Machine Learning Algorithms-based Prediction of Botnet Attack for IoT devices

Subba Reddy Borra[1], Akshaya[2], B. Swathi[2], B. Sraveena[2], B. Satya Sahithi[2]

[1,2]Department of Information Technology

[1,2]Malla Reddy Engineering College for Women (A), Maisammaguda, Medchal, Telangana.

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 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 attack. 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 neighbour, 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.

Keywords: Botnet attack, IoT, Machine learning.

1. Introduction

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 human or an object is an entity that can request or deliver a service [1]. The interconnection amongst the entities is always complex. IoT is broadly acceptable and implemented in various domains, such as healthcare, smart home, and agriculture. However, IoT has a resource constraint 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 IoT environment. Therefore, various kinds of attacks are possible due to the vulnerability of IoT devices.

IoT-based botnet attack is one of the most popular, spreads faster and create more impact than other attacks. In recent years, several works have been conducted to detect and avoid this kind of attacks [2]–[3] by using novel approaches. Hence, a plethora of relevant of relevant models, methods, and etc. have been introduced over the past few years, with quite a reasonable number of studies reported in the research domain.

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, the 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 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 anomaly-based 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. The 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 constraint devices. The experiment results indicate that the detection accuracy of our proposed system is high enough to detect the botnet attacks. Moreover, it can support the extension for detecting the new distinct kinds of attacks.

1.2. Challenging Issues

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 the traditional network devices. The heavyweight encryption methods cannot be deployed on these resource constraint devices. On the other side, the IoT devices become very cheap to set up for personal usages, like in small business and smart home appliances. The attackers were launching the attacks to 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, the new distinct kinds of attacks can be launched sometimes. Additionally, the complex processing of ML classifiers is a challenge to implement the lightweight attack detection system on the resource constraint devices.

1.3. Our Contributions

In this study, our main contributions are as follows.

(1) A botnet attacks detection framework with sequential architecture based on machine learning (ML) algorithms is proposed for dealing with attacks in IoT environments.

(2) A correlated-feature selection approach is adopted for reducing the irrelevant features, which makes the system lightweight.

(3) In our proposal, classifiers based on different ML algorithms may be applied in different attack detection sub-engines, which leads to better detection performance and shorter processing times and a lightweight implementation.

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 also can be extended with corresponding sub-engines for new kinds of attacks.

Ali et al. [6] outlined the existing proposed contributions, datasets utilised, network forensic methods utilised 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, contain a malware or not. In this research, this work under sample the dataset because the datasets contain 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 are from 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 capture of IoT devices in a large scale. Login puzzle is a variant of client puzzle, which presented a puzzle to the remote device during the login process to prevent unrestricted log-in 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 log-in attempt increases the complexity of solving the login puzzle for the next attempt. This paper introduced a novel mechanism to change the complexity of puzzle after every unsuccessful login attempt. If each IoT device had used login puzzle, Mirai attack would have required almost two months to acquire devices, while it acquired them in 20 h.

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 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 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 proposed methods ability 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 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. Afterwards, 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-the-art 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 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 Gaussian distribution used in the population initialization. Furthermore, the local search mechanism was followed by the Gaussian density function and local-global best function to achieve better exploration during each generation. Enhanced BA was further employed for neural network hyperparameter tuning and weight optimization to classify ten different botnet attacks with an additional one 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 deep-learning artificial neural network model can accurately and efficiently identify botnets.

3. Proposed System

3.1 Pre-processing

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task.

