AN INTRUSION DETECTION AND CLASSIFICATION SYSTEM FOR IOT-MQTT NETWORKS

A PROJECT REPORT

Submitted By

ESHA A.	**1905012**
KOUSIKA M	**1905024**
SHUBASHINI J	**1905049**
ASIYA BEGAM M	**2005202**

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

COIMBATORE INSTITUTE OF TECHNOLOGY

(Government Aided Autonomous Institution Affiliated to Anna University)

COIMBATORE – 641 014

ANNA UNIVERSITY – CHENNAI 600 025

DECEMBER 2022

COIMBATORE INSTITUTE OF TECHNOLOGY
(Government Aided Autonomous Institution Affiliated to Anna University)
COIMBATORE – 641 014

ANNA UNIVERSITY – CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project **"AN INTRUSION DETECTION AND CLASSIFICATION SYSTEM FOR IOT-MQTT NETWORKS"** is the Bonafide work of **ESHA A (1905012), KOUSIKA M (1905024), SHUBASHINI J (1905049), ASIYA BEGAM M (2005202)** under my supervision during the academic year 2022-2023.

SIGNATURE	**SIGNATURE**
Dr.M. MOHANAPRIYA, M.E., Ph.D	**Dr. KOUSALYA G, M.E. Ph.D.,**
ASSOCIATE PROFESSOR	**HEAD OF THE DEPARTMENT**
Department of Computer Science and Engineering,	Department of Computer Science and Engineering,
Coimbatore Institute of Technology,	Coimbatore Institute of Technology,
Coimbatore – 641 014.	Coimbatore – 641 014.

Certified that the candidates were examined by us in the **19CS65 Main Project** viva voce examination held on ____________.

INTERNAL EXAMINER   EXTERNAL EXAMINER

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

We express our sincere thanks to our Secretary **Dr. R. Prabhakar,** Advisor **Dr. V. Selladurai** and our Principal **Dr. A. Rajeswari** for providing us a great opportunity to carry out our work. The following words are a small part to express our gratitude to them. This work is the outcome of their inspiration and product of plethora of their knowledge and rich experience.

We record the deep sense of gratefulness to **Dr. G. Kousalya**, Head of the Department of Computer Science and Engineering, for her encouragement during this tenure.

We equally tender our sincere thankfulness to our project guide **Dr.M.Mohanapriya,** Associate Professor, Department of Computer Science and Engineering, for his valuable suggestions and guidance during this course.

During the entire period of study, the entire staff members of the Department of Computer Science and Engineering have offered ungrudging help. It is also a great pleasure to acknowledge the unfailing help we have received from our friends.

It is matter of great pleasure to thank our parents and family members for their constant support and co-operation in the pursuit of this endeavour.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS & SYMBOLS

CNN – Convolutional Neural Network

1D-CNN - One-dimensional Convolutional Neural Network

ReLU - Rectified Linear Activation Function

IDS – Intrusion Detection System

MQTT – Message Queuing Telemetry Transport

DOS - Denial of Service

DT - Decision Tree

EML - Ensemble Machine Learning

IoT - Internet of Things

RMSP - Root Mean Square Propagation

ABSTRACT

ABSTRACT

The growing development of IoT devices creates a large attack surface for cybercriminals to conduct potentially more destructive cyber-attacks which led to the development of the novel intrusion detection system (IDS). The invention of IDSs affects deals with both the academic community and the industry globally since it affects economic costs, damages to reputation, and legal sequences, by every cyber-attack. It is therefore of considerable importance for networks to be secured from unauthorized access, for user interaction and user data to be protected and to disclose new security vulnerabilities. An efficient security strengthening tool for detecting and protecting cyber assaults on any network or host is the IDS. IDSs are accountable for the detection of suspicious behaviors and the adequate protection of the network against attacks and the reduction in losses such as financial and functional. Therefore an intrusion detection system is to be proposed to detect and classify the security attacks (Brute-force, Denial of Service(DOS), Flood, Legitimate, Malformed, SlowITe) using Machine Learning algorithms. The proposed work will be applied on the Message Queuing Telemetry Transport (MQTT) protocol running in the application layer .This work aims to increase the security of MQTT protocol at application level.

INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1. INTRODUCTION

The growing development of IoT (Internet of Things) devices creates a large attack surface for cybercriminals to conduct potentially more destructive cyberattacks; as a result, the security industry has seen an exponential increase in cyber-attacks. Many of these attacks have effectively accomplished their malicious goals because intruders conduct cyber-attacks using novel and innovative techniques.Detection systems assume a crucial role in the cyber-security field: based on innovative algorithms such as machine learning, they are able to identify or predict cyber-attacks, hence to protect the underlying system.

An intrusion detection system is to be proposed to detect and classify the security attacks (Brute-force, Denial of Service (DOS), Flood, Malformed, SlowITe) using Machine Learning algorithms. The proposed work will be applied on the Message Queuing Telemetry Transport (MQTT) protocol running in the application layer. This work aims to increase the security of MQTT protocol at application level.

1.2. DENIAL OF SERVICE

Denial of service attacks are executed to prevent the service to serve legitimate clients. In this case, the MQTT protocol is targeted with the aim to saturate the broker, by establishing several connections with the broker and sending, for each connection, the higher number of messages possible. In order to implement this attack, we adopted the MQTT-malaria tool, usually adopted to test scalability and load of MQTT services.

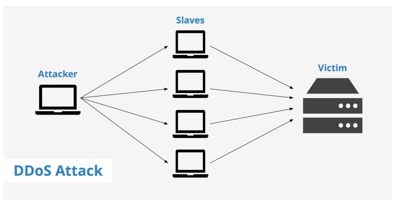


Figure 1.1 Denial of Service

1.3. FLOODING

In this case, restarting a system will usually fix an attack that crashes a server, but flooding attacks are more difficult to recover from. A malicious IoT device periodically sends a huge amount of malicious MQTT data, in order to seize all resources of the server, in terms of connection slots, networks or other resources that are allocated in limited amount. Differently on the previous attack, this attack tries to saturate the resources by using a single connection instead of instantiate multiple connections. This attack was generated in this case by using a module inside the IoT-Flock tool.

