OPEN ACCESS
Vietnam Journal of Computer Science
Vol. 8, No. 2 (2021)
© The Author(s)
DOI: 10.1142/S2196888821500123

IoT Botnet Detection Using Various One-Class Classifiers

Mehedi Hasan Raj*, A. N. M. Asifur Rahman†, Umma Habiba Akter‡, Khayrun Nahar Riya§, Anika Tasneem Nijhum¶ and Rashedur M. Rahman‖

Department of Electrical and Computer Engineering
North South University, Plot 15, Block-B
Bashundhara, Dhaka 1229, Bangladesh
**mehedihasanraj007@gmail.com*
†*anm.rahman@northsouth.edu*
‡*habibaumme13@gmail.com*
§*khayrun.nahar@northsouth.edu*
¶*anika.nijhum@northsouth.edu*
‖*rashedur.rahman@northsouth.edu*

Received 18 June 2020
Accepted 17 September 2020
Published 5 November 2020

Nowadays, the Internet of Things (IoT) is a common word for the people because of its increasing number of users. Statistical results show that the users of IoT devices are dramatically increasing, and in the future, it will be to an ever-increasing extent. Because of the increasing number of users, security experts are now concerned about its security. In this research, we would like to improve the security system of IoT devices, particularly in IoT botnet, by applying various machine learning (ML) techniques. In this paper, we have set up an approach to detect botnet of IoT devices using three one-class classifier ML algorithms. The algorithms are: one-class support vector machine (OCSVM), elliptic envelope (EE), and local outlier factor (LOF). Our method is a network flow-based botnet detection technique, and we use the input packet, protocol, source port, destination port, and time as features of our algorithms. After a number of preprocessing steps, we feed the preprocessed data to our algorithms that can achieve a good precision score that is approximately 77–99%. The one-class SVM achieves the best accuracy score, approximately 99% in every dataset, and EE's accuracy score varies from 91% to 98%; however, the LOF factor achieves lowest accuracy score that is from 77% to 99%. Our algorithms are cost-effective and provide good accuracy in short execution time.

Keywords: Botnet; IoT botnet; cybersecurity; DDOS attack; one-class classifier; support vector machine; elliptic envelope; local outlier factor.

‖Corresponding author.

1. Introduction

IoT (Internet of Things) relays to those millions of physical devices that have been linked to the internet and have been gathering various data related to IoT device operation. The IoT technology mass is a range of expertise, values, and requests, which lead from the simple construction of things to the internet to the smoothest and most composite applications that use these connected things, the data they collect and interconnect, and the different stages needed toward rule these applications. Most of the objects of IoT are observing and sensory devices that are used to propagate the continuous exhibition of the physical world phenomenon to the digital world in the system of a data stream. The IoT network is always busy for communicating data, data transmission, and modified user requests. Constructing very smart architecture, IoT supports many things like a chain network.[1] This technology is widely used in business, academia, government, and military systems. It has been measured as the third wave of information technology after the internet and mobile communication network.[2] IoT devices have seen robust development over the last years and will remain to rise their existence significantly rapidly.

It is valued that there will be more than 50 billion devices joined to the internet by the end of 2021.[14] With the development of computer network technology, different types of attacks also emerged. Previously for taking credit, attacks were launched, but now the situation is different. Nowadays, attackers oblige money to break a violent profitable place. Allowing to this shifting of the attack hypothesis, with the development of new technology, a new type of attack will be possible. In our paper, we work on the botnet attack as this is one of the most powerful attacks.

When we hear about the botnet attack, at first, it comes in our mind about cyber-attacks. First, the botnet is some internet-connected devices, each of which runs one or more bots. Botnets can be used to conduct distributed denial-of-service (DDoS) attacks, steal data, send spam, and allow the system and its connection to the attacker.[4,5] An internet-connected device (Bot) becomes infested when Botnet attacks occur.[6] It is sometimes mentioned as computer worms or zombie armies, and their owners are called botmasters or bot herders. The creation of insecure IoT devices has stemmed from a surge of IoT botnet attacks on internet substructure.

For preventing IoT botnets attacks, new techniques are invented to identify and block attack traffic from IoT botnets. For identifying malicious internet traffic, machine learning (ML) algorithms are widely used. Selecting features is an essential part of creating ML models. But IoT traffic is different from that of other internet-connected devices. For example, IoT devices frequently communicate with a small finite set of endpoints rather than a large variety of web servers. IoT devices are also more likely to have repetitive network traffic patterns, such as regular network pings with small packets at fixed time intervals for logging purposes.

In our paper, we concentrate on botnet detection. We use a network-based detection method using one-class classifiers. One-class classifiers are a particular form of ML technique that, rather than classifying an instance into one of the several predefined patterns, models a single trend and uses it to determine whether or not a new situation belongs to the pattern. In the case of cybersecurity, by detecting botnet, it can also detect malicious activities in the network, which help to improve security. Compared with other techniques, one-class classifier works faster and detects botnet faster.

This paper is structured as follows. We present prior works on IoT attacks in Sec. 2. Section 3 provides context information about conventional ML and one-class classification (OCC). Dataset description is given in Sec. 4. Section 5 describes the experimental framework. Section 6 presents the result and discussion and finally Sec. 7 concludes and gives direction for future research.