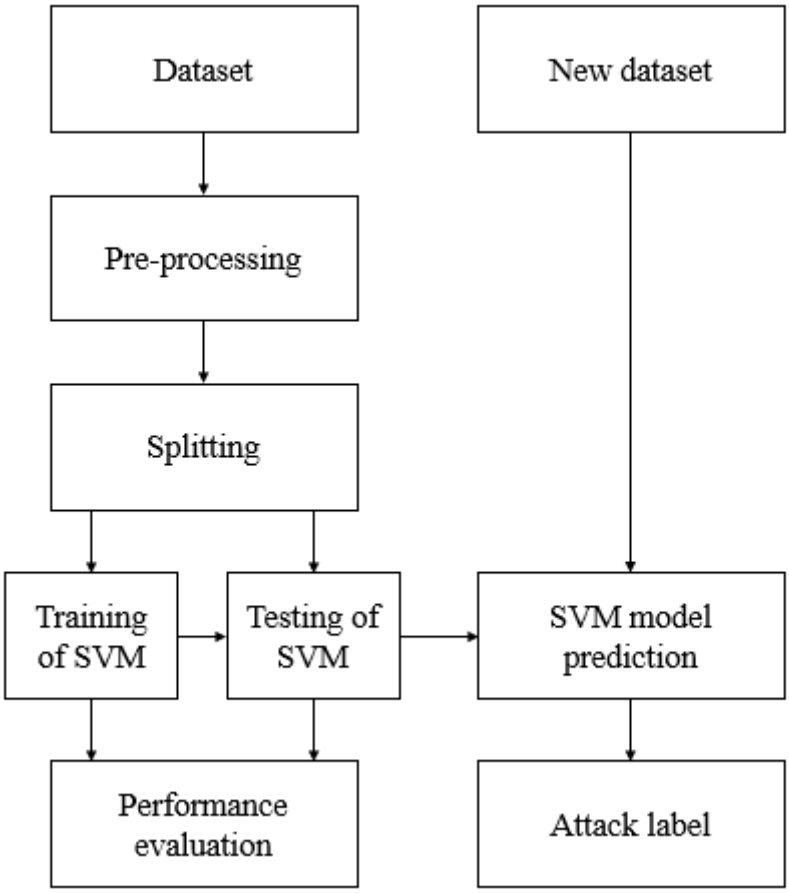


Fig. 1: Block diagram of proposed system.

Why do we need Data Pre-processing?

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

3.1.1 Splitting the Dataset into the Training set and Test set

In machine learning data pre-processing, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model.

Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models.

If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

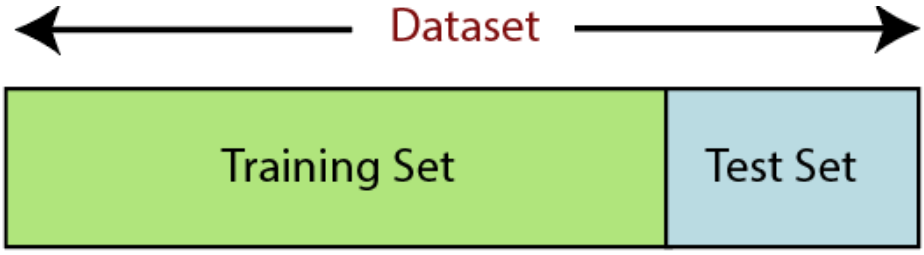


Fig. 2: Dataset splitting.

Training Set: A subset of dataset to train the machine learning model, and we already know the output.

Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

3.2 Support Vector Machine Algorithm

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 as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

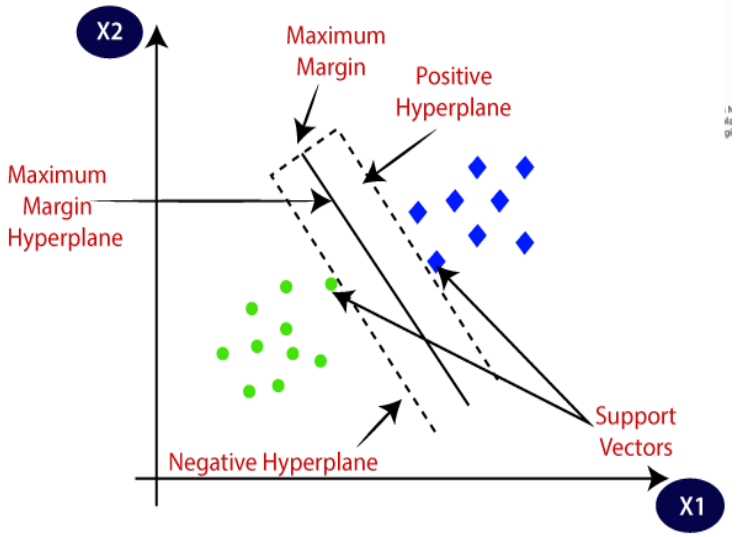


Fig. 3: Analysis of SVM.

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, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM 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 test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:

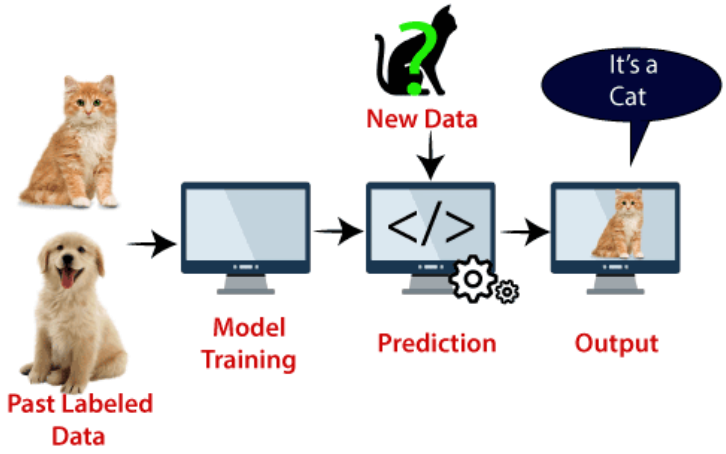


Fig. 4: Basic classification using SVM.