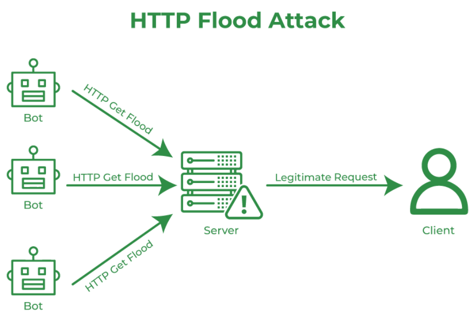


Figure 1.2 Flooding

1.4. SLOWITE

The Slow DoS against Internet of Things Environments (SlowITe) attack is a novel denial of service threat targeting the MQTT application protocol. Particularly, unlike previous threats, being a Slow DoS Attack, SlowITe requires minimum bandwidth and resources to attack an MQTT service. Particularly, SlowITe initiates a large number of connections with the MQTT broker, in order to seize all available connections simultaneously. Under these circumstances the denial-of-service status would be reached.

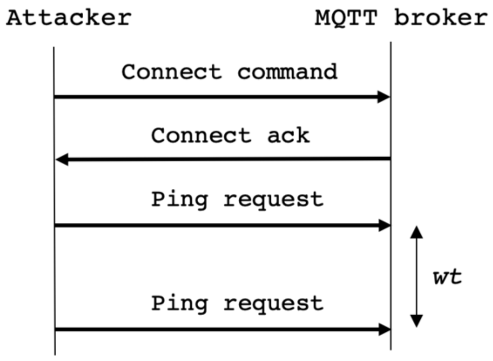


Figure 1.3 SlowITe

1.5. MALFORMED DATA

A malformed data attack aims to generate and send to the broker several malformed packets, trying to raise exceptions on the targeted service. Considering MQTTset, in order to perpetrate a malformed data attack, we adopted the MQTTSA tool, sending a sequence of malformed CONNECT or PUBLISH packets to the victim in order to raise exceptions on the MQTT broker. A malformed packet attack occurs when malformed IP packets are sent to a target system, causing the system to work abnormally or break down. With the capability of defending against such attacks, a device can detect and discard malformed packets in real time.

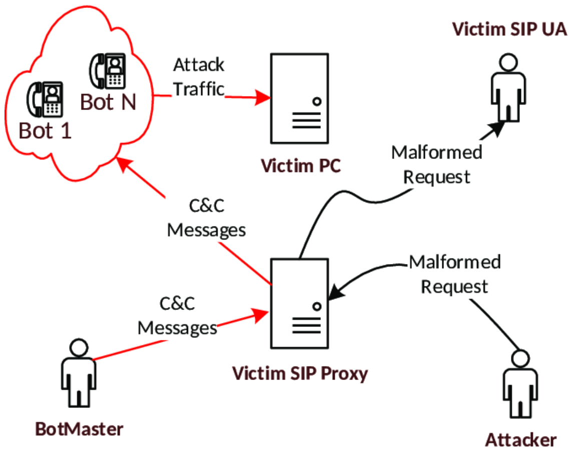


Figure 1.4 Malformed Data

1.6. BRUTE FORCE AUTHENTICATION

A brute force attack consists in running possible attempts to retrieve users credentials used by MQTT. Regarding MQTTset, the attacker's aim is to crack users' credentials (username and password) adopted during the authentication phase. Also in this case, we used the MQTTSA tool. Particularly, in order to recall to a real scenario, we adopted the rockyou.txt word list, that is considered a popular list, widely adopted for brute force and cracking attacks. For our tests, the credentials are stored on the word list used by the attacker

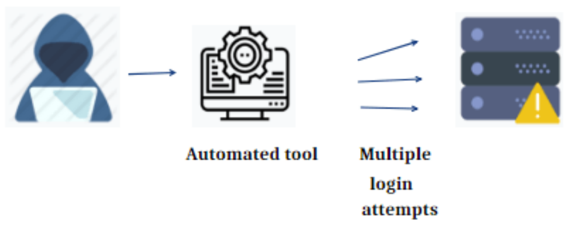


Figure 1.5 Brute Force Authentication

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

2.1. INTRODUCTION

The increased use of the Internet has prompted IoT protection firms to build more sophisticated technologies and standard security approaches to protect IoT devices from intruders. There are a wide variety of methods that are available for network anomaly detection. Machine learning has been both necessary and effective in the timely identification of cyber-attacks. While several anomaly-detection methods are used, fewer attempts were made to implement CNN (Convolutional Neural Network) for anomaly detection. Malicious actions in IoT networks must be detected and stopped immediately; therefore, the function of IDS has become critical in securing IoT networks by identifying anomalies.

2.2. RELATED WORK

In this paper [1] the author has presented an IDS for IoT, which processes data from IoT devices using machine learning to detect deviations from the normal behaviour of the system and generates alerts in case of anomalies.The system has a prototype of the system using IoT devices subscribed to topics at an MQTT broker and provide experimental evaluation of the system under MQTT-related attacks. It suggests that ML based intrusion detection in MQTT based IoT networks can achieve impressive results even when not trained with previously known attacks. It suggests that ML based intrusion detection in MQTT based IoT networks can achieve impressive results even when not trained with previously known attacks. It suggests that ML based intrusion detection in MQTT based IoT networks can achieve impressive results even when not trained with previously known attacks. It evaluated the performances of the various methods with ROC Scores, AUC Scores and Accuracy Scores. It doesn't provide classification model for classification of various attacks.

In this paper [2] the work aims to increase the security of this protocol at application level, particularly mitigating Denial of Service (DoS) attacks. The system is based on the use of a host Intrusion Detection System (IDS) which applies a threshold-based packet discarding policy to the different topics defined through MQTT. It is solution for different type of attacks. It only used to ensure the security rather than detection and classification of MQTT attacks.