2. Related Works

Some recent works for botnet detection are based on host-based data that allow the presence of bot malware using data that are collected from resource consumption of the device. A botnet is a network of a private computer that is contaminated with malware and operated as a community without the awareness of the users. IoT detection system (IoTDS) is a common term of IoT for a host-based botnet detection technique. The IoTDS is mainly divided into two stages: model induction and continuous evaluation. For each IoT system, those two stages are implemented. There are two elements of network configuration: the administrator console that is installed on the IoT gadget and accountable for processing the data of an application, and the control console that is mounted on a different computer server where generally warning and situation summary are processed.[7]

Botnets are widely known threats that exploit weaknesses in IoT systems to infect several systems and execute orchestrated attacks. New techniques for the identification of IoT botnets are required to counter this. Several works are done by collecting data from an artificial environment which is made by the researchers. Some researchers also tried to evaluate their work in some real scenarios. In a managed network, the predictive efficiency and resource usage of the proposed solution were tested utilizing three separate legal settings and seven IoT botnets. The findings report that the device being suggested is successful in detecting different botnets with low resource consumption.[8]

The bulk of IoT intrusion detection research is focussed on virtual networks utilizing the 6LoWPAN protocol. Contiki and TinyOS are the two most popular operating systems implemented on those 6LoWPAN-based computers. These experiments have encouraging outcomes by investigating hybrid IDS and looking at the network-based and finally host-based system. Reliance on the 6LoWPAN network hinders the deployment in networks that is based on various IoT protocols like TCP/IP stack as well as Bluetooth Low Energy.[9]

Other IoT researches rely on traditional TCP/IP stack and domestic IoT applications for intrusion detection. A few of these functions are intrusion detection systems (IDSs) that are purposeful to be gateway nodes or router and use whitelists and various ML techniques or predefined guidelines. The work on IDS called Heimdall is introduced. Preventing the malicious URLs of linking IoT devices, the IDS applies a whitelist, interactions with C and C (Command and Control) botnets, or leakage of the private data. The intention of Heimdall is to function as a node in the local router, serving for IoT applications like a conduit, and utilizing the third-party revise of various malicious or anomaly addresses by institutions like VirScan. Another IDS name is Metadefender and many of them are in tandem DNS confirmation. Those systems have limited overhead and are successful against the attacks produced by the researchers. It is not tested in the actual botnet; however, the IDS is based on the third-party device maintenance.[10]

In Ref. 3, a network-based method is proposed for IoT botnet detection. The work mainly focussed on legitimate behaviors of IoT devices and the authors used deep autoencoders to detect IoT botnets for malicious network traffic. For building the model, they used normal traffic without labeled data. By using deep autoencoders, they got good success rate with a low false rate. They tested this method on nine IoT devices by using two famous known IoT botnets: Mirai and BashLite. IoT devices performed simple repeated and systematic tasks. Though the reported accuracy is higher, their method needs huge data and becomes very computationally intensive when deep autoencoders are used.

In Ref. 14, the authors worked on DDoS detection in IoT network traffic. For network behaviors, the authors have used a limited variety of endpoints and standard time intervals between packets. Botnets such as Mirai have used vulnerable consumer IoT devices to target critical internet infrastructure with DDoS. This motivates the development of the latest techniques for the automated detection of user IoT attack traffic. The authors developed an ML pipeline that performed data collection, extraction of features, and binary classification for DDoS detection of IoT traffic. The features are designed to take advantage of IoT-specific network behaviors, whereas conjointly leveraging network flow characteristics such as packet length, inter-packet intervals, and protocol are used. The authors compared five classifiers for attack detection, including K-nearest neighbors, random forests, decision trees, support vector machines (SVMs), and neural networks. They developed training data on classifiers by simulating a network of consumer IoT devices. K-nearest neighbors, random forest, and neural network classifiers were significantly effective and successfully detected attack traffic. The paper suggests that home gateway routers or any kind of different network middle boxes may automatically discover local IoT device sources of DDoS attacks using some common ML algorithms.

In Ref. 21, analyzing network traffic behavior, a new method was proposed for characterizing and detecting botnets. The main focus was on detecting botnets before they could launch attack. The main goal of the paper was to find P2P bots that were represented as the newest and challenging type of botnet. The detection of the botnet command-and-control (C&C) phase is very critical as it enables the identification of bots that travel under the radar by websites, ad-hoc wireless network, file-sharing networks, malicious emails, and before these bots strike or accomplish their objectives. They have used their model for online botnet detection and used five types of ML technologies. During the botnet C&C process, they have used their model on datasets and successfully detected botnets.

The authors in Ref. 22 introduced a new method to detect unauthorized network intrusions based on traffic flow data and Cisco NetFlow protocol application. The method could detect common network attacks and also could provide a list of trespasser's IP addresses and help to lock them. For this work, the authors collected data by emulating the attempted DDoS attacks. They structured the algorithm based on the sizes of the flow and the duration of the flow. The first step of the algorithm was to find the size of the flow where shorter flows, for example, up to 50 bytes were considered as port scanning. In the second step, they found out the IP addresses which had larger duration of the flow and very long streams, for example, lasting for more than 5 min, might be considered as the way of carrying out DDoS attack. They implemented their algorithm as a script in Perl and installed it on a protected server. The output of the script was a list of suspicious IP addresses which could be blocked for some time by using firewall iptables. Basically, in their model, the authors analyzed the data and found out two key attributes to distinguish attack and normal flow. But being static and dependent on these attributes, the model might provide errors on detecting some other attacks.

For various issues we found in the previous works, for instance, training is computationally very expensive for deep autoencoders. Some other techniques were developed to detect only DDoS attack by a binary class classifier that needs very large labeled data for both attack and non-attack traffic. The previous works did not use one-class classifier algorithms on network flow traffic to detect IoT botnets and did not report any comparative study on those classifiers. To consider these issues, we propose a method that is easy to implement with very low computation cost. Besides, we do not need a huge dataset to train the model, and there is no need to label the malicious traffic data. We also use different one-class classifier algorithms, compare these algorithms, and finally, we suggest the best algorithm that can detect different botnets in various scenarios.