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 non-linear data and classifier used is called as Non-linear SVM classifier.

3.3 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:

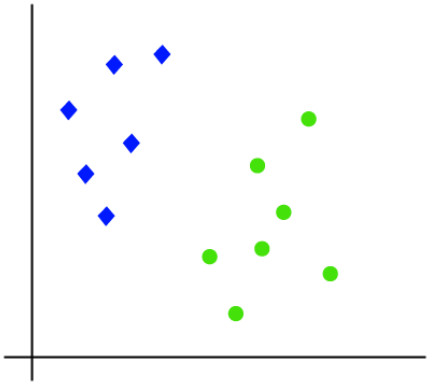


Fig. 5: Linear SVM.

So as it is 2-d space so 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:

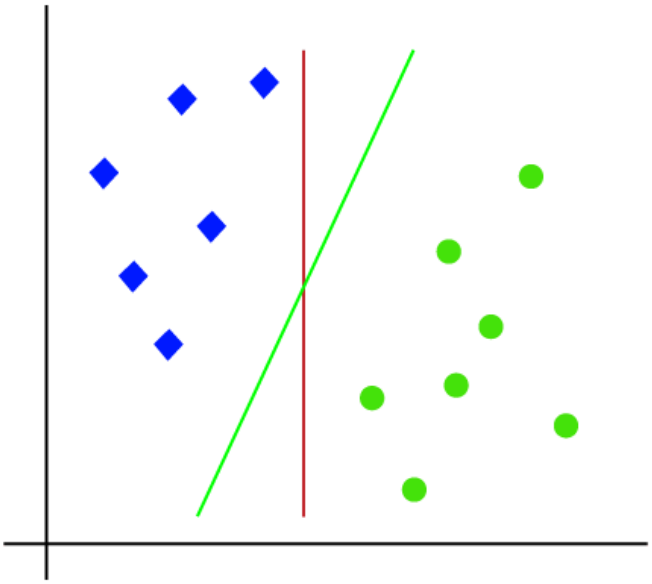


Fig. 6: Test-Vector in SVM.

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

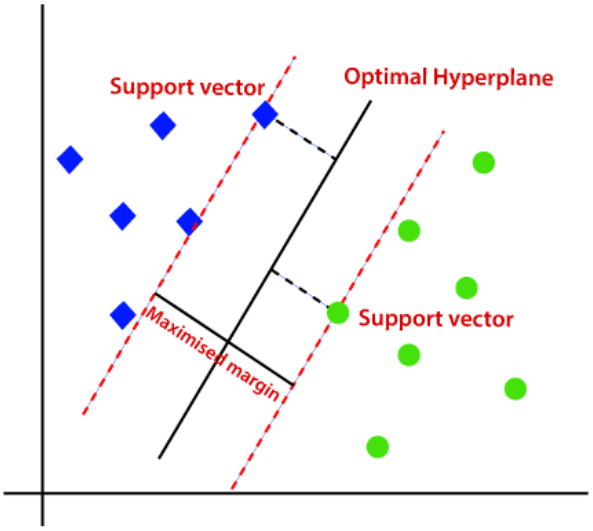


Fig. 7: Classification in SVM.

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:

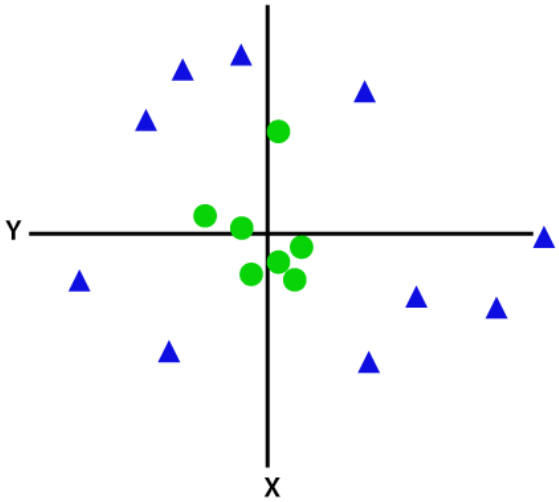


Fig. 8: Non-Linear SVM.