In this paper [3] the proposed method is Framework for attack detection and proposed SEN-MQTTSET dataset generation, DoS attack model, proposed multi-context features generation and EML, packet discarding policy to the different topics defined through MQTT. An ensemble multi-view cascade feature generation algorithm was used to generate optimized SENMQTT-SET from the raw dataset.Later, those features are evaluated by using ML algorithms such as Linear Regression, K-Nearest Neighbor, Random Forest, Naive Bias, Support Vector Machine, Gradient Boosting, and Decision Tree (DT). The generated proposed features show the effective intrusion detection system in MQTT cyber-attacks and achieve an accuracy of more than 99%. As the features of the dataset are in large volume, the space required to store the dataset is greater and some of the ML algorithms may not work properly.

In this paper [4] they designed and developed an MQTT parsing engine that can be integrated with network-based IDS as an initial layer for extensive checking against IoT protocol vulnerabilities and improper usage through a rigorous validation of packet fields during the packet-parsing stage. The steps include protocol intuition, vulnerabilities assessment, protocol identification, protocol parsing, strict protocol validations, and rules definition. Efficient data transmission and quick to implement, due to its being a lightweight protocol. Low network usage, due to minimized data packets. Efficient distribution of data. Successful implementation of remote sensing and control. Classification of attacks was not implemented.

In this paper [5] the proposed system is classification of the various attacks in Iot devices using CNN. This proposed model is implemented using CNN in 1D,2D,3D with various kind of datasets.1D gives the best classification scores. It automatically detects the important feature without any human supervision. CNN are excellent alternative for anomaly detection and classification. CNN are excellent alternative for anomaly detection and classification faster computations. Analysis of various dimension CNN model. CNN is complex algorithm to implement. Takes more time to implement CNN in all dimension form.

In this paper [6] Nmap is a security scanner used to discover open ports and services running on that port in a computer network. In this article ,Nmap is used for performing port scan by using Nmap it use any platform like Windows, Linux, Mac OS etc. Nmap has come with GUI or command line interface. It performs port scan using Nmap it will give information about all port state and services running on that port by using various techniques. It will help us to narrow our choice to whether to attack on that host or not. It doesn't provide any implementation on real time network.

In this paper [7] Port scanning is a reconnaissance method, which incorporates scanning the host for open ports and services. It often involves sending a message to each of the open ports and detecting which can be open. Using the time options to control the scanning time, in this experiment we use default time and 3 other time option and compared the scanning time Nmap network sweeping scan using -sn parameter which use ICMP packet to prob host in the network.Scan one host against 1000 tcp popular ports show traffic sent by Nmap scanner and time taken to complete the scan process. Scan strategy that how the penetration testing deal with large volumes of host like class B and class A to balance time and any traffic restriction. Able to detect only active attacks.

2.3. INFERENCE FROM LITERATURE SURVEY

- Literature survey of different papers provides a great insight over different MQTT Attacks on Iot Devices and various approaches to detect them most preferably Neural Networks.

- The Methodologies used in the paper include ML Algorithms such as Linear Regression, K-Nearest Neighbour, Random Forest, Naive Bias, Support Vector Machine, Gradient Boosting, and Decision Tree (DT)and so on.
- The system defined in the paper is based on the use of a host Intrusion Detection System (IDS) which applies a threshold-based packet discarding policy to the different topics defined through MQTT.

2.4. OBJECTIVE OF THE PROJECT

- It is important to monitor network traffic for suspicious activity and alert when such activity is discovered. It is also important to develop a system where we can analyse the activity and classify them. The aim of the work is to develop an Intrusion Detection System (IDS) to detect and classify Message Queuing Telemetry Transport (MQTT) based attacks on Internet of Things(IoT) devices. The Five types of MQTT attacks classified are Brute-force, Denial of Service (DOS), Flood, Malformed, SlowITe.

SYSTEM SPECIFICATION

CHAPTER 3

SYSTEM SPECIFICATION

The hardware and the software requirement of the proposed model are stated below.

3.1. SOFTWARE SPECIFICATION

IDE	: Jupyter notebook, Anaconda Navigator
Coding Language	: Python (machine learning model, deployment)

3.2. HARDWARE SPECIFICATION

Operating system	: Windows 10
Processor	: Intel core i5
Hard disk	: 1TB
RAM	: 8 GB

SYSTEM ANALYSIS

CHAPTER 4

SYSTEM ANALYSIS

4.1. INTRODUCTION

The growing development of IoT (Internet of Things) devices creates a large attack surface for cybercriminals to conduct potentially more destructive cyberattacks; as a result, the security industry has seen an exponential increase in cyber-attacks. Many of these attacks have effectively accomplished their malicious goals because intruders conduct cyber-attacks using novel and innovative techniques. There are a variety of tools available to detect attacks and exploits and take steps to block or stop cyber-attacks. Things like firewalls to prevent unauthorized traffic from entering the network, spam filters to reject unwanted email messages, and antimalware tools to protect endpoints from malware are universal across just about every organization, regardless of size or industry. Another valuable security tool that is almost as ubiquitous is a network IDS—or intrusion detection system. The block diagram of the proposed model is illustrated in Figure 4.1.

BLOCK DIAGRAM:

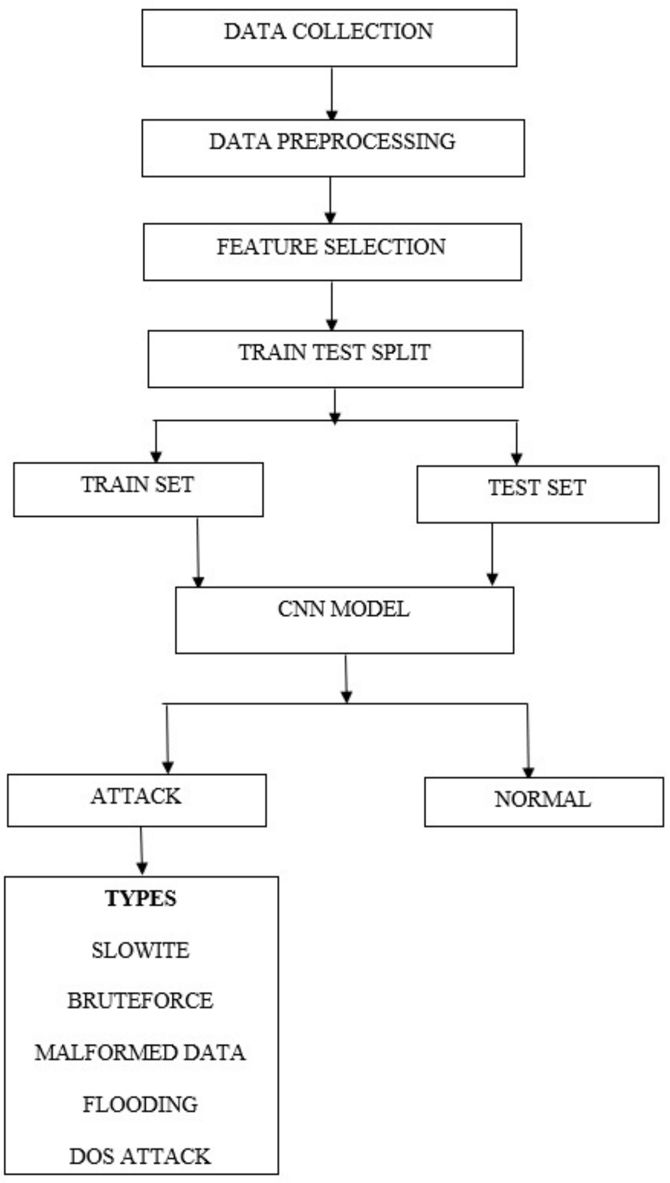


Figure 4.1 System Architecture diagram

4.2 METHODOLOGIES

DATASET

The dataset is composed by IoT sensors based on MQTT where each aspect of a real network is defined. In particular, the MQTT broker is instantiated by using Eclipse Mosquito and the network is composed by 8 sensors. The scenario is related to a smart home environment where sensors retrieve information about temperature, light, humidity, CO-Gas, motion, smoke, door and fan with differenttime interval since the behavior of each sensor is different with the others. According to Figure 1, sensors are located into two separate rooms.

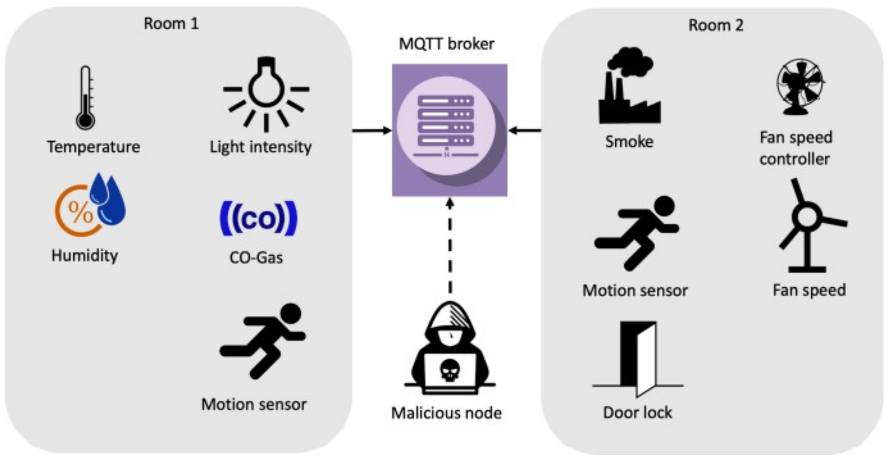


Figure 4.2 The scenario considered in MQTT set

As mentioned, the dataset isc composed by 8 MQTT sensors with different features. In table, the MQTT sensors are reported. Each sensor is associated with a data profile and a topic linked to the MQTT broker. The data profile consists of the type of data that the sensors communicate while the topic is defined by the sensor when sending the data to the broker. Finally, the sensors were conceptually divided into two rooms as if they were distributed in a smart house and the MQTT broker has 10.16.100.73 as IP address with 1883 as clear text communication port. In the table, the time could be periodic o random. This concept is important since a temperature sensor has a periodic behaviour over time, i.e., cyclically sending information retrieved from the environment periodically (defined as P). Instead, a motion sensor has a more random behavior since it sends information only when a user passes in front of the sensor (defined as R)). By analysing also this aspect, the dataset is even more valid as a real behavior of a home automation is simulated and implemented.

Attack	PACP Size(bytes)	Number of packets
Flooding denial of service	49,875,539	130,223
MQTT publish flood	8,212,656	613
SlowITe	972,272	9,202
Malformed data	1,038,590	10,924
Brute force authentication	1,397,132	14,501

Table 4.1 Types of attacks

Features of the Dataset

The dataset has 33 features and list of features is given in the Figure 4.4.

NO	NAME	DESCRIPTION	PROTOCOL LAYER
1	tcp.flags	TCP flags	TCP
2	tcp.time.data	Time TCP stream	TCP
3	tcp.len	TCP segment Len	TCP
4	mqtt.conack.flags	Acknowledge Flags	MQTT
5	mqtt.canack.flags.reserved	Reserved	MQTT
6	mqtt.conack.flags.sp	Session Present	MQTT
7	mqtt.conack.val	Return code	MQTT
8	mqtt.conflag.cleansess	Clean session flag	MQTT
9	mqtt.conflag.passwd	Password flag	MQTT
10	mqtt.conflag.qos	Qos level	MQTT
11	mqtt.conflag.reserved	(Reserved)	MQTT
12	mqtt.conflag.retain	Will retain	MQTT
13	mqtt.conflag.uname	User name flag	MQTT
14	mqtt.conflag.willflag	Will flag	MQTT
15	mqtt.conflag	Connect flags	MQTT
16	mqtt.dubflags	Dup flag	MQTT
17	mqtt.hdrflags	Header flags	MQTT
18	mqtt.kalive	Keep alive	MQTT
19	mqtt.len	Msg len	MQTT
20	mqtt.msg	Message	MQTT
21	mqtt.msgid	Message identifier	MQTT
22	mqtt.msgtype	Message type	MQTT
23	mqtt.proto_len	Protocol name length	MQTT
24	mqtt.protoname	Protocol name	MQTT
25	mqtt.qos	Qos level	MQTT
26	mqtt.retain	Retain	MQTT
27	mqtt.sub.qos	Requested qos	MQTT
28	mqtt.suback.qos	Granted qos	MQTT
29	mqtt.var	Version	MQTT
30	mqtt.willmsg	Will message	MQTT
31	mqtt.willmsg_len	Will message length	MQTT
32	mqtt.willtopic	Will topic	MQTT
33	mqtt.willtopic_len	Will topic length	MQTT