3. One-Class Classifiers

OCC is a classical ML technique for modeling the class trying to detect objects of a certain class and other objects out of this class, which will be distinguished. It is known as unary classification and also a difficult one. OCC mainly does not make a model for all the classes that can be found from a dataset, rather it deals with one class which is different from other classes and distinguishes these classes from the focussed one. The OCC algorithm can be a substitution of binary classification. In binary classification, the labeled dataset can be divided into two different classes according to their features. In one-class classifier, a training dataset can be one class and then identify the holdout test dataset. In supervised ML approaches, the dataset needs to be labeled. Sometimes, it consumes high cost and huge resources to label data representing different classes. For example, collecting labeled data for botnet detection requires a special environment combining vulnerable devices. OCC does not require label data and can be trained with one type of data. For botnet detection algorithms can be trained with usual data and while testing, it can detect whether it is similar to the legitimate data or not. Descriptions of some one-class classifiers that are used in this research are presented below.

3.1. *Elliptic envelope*

One-class classifier has only one specific class for training the data, whereas for specific data, the ML model makes a layer. There are two types of layers. The inside layer is called inlier and the outside layer is called an outlier. The polluted things mainly lie on the outlier, whereas normal or not polluted data lie in the inlier. Our main goal is to separate the regular observation from some unnecessary or pollutions that are called the outlier. There are many ways to detect the outlier but one common way to do this is to consider regular data that come from a well-known distribution. Data generally follow the Gaussian distribution. From this consideration, the shape of the data is tried to find out. The shape generally looks like an ellipse. Elliptic envelope (EE) is an object that is used for detecting outlier from Gaussian distributed data. There are many parameters such as precision that specify whether the estimated precision is stored or not; support-fraction defines the quantity of contamination for the dataset. Also, EE has various attributes that are useful for making a good model and they are locations that estimate the robustness and shape is an n-dimensional array. Covariance defines a robust covariance matrix that defines location, precision, support, and so on.

In our work, the EE works by creating only one layer. We trained the algorithm with only one class which is normal traffic data. We considered the data followed by the Gaussian distribution and sorted the outlier. To find out the shape of the ellipse and the size of it, the most common behavioral data are considered and the fast-minimum covariance determinant (FAST-MCD) method is used to analyze the behaviors of different features. While fitting the ellipse, the method uses the Mahalanobis distance that works by calculating the distance between P and D, where P is a point and D is a distribution. This algorithm uses contamination hyperparameter and analyzes the percentage of observances to separate the outliers.[11]

3.2. *Local outlier factor*

Local outlier factor (LOF) works for unsupervised anomaly detection. It is an anomaly detection tool. This algorithm compares the density of a given point and the density of its neighbor. To search the nearest neighbor, the LOF concept is used. It is not used for decision function or prediction of objects. It calculates the local density deviation of a given point with respect to its neighbor point. LOF is used for outlier detection. There are some steps for detection and the steps are described below:

- Find out the kth nearest neighbors.
- Calculate the distance between the point in consideration and kth nearest neighbor.
- Local reachability density (LRD) is the estimated distance in any direction from a given point to its neighbor point. It calculates the number of items within k-nearest neighbor and then divides this sum of items by the max of the kth nearest neighbor of the point and the distance between the point and its adjoining point.
- Finally, calculate the local outlier of each point. For each point, at first, calculate the sum of all LRDs and multiply it by the sum of reachable distances. Then divide it by the square of the number of the items within the k-nearest neighbor set. If LOF value is less than 1 then it is a good indicator of inlier. If LOF value is greater than 1, then it is an indicator of an outlier.

In this work, the LOF is also trained with normal traffic data which works by comparing the density of a given point and the density of its neighbors. LOF observes the data points by calculating the k-NN distance, LRD, and then the local outlier of each point. Finally, if the LOF value is -1, then it indicates that it is an outlier.[16]

3.3. *One-class SVM*

An SVM is used for supervised machine learning models for analyzing data and recognizing patterns. SVM is used for both classification problems and regression problems. Generally, there are two classes of examples that are given to the SVM algorithm; however, a one-class SVM (OCSVM) has some differences from normal SVM. One-class SVM is trained by only one class which is normally “normal” class. It is unsupervised learning, unlike SVM. It makes a property for normal cases. After that, it tries to make a prediction, example of which is not like the normal examples. This algorithm is very much useful for anomaly detection. There are various advantages of this classifier in high dimensional spaces, and it is good when the dimension number is greater than the number of samples. However, it does not work better when it has a huge number of features, and the sample size is small. OCSVM has many parameters including the kernel, degree, gamma, coef0, tol, and so on. Also, there are many attributes for this, including support vector, dual_coef_, intercept_, offset_, and so on. The most important thing is that it returns two types of integers: 1 for inliers and another is -1 that defines outliers.[12]

The LOF, EE, and OCSVM are used as one-class classifiers. One-class classifier algorithms take one class data for training and create an outlier based on data features. If test data satisfies normal traffic, the output will be 1, otherwise -1. Although all of these three algorithms are one-class classifiers, it is clear that they are different from each other based on their data observation patterns. The EE works by creating a layer. The inlier or, in this project, the non-attacked data lies inside the layer. The outlier or the attacked data lies on the outside of the layer. LOF works by comparing the density of a given point and the density of its neighbors. LOF observes the data points by calculating the k-NN distance, LRD, and then the local outlier of each point. In contrast, the OSVM in our work follows the one-class SVM hyperplane, which is conceived as a restricted question of optimization. We need to process the data in kernel functions, which are configured by a hyper-parameter with the usage of SVM. The kernel functions take nonlinearly separable data and indirectly move it to a higher dimensional plane that can be segregated linearly.

OCSVM has various advantages in high dimensional spaces. EE features form a multi-dimensional space. LOF works better in low-dimensional dataset.

4. Dataset

We collected "IOT Security Dataset" from http://www.uel.br/grupo-pesquisa/secmq/dataset-iot-security.html.

4.1. *Device profile*

In the dataset, there are three typical types of IoT devices which were implemented in a Raspberry Pi in Ref. 13. The profiles of the devices are as follows.

4.1.1. *Multimedia center (MC)*

MC is used for entertainment purposes. By this device, regular television turns into Smart TV. It consumes tv programs. It also has two other functions. It can install other programs in this device and can be updated automatically by itself. It consumes huge resources by video streaming. This device is normally found in our house.