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:

$$z=x^2+y^2$$

By adding the third dimension, the sample space will become as below image:

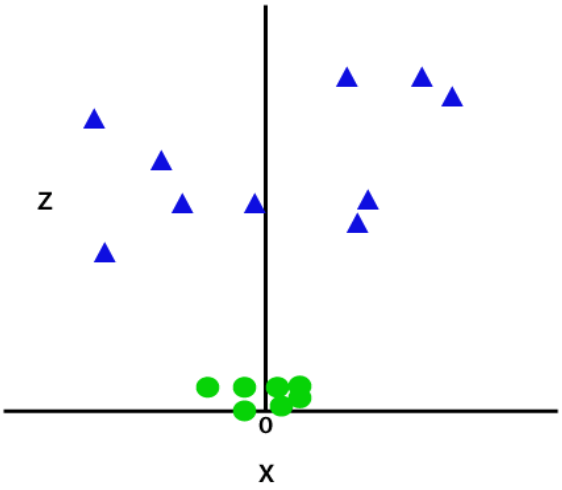


Fig. 9: Non-Linear SVM data separation.

So now, SVM will divide the datasets into classes in the following way. Consider the below image:

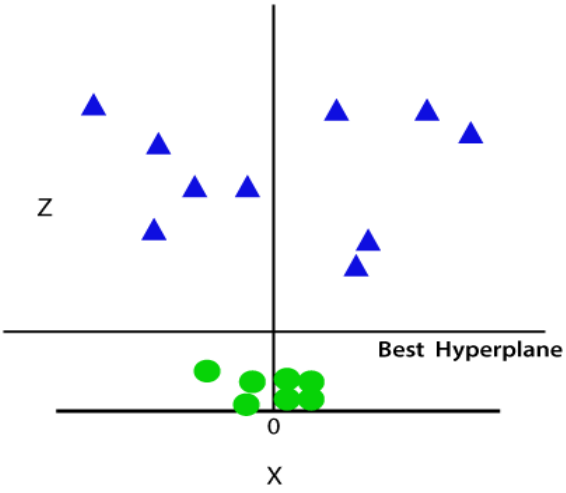


Fig. 10: Non-Linear SVM best hyperplane.

Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:

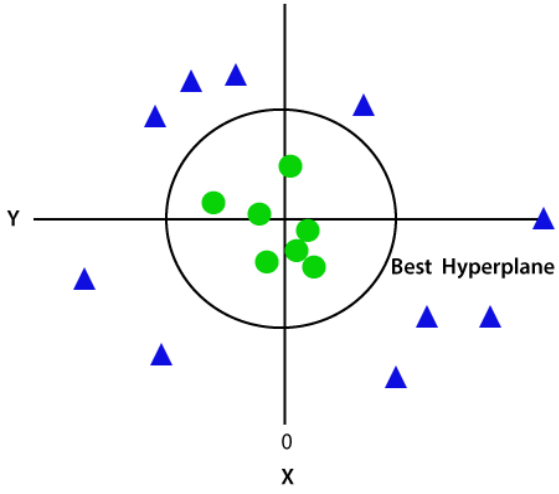


Fig. 11: Non-Linear SVM with ROC.

Hence, we get a circumference of radius 1 in case of non-linear data.

3.4 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.

4. Results and Discussion

Earlier experimental studies on the detection of IoT botnets or IoT traffic anomalies typically relied on emulated or simulated data. In 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 IoT-based 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):

-- How much recent history of the stream is capture 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: ...

radius: The root squared sum of the two streams' variances.

magnitude: The root squared sum of the two streams' means.

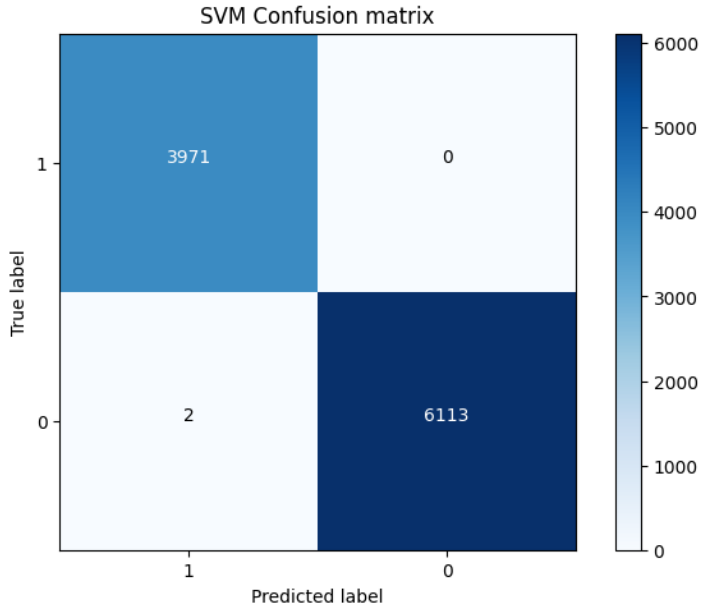
cov: An approximated covariance between two streams.

