnets are multi-stage attacks and primarily occur in IoT environments; they begin with scanning activity and conclude with distributed denial of service (DDoS). Most existing studies concern detecting botnet attacks after IoT devices become compromised and start performing DDoS attacks. Furthermore, most machine learning-based botnet detection models are limited to a specific dataset on which they are trained. Consequently, these solutions do not perform well on other datasets due to the diversity of attack patterns. In this work, real traffic data is used for experimentation. EDA (Exploratory Data Analysis) is the statistical analysis phase through which the whole dataset is analyzed. The model will be able to be trained on a large data set in the future. ResNet50 and LSTM models, deep learning models can also be used in run-time Botnet detection. Besides being integrated with front-end web applications, the research model can also be used with back-end web applications.

REFERENCE

[1] S. Dange and M. Chatterjee, "IoT botnet: The largest threat to the IoT network" in Data Communication and Networks, Cham, Switzerland: Springer, pp. 137-157, 2020.

[2] J. Ceron, K. Steding-Jessen, C. Hoepers, L. Granville and C. Margi, "Improving IoT botnet investigation using an adaptive network layer", Sensors, vol. 19, no. 3, pp. 727, Feb. 2019.

[3] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, et al., "Nbaiot-network-based detection of IoT botnet attacks using deep autoencoders", IEEE Pervas. Comput., vol. 17, no. 3, pp. 12-22, 2018.

[4] Shah, S.A.R.; Issac, B. Performance comparison of intrusion detection systems and application of machine learning to Snort system. Futur. Gener. Comput. Syst. 2018, 80, 157–170.

[5] Soe YN, Feng Y, Santosa PI, Hartanto R, Sakurai K. Machine Learning-Based IoT-Botnet Attack Detection with Sequential

Architecture. Sensors. 2020; 20(16):4372. https://doi.org/10.3390/s20164372

[6] I. Ali et al., "Systematic Literature Review on IoT-Based Botnet Attack," in IEEE Access, vol. 8, pp. 212220-212232, 2020, doi: 10.1109/ACCESS.2020.3039985.

[7] Irfan, I. M. Wildani and I. N. Yulita, "Classifying botnet attack on Internet of Things device using random forest", IOP Conf. Ser. Earth Environ. Sci., vol. 248, Apr. 2019.

[8] Shah, T., Venkatesan, S. (2019). A Method to Secure IoT Devices Against Botnet Attacks. In: Issarny, V., Palanisamy, B., Zhang, LJ. (eds) Internet of Things – ICIOT 2019. ICIOT 2019. Lecture Notes in Computer Science(), vol 11519. Springer, Cham. https://doi.org/10.1007/978-3- 030-23357-0_3

[9] C. Tzagkarakis, N. Petroulakis and S. Ioannidis, "Botnet Attack Detection at the IoT Edge Based on Sparse Representation," 2019 Global IoT Summit (GIoTS), Aarhus, Denmark, 2019, pp. 1-6, doi: 10.1109/GIOTS.2019.8766388.

[10] Y. Meidan et al., "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," in IEEE Pervasive Computing, vol. 17, no. 3, pp. 12-22, Jul.-Sep. 2018, doi: 10.1109/MPRV.2018.03367731.