4.1.2. *Surveillance camera with additional traffic (ST)*

The surveillance camera with additional traffic is used for security purpose. This device does not only transmit video streams to a user, but also configure a website. It can also present traffic from other protocols. It does not consume huge resources. It is generally found in home, office, or secured places.

4.1.3. *Surveillance camera (SC)*

Like the surveillance camera with additional traffic (ST) profile, this device is used for the security purpose. But there is a difference with the ST profile. It cannot configure a website. It only transmits the videos to the user. It consumes fewer resources than ST profile. It is found in many places other than ST.

4.2. *Network scenario*

In this section, we are going to discuss how the authors in Ref. 13 created the network environment to support the devices profiles.

(1) A switch with Ethernet and Wi-Fi interfaces.
(2) There are four computers which are defined as (1) Machine 1, (2) Machine 2, (3) Machine 3, and (4) Gateway which is connected to the switch in the environment via Ethernet.
(3) A Raspberry Pi (3B) connected to the switch through Wi-Fi.

In Fig. 1, it describes how the network environment is used to originate the dataset.[13] The Raspberry Pi (3B) was used for running the MC, ST, SC. Machine 1 was responsible for hosting a DNS server for the Mirai botnet, a VLC client for SC and ST profiles for consuming the video streaming. It was also responsible for hosting a VLC server for the MC profile to generate video streaming. To get access in the Internet and Dynamic Host Configuration Protocol service to the network, Gateway was used in the environment. Machine 2 was responsible for hosting the Web Server as well as the Device Admin. Machine 3 was liable for infecting the MS, SC, ST with botnet samples on the network. The Device Admin emulated the transactions of associate administration software system accustomed management and came upon the emulated camera within the ST profile. The Web Server was used for providing a static Apache website that was accessed by the Raspberry Pi employing script MC, ST, SC profiles to emulate the consumption of web service in the network.

4.3. *Dataset creation*

There are two types of logs provided in Ref. 13; one is containing only legitimate activities and the other is about both legitimate and malicious activities. In the beginning, each profile MC, ST, SC was executed for one hour as usual. The goal was to collect data exclusively related to legitimate activities. In ST and SC profiles, the Raspberry Pi transmitted a video stream using VLC over HTTP through the port

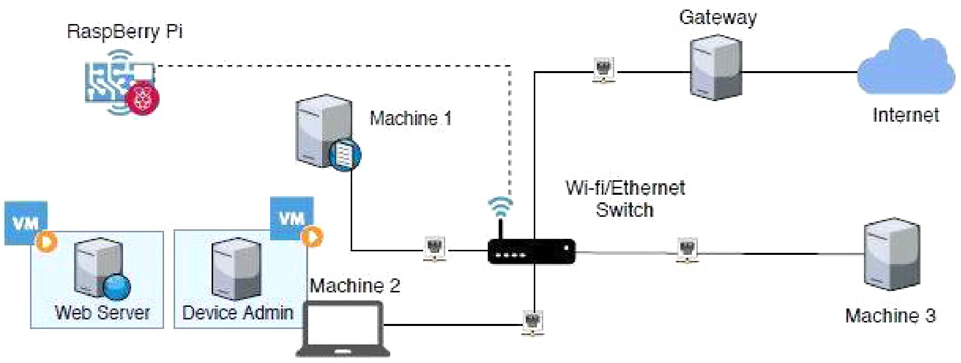

Fig. 1. Network topology of the experimental environment.[13]

Table 1. Description of the experimental environment components.[13]

Component	Function	Specs	Operating system	Virtual/real
Raspberry Pi	Mimic different types of IoT devices	Ram: 1 GB CPU: ARM 1.2 GHz	Rasbian Stretch	Real
Machine 1	DNS server and video consumer	RAM: 4 GB CPU: Intel i5 3.2 GHz	Ubuntu 17.04	
Gateway	Provide internet access and routing			
Machine 3	Deliver and control malware	RAM: 8 GB CPU: Intel Xeon 3.1 GHz		
Device Admin	Makes SSH and Telnet connections with the Raspberry Pi and updates	RAM: 1 GB CPU: 1 GHz		Virtual
Web Server	Host a static web page used for updates in the device			

Table 2. BotNets and number of samples.

Infection type	Profile	Method of infection	Botnet(s)	Total samples
1	MC	SSH	BashLite	11
	SC		Aidra	
	ST		Hajime	4
2	MC		Mirai	1
	SC			
	ST			
3	MC		Mirai, Doflo, Tsunami, Wroba	28

8080 for 20 min, with a total size of 4 MB to Machine 1. The Raspberry Pi consumed videos from Switch 11 and YouTube in the MC.

After that, data are captured on combined legitimate and malicious behavior. Three types of infection data were organized: Type 1, Type 2, and Type 3. Table 2 presents the botnets, method of infection, and the number of samples running in each profile.

In the Type 1 infection, three botnets were used: Bashlite, Aidra, and Hajime. Machine 3 was connected to the device via SSH using Rasbian default credentials (Raspberry: Pi) to infect the Raspberry Pi with these malwares. Then, malware samples were transferred to the device through the established connection.

Type 2 infection was on Mirai botnet. It created a complete scenario with the malware and the C& C server in the environment.

In the Type 3 infection, Mirai, Doflo, Tsunami, and Wroba were selected. The authors in Ref. 13 also included the Mirai botnet to check whether the most popular

Table 3. Composition of dataset.

Infected type	Profile	Malicious activity
No infection	MC L SC L ST L	No
Type 1	MC I1 SC I1 ST I1	Both malicious and legitimate
Type 2	MC I2 SC I2 ST I2	
Type 3	MC I3 SC I3 ST I3	

IoT botnet would dominate or be dominated by its competitors. The infection process was carried out via SSH connection. Each type of infection was executed in each profile for one hour.