Table 4.2 The List of Features

4.3 ALGORITHM USED

CONVOLUTIONAL NEURAL NETWORK (CNN)

One-dimensional CNN (1D-CNN) is used for the detection of network intrusion. 1D-CNN is designed specifically to operate in 1-dimensional data. 1D-CNN consists of 1-dimensional convolution layers, pooling layers, dropout layers, and activation functions for handling the 1-dimensional data. 1D-CNN is configured using the following hyper-parameters: number of CNN layers, neurons in each layer, size of the filter, and subsampling factor of each layer. The convolution layer is the basic application of the filter to an input. Utilizing the filtering operation repeated times creates a feature map, which indicates the particular attributes related to the data points. Convolution is a linear operation that involved containing the multiplication with inputs with a set of weights. For this case, inputs are multiplied with the single-dimensional array weights, known as the kernel. This operation gives a unique value for each pass and executing this operation results in multiple values, known as a feature map.

ACTIVATION ReLU

Once the feature map is computed, each value is passed to the ReLU activation function. ReLU is a linear activation function that transforms the input as zero if it is negative, otherwise, it outputs the same input. The ReLU activation function allows the model to perform better, learn from the training data faster, and overcomes the vanishing gradient problem. It is demonstrated using Equation (4.1) as follows:

$$R(z) = \max(0, z) \qquad (4.1)$$

Here, z represents the input being provided to the activation function and R(z) represents a positive output of the activation function.

SOFTMAX LAYER

The Softmax regression is a form of logistic regression that normalizes an input value into a vector of values that follows a probability distribution whose total sum up to 1. Each value in the output of the softmax function is interpreted as the probability of membership for each class .

ADAM OPTIMIZER

Adaptive Moment Estimation is an algorithm for optimization technique for gradient descent. The method is really efficient when working with large problem involving a lot of data or parameters. It requires less memory and is efficient. Intuitively, it is a combination of the 'gradient descent with momentum' algorithm and the 'RMSP' algorithm.

Momentum:

This algorithm is used to accelerate the gradient descent algorithm by taking into consideration the 'exponentially weighted average' of the gradients. Using averages makes the algorithm converge towards the minima in a faster pace.

Root Mean Square Propagation (RMSP):

Root mean square prop or RMSprop is an adaptive learning algorithm that tries to improve AdaGrad. Instead of taking the cumulative sum of squared gradients like in AdaGrad, it takes the 'exponential moving average'.

Here, we control the rate of gradient descent in such a way that there is minimum oscillation when it reaches the global minimum while taking big enough steps (step-size) so as to pass the local minima hurdles along the way. Hence, combining the features of the above methods to reach the global minimum efficiently. Here, we control the rate of gradient descent in such a way that there is minimum oscillation when it reaches the global minimum while taking big enough steps (step-size) so as to pass the local minima hurdles along the way. Hence, combining the features of the above methods to reach the global minimum efficiently.

SYSTEM IMPLEMENTATION

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 MODULES

The modules used in the system are

- Import dataset and Data Pre-processing.
- Merging Attack and Normal data.
- Feature selection.
- Add Convolution Dense Layers with Activation Function.
- Add Output Layer with optimization Function.
- Train Test split.
- Training model for Detection of Attack and its classification.
- Testing the model.
- Obtaining Performance metrics.

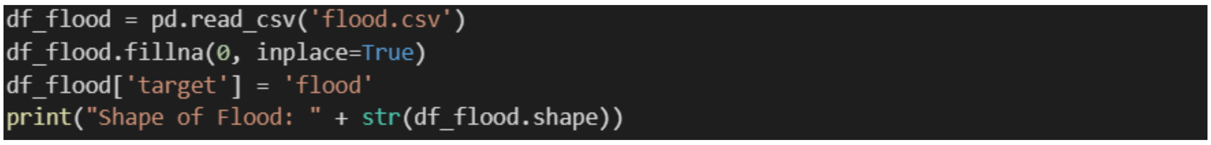
IMPORT DATASET AND DATA PRE-PROCESSING

The dataset holds both normal and abnormal network traffics where Legitimate is normal and different categories of attacks are abnormal network traffic which is attained by capturing the data of 8 consecutive days. The acquired data was divided into 6 different dataset.

There were a total of 6 different dataset in which 5 were different MQTT attack dataset and 1 was Legitimate users normal dataset. All the 6 dataset were imported and pre-processed.

Data Pre-processing is a technique used in data mining for the transformation of raw data into a format that can be utilized efficiently. It encompasses different techniques for data cleaning, data transformation, and data reduction. Data Cleaning is an essential step in pre-processing that removes the irrelevant and missing information from the dataset that could lead to the incorrectestimation of the target class.

The fillna() method was used to replaces the NULL values with a specified value 0. The different types of attacks and number of records were displayed in Table 4.1.