pcc: An approximated correlation coefficient between two streams.

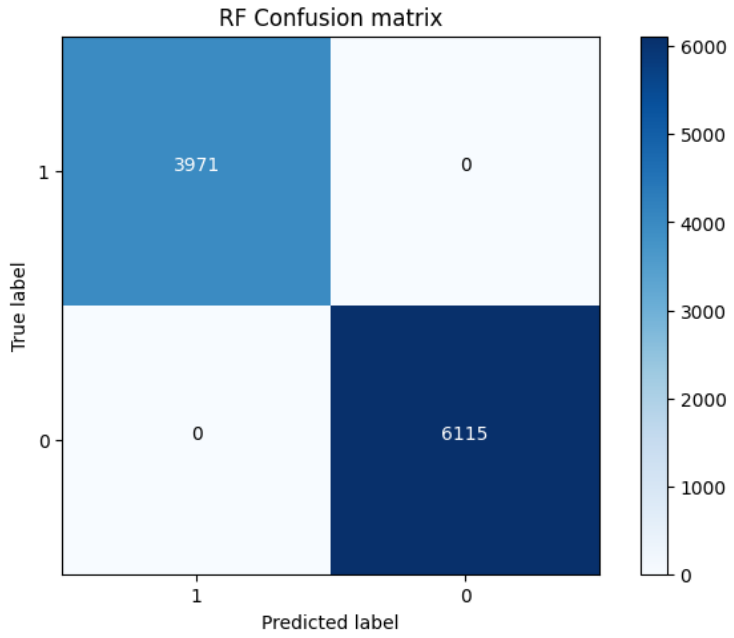
Non-malicious Dataset

	MI_dir_L5_weight	MI_dir_L5_mean	MI_dir_L5_variance	MI_dir_L3_weight	MI_dir_L3_mean	MI_dir_L3_variance	MI_dir_L1_weight	MI_dir_L1_mean	MI_dir_L1_
0	1.000000	60.0	0.0	1.000000	60.0	0.0	1.000000	60.0	
1	1.000000	60.0	0.0	1.000000	60.0	0.0	1.000000	60.0	
2	1.000000	60.0	0.0	1.000000	60.0	0.0	1.000000	60.0	
3	1.000000	590.0	0.0	1.000000	590.0	0.0	1.000000	590.0	
4	1.927179	590.0	0.0	1.955648	590.0	0.0	1.984992	590.0	

5 rows × 115 columns



	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



	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

5. Conclusion and Future Scope

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.

References

[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., "N-baiot-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.

[11] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh and O. Jogunola, "Federated Deep Learning for Zero-Day Botnet Attack Detection in IoT-Edge Devices," in IEEE Internet of Things Journal, vol. 9, no. 5, pp. 3930-3944, 1 March1, 2022, doi: 10.1109/JIOT.2021.3100755.

[12] F. Hussain et al., "A Two-Fold Machine Learning Approach to Prevent and Detect IoT Botnet Attacks," in IEEE Access, vol. 9, pp. 163412-163430, 2021, doi: 10.1109/ACCESS.2021.3131014.

[13] Abu Al-Haija Q, Al-Dala'ien M. ELBA-IoT: An Ensemble Learning Model for Botnet Attack Detection in IoT Networks. Journal of Sensor and Actuator Networks. 2022; 11(1):18. https://doi.org/10.3390/jsan11010018

[14] Alharbi A, Alosaimi W, Alyami H, Rauf HT, Damaševičius R. Botnet Attack Detection Using Local Global Best Bat Algorithm for Industrial Internet of Things. Electronics. 2021; 10(11):1341. https://doi.org/10.3390/electronics10111341

[15] Ahmed, A.A., Jabbar, W.A., Sadiq, A.S. et al. Deep learning-based classification model for botnet attack detection. J Ambient Intell Human Comput 13, 3457–3466 (2022). https://doi.org/10.1007/s12652-020-01848-9