Lastly, NetFlow files were created from all the captured network packets where all the network traffic was collected from the Raspberry Pi. After that, all flows composed of packets that were captured during one hour of execution without infection were labeled as legitimate.

4.4. *Dataset file description*

The suffix "L" refers to the data which were not infected and the suffix "I" which included legitimate and malicious activities (Table 3).[13] There are three files: the host data collected (the number of tasks, CPU and memory consumption, electric potential difference and CPU temperature). The second is IP flow data labeled as malicious or legitimate. The third one which captured the Raspberry Pi is a PCAP file with network packets.

5. Methodology

In this section, we discuss our working methodology to identify botnets in IoT devices by different one-class classifiers. IoT devices are the most utilized devices these days in the world. IoT devices run simple straightforward work, accomplish characterized work and sometimes do tedious tasks as well. Their practices are simply to accomplish a similar fill in as long as no malicious activities are running on it. We have divided the experiment into the following stages:

- Data preprocessing;
- Training various ML algorithms with the data;
- Evaluation of the model with difference performance parameters.

Table 4. Data analysis.

Descriptions	Time duration (attack), ns	Time duration (normal), ns	Input packet size (normal), byte	Input packet size (attack), byte
Mean	2.882497e+02	5.336968e+04	4.117417e+05	1.604116e+02
Standard deviation (std)	7.451682e+03	2.358861e+05	3.280656e+06	3.696880e+04
25%	0.000756e+00	1.332000e+00	7.750000e+01	4.000000e+01
50%	0.000856e+00	7.100000e+01	4.200000e+02	4.000000e+01
75%	0.000654e+00	1.081150e+04	3.689500e+03	4.000000e+01
Max	2.841553e+06	3.064916e+06	7.923012e+07	2.024552e+07

5.1. *Data preprocessing*

The preprocessing phase of the data has several steps that are required for our ML model to perform better. Our preprocessing phase includes the selection of features, managing categorical features, and data normalization. Those are discussed in detail below.

5.1.1. *Features selection*

The dataset contains two classes of attributes, and we pick a portion of the features among them. The features are important in differentiating the normal and botnet traffic for IoT devices. A few examples can be useful for better learning of our model. We have utilized a case of an MC to understand our features better.

5.1.2. *Time duration (Δt)*

Normal IoT traffic has certain time-limit limitations.[14] IoT gadgets send packets at regular time intervals while in the attack (e.g. botnet), and IoT devices send packets frequently around zero nanoseconds so the most extreme number of requests can be sent inside a timeframe. Mean value, MCI3 data from the dataset, of time length was roughly 2.882497e+02 ns and 5.336968e+04 ns for normal and malicious traffic (Table 4). In correlation, normal traffic periods are a lot higher than attack IoT traffic, and the majority of the attack has happened inside zero nanoseconds (Table 4). Along these lines, time duration (Δt) can be a significant feature for separating the normal and botnet IoT traffic. The general view among normal and botnet IoT traffic is in Fig. 8.

5.1.3. *Input packet (ibyte)*

There are critical contrasts in the data packet size among normal and botnet IoT traffic. In normal traffic, the average input packet size is 4.117417e+05 byte where least and greatest size changes from 3.200000e+01 byte to 7.923012e+07 byte (Table 4) because of video streaming data. Seventy-five percent of input packet size of our dataset is 3.689500e+03 byte (Table 4). In any case, a botnet IoT traffic of data packet size is not the same as would be expected IoT traffic. A large portion of the botnet traffic has a small packet in size so it can attempt to open as many

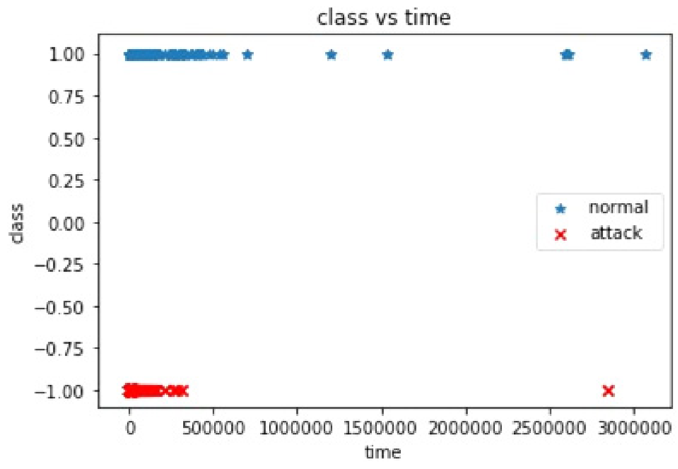

Fig. 2. Class versus time.

requests as possible with victim IoT devices to exhaust resources.[14] Figure 3 depicts this. If packet size becomes small then botnets can make lot of connections with victim IoT devices that are the objectives of a botnet. In botnet traffic, the average input packet size is 1.604116e+02 and most of the packet size is less than 2.024552e+07 (Table 4). As a feature, the input packet size can more readily help the ML calculation to classify the normal and botnet IoT traffic.

5.1.4. *Protocol (pr)*

Distributions of protocol highly vary for normal and botnet IoT traffic. In normal IoT traffic, protocol distributions are much more closely related to UDP and TCP.

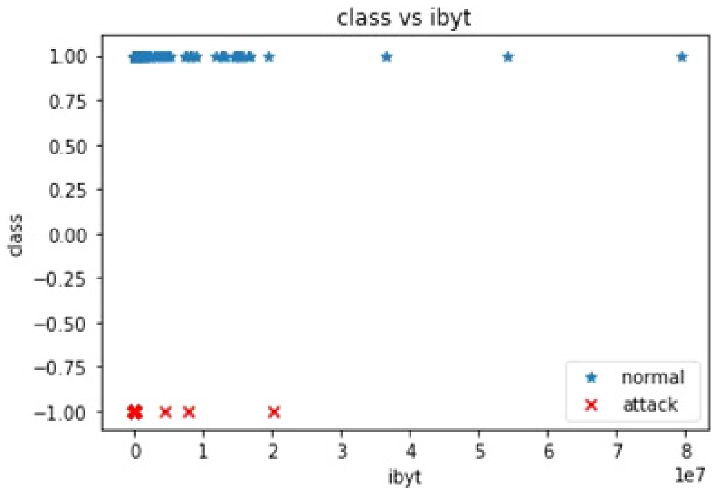

Fig. 3. Class versus i byte.