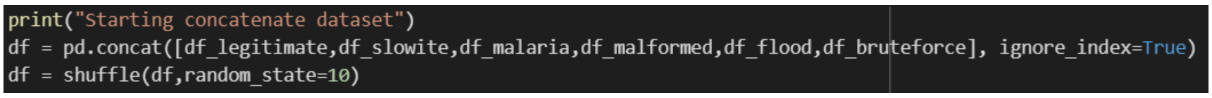


```
df_flood = pd.read_csv('flood.csv')
df_flood.fillna(0, inplace=True)
df_flood['target'] = 'flood'
print("Shape of Flood: " + str(df_flood.shape))
```

Figure 5.1 Import Dataset

MERGING ATTACK AND NORMAL DATA

All the 6 preprocessed dataset were merged into one single dataset. Data are combined or mixed to generate a single dataset including bothlegitimate and malicious traffic data. The entire work proposed was relying on this single MQTT dataset that was generated. The resulting MQTT dataset contains 2,830,743 records and 61 records in a dataset.



```
print("Starting concatenate dataset")
df = pd.concat([df_legitimate,df_slowite,df_malaria,df_malformed,df_flood,df_bruteforce], ignore_index=True)
df = shuffle(df,random_state=10)
```

Figure 5.2 Merge Attack and Normal data

FEATURE SELECTION

All available 61 features that were able to describe a connection were then recovered directly from the raw network data. Features are selected and filtered in order to focus on the most relevant ones able to characterize potential attacks and legitimate traffic. In particular, the features removed were:

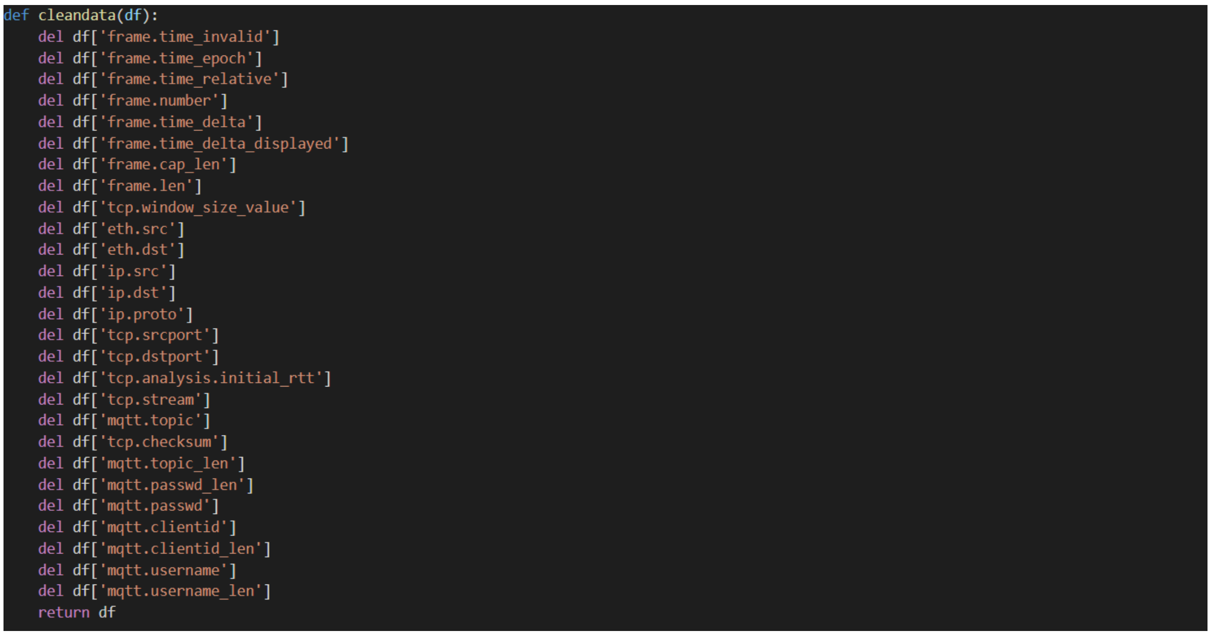
- Source/destination addresses and ports: Removed in order toallow detection to be more independent on networking configuration details (useful for DoS/DDoS attacks).

- Communication times: Removed since the identification ofattacks must not be dependent on times or schedules.
- tcp.stream: These parameter are related to a single execution and not useful for detection
- tcp.checksum: Removed as it represents the unique value for each packet of the communication
- MQTT ClientID, password, username and related lengths: Removed as they were related to a single execution and configuration and can be easily altered by an attacker.
- MQTT topic: Removed as this parameter easy to tune by the attacker and could be adoptedto discriminate legitimate and malicious behavior.
- iRTT: Removes as this parameter is used from Wireshark to define time between packets and not related to a connection.
- tcp.window_size_value: Removed as it was related to a single packet.

We removed values that are either missing or contain infinite floating point valuesthat can't be processed.

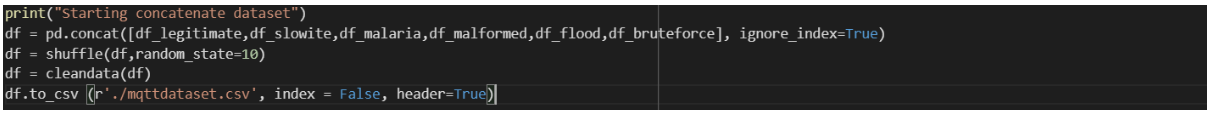
After removing unwanted features nearly 34 features were left in which the last feature, i.e., the 34th feature represents the category of attack traffic and denotedas the target.

The selected list of features after dropping all these unwanted features was displayed in Table 4.2.



```
def cleandata(df):
    del df['frame.time_invalid']
    del df['frame.time_epoch']
    del df['frame.time_relative']
    del df['frame.number']
    del df['frame.time_delta']
    del df['frame.time_delta_displayed']
    del df['frame.cap_len']
    del df['frame.len']
    del df['tcp.window_size_value']
    del df['eth.src']
    del df['eth.dst']
    del df['ip.src']
    del df['ip.dst']
    del df['ip.proto']
    del df['tcp.srcport']
    del df['tcp.dstport']
    del df['tcp.analysis.initial_rtt']
    del df['tcp.stream']
    del df['mqtt.topic']
    del df['tcp.checksum']
    del df['mqtt.topic_len']
    del df['mqtt.passwd_len']
    del df['mqtt.passwd']
    del df['mqtt.clientid']
    del df['mqtt.clientid_len']
    del df['mqtt.username']
    del df['mqtt.username_len']
    return df
```