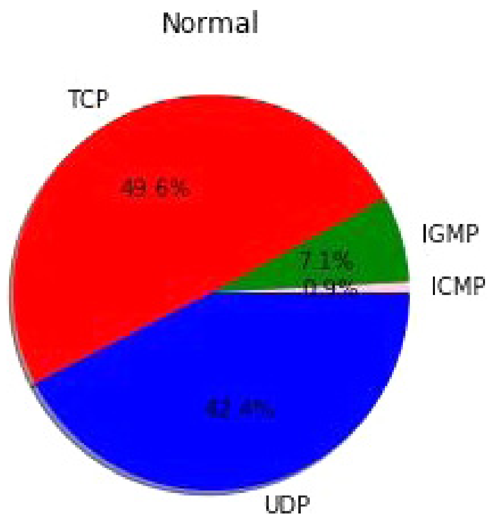

Fig. 4. Normal traffic protocol distributions.

For our dataset, UDP and TCP disseminations are somewhere in the range of 42.4% and 49.6% (Fig. 4). However, there is huge difference when IoT devices are victimized by the botnet. In botnet traffic, protocol distributions for UDP and TCP are between 1.7% and 96.3% (Fig. 5). TCP is out of the number in botnet IoT traffic compared with other protocols. It can be a pleasant feature to be learned by our model.

5.1.5. *Source port (sp)*

Source port number is a number for correspondence that is related to the beginning applications on the local host. IoT gadgets have some predetermined numbers of

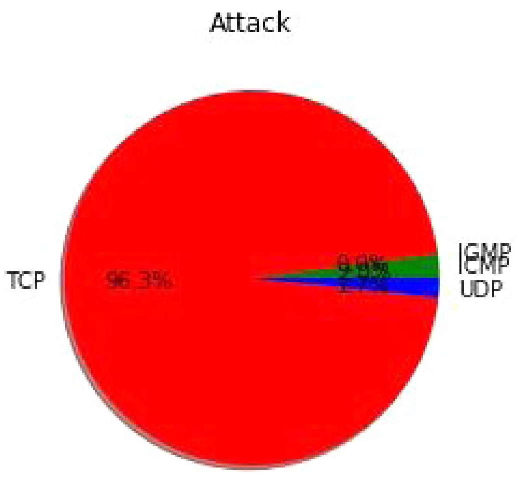

Fig. 5. Botnet protocol distributions.

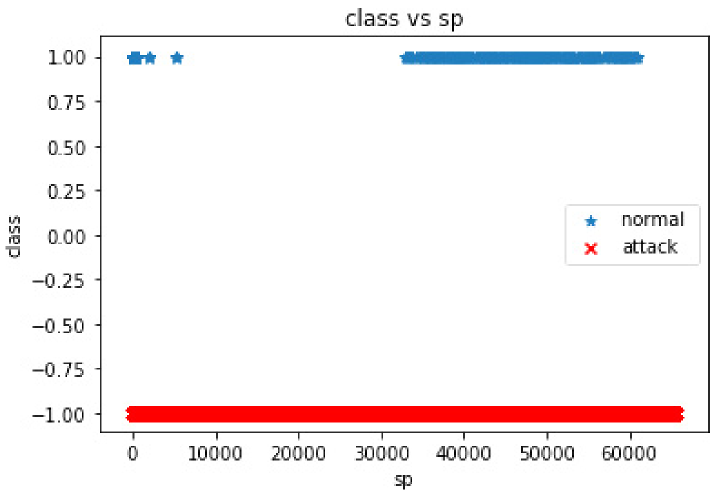

Fig. 6. Class versus sp.

communications. Generally, normal traffic changes less source ports; however, malicious or botnet traffic changes lots of source ports with the goal that it can send its activities from the different gadgets and can utilize IoT resources (Fig. 6). The more changes the number of source ports, the greater is the likelihood of the malicious attack on IoT devices.

5.1.6. *Destination port (dp)*

The destination port number is the number for this correspondence related to the destination application on the remote host. IoT devices have some short communication with other devices, that is why its destination ports are less in number and rarely change (Fig. 7). However, botnet regularly changes its destination to keep the IoT devices busy. It is a significant characteristic for the botnet related to IoT traffic.

5.1.7. *Categorical feature and normalization the data*

There are various methods to deal with categorical features. Our selected features have one categorical feature named protocol. Most of the ML algorithms do not work directly on categorical features. There are different techniques to deal with categorical features. Scikit-learn library and pandas library are mostly used for taking care of categorical features. We deal with this categorical feature by using a pandas library called get dummies(). For to learn our machine learning algorithm easily, we additionally normalize our data by the following equation:

$$X_{\text{newvalue}} = (X - X_{\min})/(X_{\max} - X_{\min}), \tag{1}$$

where X_{newvalue} is the Normalized value, X is a set of the observed values present in X, $X_{\min}$ is the minimum value in X, and $X_{\max}$ is the maximum value in X.

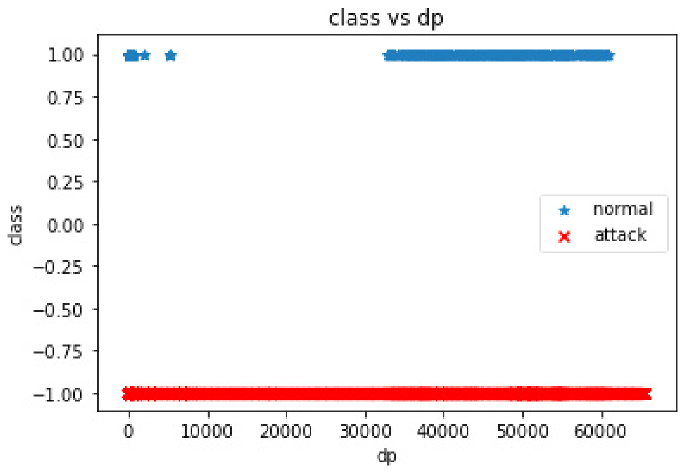

Fig. 7. Class versus dp.

5.2. *Training the data using various ML algorithm*

The dataset consists of an imbalance number of label data. It is about 95% for normal traffic and 5% for botnet (attack) traffic data. We definitely realize that IoT gadgets have some particular tasks to do and IoT gadgets carry out these responsibilities over and over until some malicious attack takes place on it. That is the reason the quantity of attack data is quite smaller in number. We label the data 1 for normal traffic and −1 for botnet (attack) traffic as we want to apply various ML algorithms on it. All in all, one-class classifier algorithm takes one-class data as train data and makes outliers based on data features. From that point onward, if test data satisfy normal traffic, it returns 1 otherwise −1. We used a method called train test split() from the Scikit-learn library to split our normal data into 80% and 20% for the training phase and test phase, respectively. After that, we apply various ML algorithms including OCSVM, LOF, and EE on it. Our normal traffic data is 1300, out of those data, we use 1040 data for training and the remaining data and malicious traffic data comprise testing data.

5.3. *Evaluation of the model by performance parameter*

How our model performs has to be evaluated by its testing stage. In the testing stage, we do concatenation of our test data, and we split from the train test split() method. We test our model by using both normal data and botnet traffic data. In the test phase, our model is good enough as it can classify normal traffic and botnet traffic quite successfully and that is our main goal. We likewise check our model's performance by different measurements:

- Accuracy $= (\mathrm{TP} + \mathrm{TN})/(\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN})$;
- Recall $= \mathrm{TP}/(\mathrm{TP} + \mathrm{FN})$;

- Precision = TP/(TP + FP);
- F2 score = (5 ∗ Precision ∗ Recall)/(4 ∗ Precision + Recall);

where

TP = true positive,
TN = true negative,
FP = false positive,
FN = false negative.

6. Results and Discussion

The execution flow of the methodology that we described earlier is depicted in Fig. 8. In this section, we discuss the experimental result of our work. We test three ML algorithms to differentiate normal traffic and botnet traffic:

- OCSVM,
- LOF,
- EE.

We implement these ML algorithms from the Scikit-learn library.[15] We also use the default hyperparameter in our experiment. We train our model and we check our model by various performance metrics parameters (Table 5). In Table 5, we represent the accuracy, recall, precision, and F2 score. In our training phase, we use MC I1, SC I3, and ST I3 data. Note that no botnet data are used during training the data. The accuracy level of our used ML algorithms varies between 0.772 and 0.99. In the MC I1 dataset, OCSVM and EEP perform better, achieving accuracy of approximately 0.999; however, LOF performs the worst, achieving an accuracy level of 0.772. In both SC I3 and ST I3 datasets, again OCSVM and EEP perform better and their accuracy level remains the same. Although LOF performs the worst for this dataset, its accuracy level increases to 0.848 and 0.878 (Table 5). If we consider three

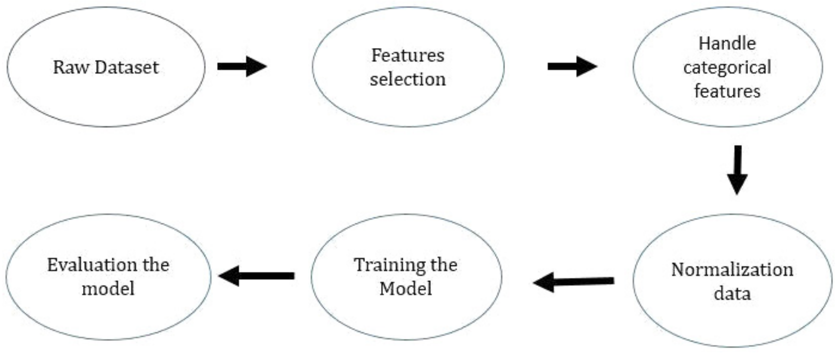

Fig. 8. Working diagram.

Table 5. Experimental result with various performance metrics.

Performance metrics	MC I1			SC I3			ST I3		
	OCSVM	LOF	EEP	OCSVM	LOF	EEP	OCSVM	LOF	EEP
Accuracy	0.999	0.772	0.911	0.999	0.848	0.999	0.999	0.878	0.998
Recall	0.999	0.772	0.911	0.998	0.847	0.998	0.999	0.878	0.998
Precision	0.999	0.999	0.999	0.997	0.999	0.999	0.998	0.998	0.999
F2 score	0.999	0.809	0.927	0.998	0.875	0.997	0.999	0.900	0.998

algorithms that we used, among them, the OCSVM performs the best. The second place will be for EE, and finally the third place will be for LOF.

The result varies among the three algorithms because of the dataset as well as for the algorithms. OSCVM deals with the outlier nicely. We could have the possibility to encounter some data that has never been observed before. However, it would be simple for OSCVM to identify this, but for a classification model, it may suffer to deal with.[17] As our goal is to detect botnet data, a new type of data, it performs better in this scenario. It also has a good ability to deal with the categorical features.[18] As our dataset has categorical features, OSCVM performs better in our dataset. Our dataset is an imbalance as well and OSCVM is also good for an imbalanced dataset. For that, OSCVM performs best in our dataset.

EE assumes that the data are from a known distribution to define the outlier. It tries to define a shape from the data. It has good quality to handle the data that has the mean value near to zero.[19] As we have noticed from the features table, our data has some of the features whose mean value is near to zero. This ability has made EE perform also better in our dataset.