Figure 5.3 Feature Selection



```
print("Starting concatenate dataset")
df = pd.concat([df_legitimate,df_slowite,df_malaria,df_malformed,df_flood,df_bruteforce], ignore_index=True)
df = shuffle(df,random_state=10)
df = cleandata(df)
df.to_csv (r'./mqttdataset.csv', index = False, header=True)
```

Figure 5.4 Concatenate Dataset

ADD CONVOLUTION DENSE LAYERS WITH ACTIVATION FUNCTION

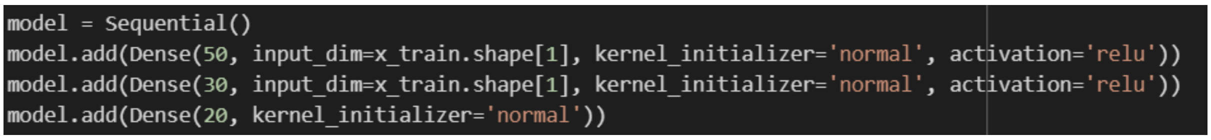
The neural network was implemented by using the sequential algorithm with Keras(Sequential) with the first hidden layer consisting of 50 nodes, the second of 30, the third of 20 and finally the last with 6 nodes relating to the 6 classes.
The hidden layer are characterized by a ReLU activation function and a normal kernel initializer. The last output layer was set with a softmax activation function

Hidden Layer

The input of size (33,None) was given to the dense layer and asking the dense layer to provide the output array of shape (None,50) by using the units parameter as 1700 with the ReLU activation function.

The input of size (33,None) was given to the dense layer and asking the dense layer to provide the output array of shape (None,30) by using the units parameter as 1530 with the ReLU activation function.

The input of size (33,None) to the dense layer and asking the dense layer to provide the output array of shape (None,20) by using the units parameter as 620.



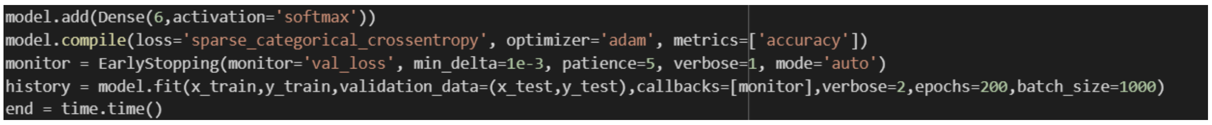
```
model = Sequential()
model.add(Dense(50, input_dim=x_train.shape[1], kernel_initializer='normal', activation='relu'))
model.add(Dense(30, input_dim=x_train.shape[1], kernel_initializer='normal', activation='relu'))
model.add(Dense(20, kernel_initializer='normal'))
```

Figure 5.5 Hidden layer

ADD OUTPUT LAYER WITH OPTIMIZATION FUNCTION

The final layer will reduce the vector of height 20 to a vector of six since there were six classes to predict. This reduction is done by another matrix multiplication. Softmax is used as the activation function. It forces all six outputs of the neural network to sum up to one. The output value will therefore represent the probability for each of the six classes. Adam optimizer and early stopping mechanism were also used.

Total Trainable param were 3976.

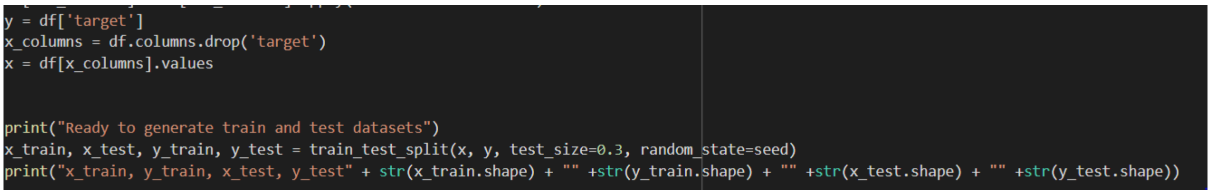


```
model.add(Dense(6,activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3, patience=5, verbose=1, mode='auto')
history = model.fit(x_train,y_train,validation_data=(x_test,y_test),callbacks=[monitor],verbose=2,epochs=200,batch_size=1000)
end = time.time()
```

Figure 5.6 Adding Output Layer with Optimization Function

TRAIN TEST SPLIT

Data were divided into features (X) and labels (y). The data frame gets divided into X_train, X_test, y_train and y_test. X_train and y_train sets were used for training and fitting the model. The train_test_split () method was used to split data into train and test sets. The train and test set were splitted in 70:30 ratio.

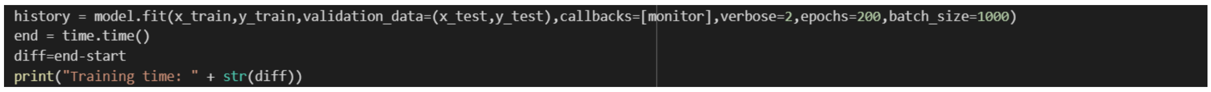


```
y = df['target']
x_columns = df.columns.drop('target')
x = df[x_columns].values


print("Ready to generate train and test datasets")
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=seed)
print("x_train, y_train, x_test, y_test" + str(x_train.shape) + "" +str(y_train.shape) + "" +str(x_test.shape) + "" +str(y_test.shape))
```

Figure 5.7 Train Test Split

TRAINING MODEL FOR DETECTION OF ATTACK AND ITS CLASSIFICATION

The built CNN model was trained using train set detect normal legitimate user data or classify attack data.