LOF does not define the outlier and it is an algorithm that follows the nearest-neighbor method like the k-NN algorithm. LOF calculates the local deviation of the density of a given sample in comparison to its neighbors.[20] For this, it performs worse than other used algorithms.

7. Conclusion

The accelerated growth of the internet over the last decade has evidently encouraged a spike in cyber-attack incidents. Nowadays, the enormous number of IoT devices is usually more insecure than ordinary desktop computers. So, they pose security threats to information systems. The implementation of IDS to identify infected devices and the appropriate protection measures are a way of addressing this problem.

In this paper, we have established an approach for detecting botnets on IoT devices using OSVM, EE, and LOF which have been trained in network-based data, including time duration, input packet size, protocol distributions, source port, and destination port. Experimental results from the botnet dataset suggest that the ML models have achieved accuracy between 77% and 99%. We have found almost 99% accuracy from the OSVM algorithm. On the basis of this result, we proposed an

OSVM algorithm for the detection of botnets for IOT devices. In future, we have planned to use deep neural network to achieve more accurate results in botnet detection.

References

1. J. Liu, Y. Xiao, K. Ghaboosi, H. Deng and J. Zhang, Botnet: Classification, attacks, detection, tracing, and preventive measures, *EURASIP J. Wirel. Commun. Network.* **2009**(1) (2009) 692654.
2. C. Zhang and R. Green, Communication security in internet of thing: Preventive measure and avoid DDOS attack over IoT network, in *Proc. 18th Symp. Communications & Networking* (2015), pp. 8–15.
3. Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher and Y. Elovici, Network-based detection of IoT botnet attacks using deep autoencoders, *IEEE Pervas. Comput.* **17**(3) (2018) 12–22.
4. E. Alomari, S. Manickam, B. Gupta, S. Karuppayah and R. Alfaris, Botnet-based distributed denial of service (DDOS) attacks on web servers: Classification and art, arXiv preprint arXiv:1208.0403 (2012).
5. K. M. Prasad, A. R. M. Reddy and K. V. Rao, Dos and DDOS attacks: Defense, detection and traceback mechanisms-a survey, *Global J. Comp. Sci. Technol.* (2014).
6. X. D. Hoang and Q. C. Nguyen, Botnet detection based on machine learning techniques using DNS query data, *Fut. Internet* **10**(5) (2018) 43.
7. V. H. Bezerra, V. G. T. da Costa, S. Barbon Junior, R. S. Miani and B. B. Zarpelão, Iotds: A one-class classification approach to detect botnets in internet of things devices, *Sensors* **19** (2019) 3188.
8. H. Bezerra, V. G. T. da Costa, S. B. Junior, R. S. Miani and B. B. Zarpelao, One-class classification to detect botnets in iot devices, in *Anais do XVIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais* (2018), pp. 43–56, SBC.
9. K. Ashton *et al.*, That 'internet of things' thing, *RFID J.* **22**(7) (2009) 97–114.
10. J. Habibi, D. Midi, A. Mudgerikar and E. Bertino, Heimdall: Mitigating the internet of insecure things, *IEEE Inter. Things J.* **4**(4) (2017) 968–978.
11. Available at https://scikit-learn.org/stable/modules/generated/sklearn.covariance. Elliptic Envelope.html?fbclid=IwAR328WQljztPiuyxT-EqpbutlXGVfTFXgvd8BuLclroru 14oL8wsjdng7t4.
12. Available at https://scikit-learn.org/stable/modules/generated/sklearn.svm. One-ClassSVM. html?fbclid= IwAR2Vn41qz9zzxtGduzra_CHtPx9EmGX66QUqVuPM50lKunKR 0u5mDaZR7 _w#sklearn.svm.OneClassSVM.
13. V. H. Bezerra, V. G. T. da Costa, R. A. Martins, S. B. Junior, R. S. Miani and B. B. Zarpelao, Providing IoT host-based datasets for intrusion detection research, in *Anais do XVIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais* (2018), pp. 15–28, SBC.
14. R. Doshi, N. Apthorpe and N. Feamster, Machine learning DDOS detection for consumer internet of things devices, in *IEEE Security and Privacy Workshops (SPW)* (2018), pp. 29–35.
15. Available at https://scikit-learn.org/stable/.
16. M. M. Breunig, H. P. Kriegel, R. T. Ng and J. Sander, LOF: Identifying density-based local outliers, in *Proc. 2000 ACM SIGMOD International Conference on Management of Data (2000)*, Dallas, TX, USA, 16–18 May 2000, pp. 93–104.
17. V. Hodge and J. Austin, A survey of outlier detection methodologies, *Artif. Intell. Rev.* **22** (2004) 85–126. https://doi.org/10.1023/B:AIRE.0000045502.10941.a9.

18. B. Scholkopf, R. C. Williamson, A. J. Smola, J. S. Taylor and J. C. Platt, Support vector method for novelty detection, in *Advances in Neural Information Processing Systems 12 (NIPS 1999) Conf.*, The MIT Press, Vol. **11** (1999).
19. Available at https://scikit-learn.org/stable/modules/generated/sklearn.covariance. Elliptic Envelope.html.
20. Available at https://scikit-learn.org/stable/modules/generated/sklearn.neighbors. Local OutlierFactor.html?highlight =localoutlier#sklearn.neighbors.LocalOutlierFactor.
21. S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix and P. Hakimian, Detecting P2P botnets through network behavior analysis and machine learning, in *2011 Ninth Annual Int. Conf. Privacy, Security and Trust* (2011), pp. 174–180.
22. A. A. Galtsev and A. M. Sukhov, Network attack detection at flow level, in *Smart Spaces and Next Generation Wired/Wireless Networking*, eds. S. Balandin, Y. Koucheryavy and H. Hu, ruSMART 2011, NEW2AN 2011, pp. 326–334. *Lecture Notes in Computer Science* (2011), Vol. 6869. Springer. https://doi.org/10.1007/978-3-642-22875-9_30.