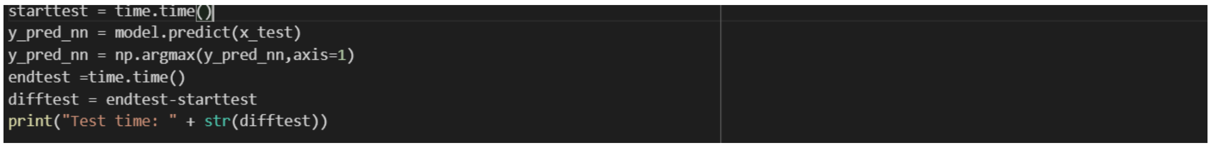


```
history = model.fit(x_train,y_train,validation_data=(x_test,y_test),callbacks=[monitor],verbose=2,epochs=200,batch_size=1000)
end = time.time()
diff=end-start
print("Training time: " + str(diff))
```

Figure 5.8 Training Model

TESTING THE MODEL

The trained model was tested using test dataset.



```
starttest = time.time()
y_pred_nn = model.predict(x_test)
y_pred_nn = np.argmax(y_pred_nn,axis=1)
endtest =time.time()
difftest = endtest-starttest
print("Test time: " + str(difftest))
```

Figure 5.9 Testing Model

OBTAINING PERFORMANCE METRICS

Evaluated the performance of the 1D-CNN classifier using different performance metrics such as confusion matrix, accuracy and F1 score. Accuracy and F1 Score can be calculated using the formula 5.1,

$$\text{Accuracy} = (TP + TN) \div \text{Total samples} \qquad (5.1)$$

$$\text{F1 Score} = (T2 \times \text{Precision} \times \text{Recall}) \div (\text{Precision} + \text{Recall}).$$

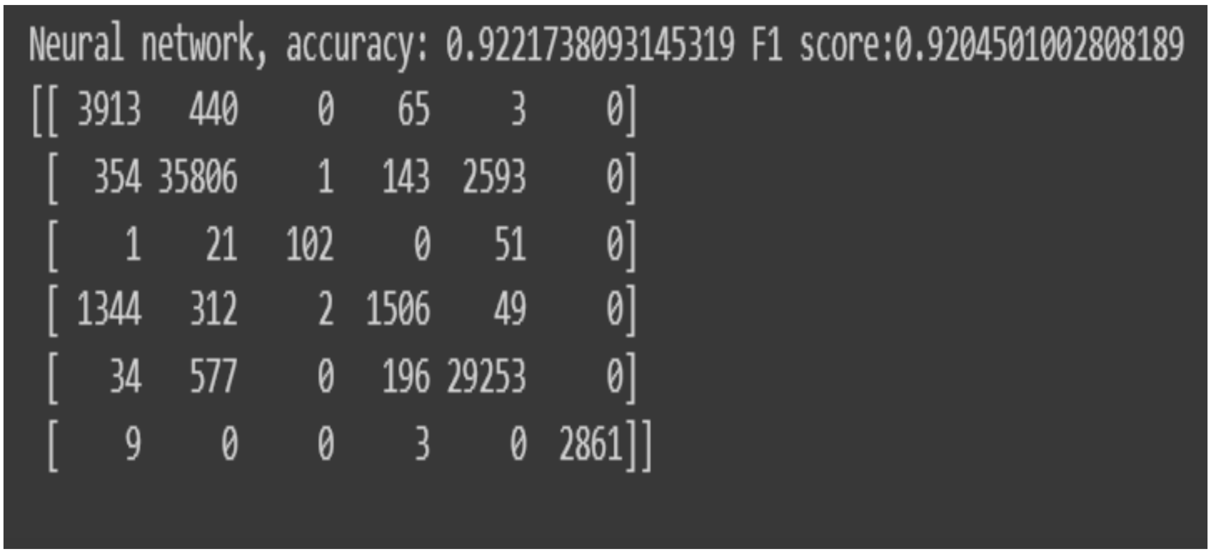
where,

• TP represents the number of attacks that are correctly classified as attacks

• FP represents the normal data classified as an attack

• TN represents the normal data classified as normal

• FN represents the attack as normal.



```
Neural network, accuracy: 0.9221738093145319 F1 score:0.9204501002808189
[[ 3913   440     0    65     3     0]
 [  354 35806     1   143  2593     0]
 [    1    21   102     0    51     0]
 [ 1344   312     2  1506    49     0]
 [   34   577     0   196 29253     0]
 [    9     0     0     3     0  2861]]
```

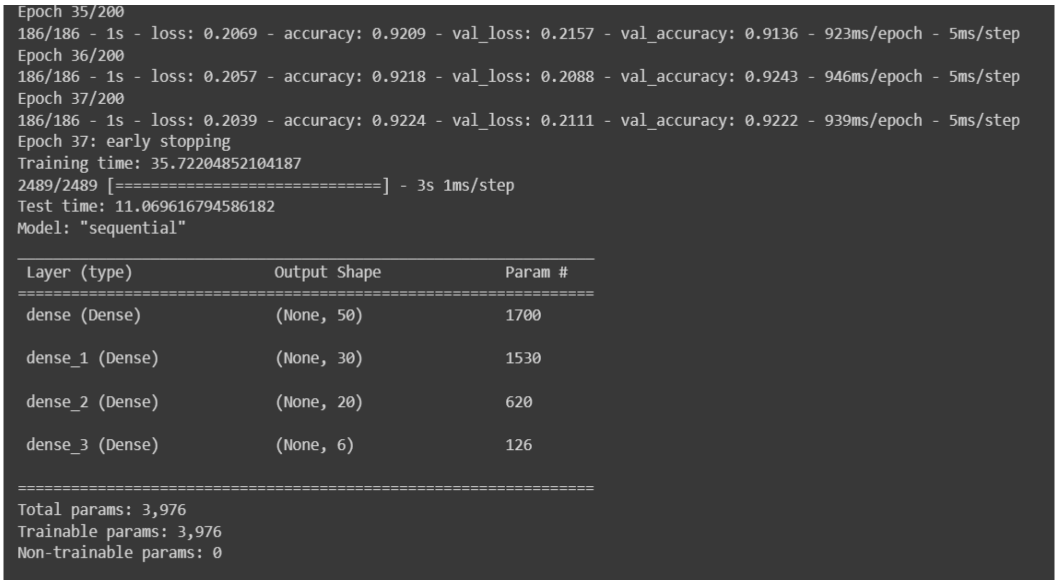
Figure 5.10 Obtaining Performance Metrics

RESULTS AND DISCUSSIONS

CHAPTER 6

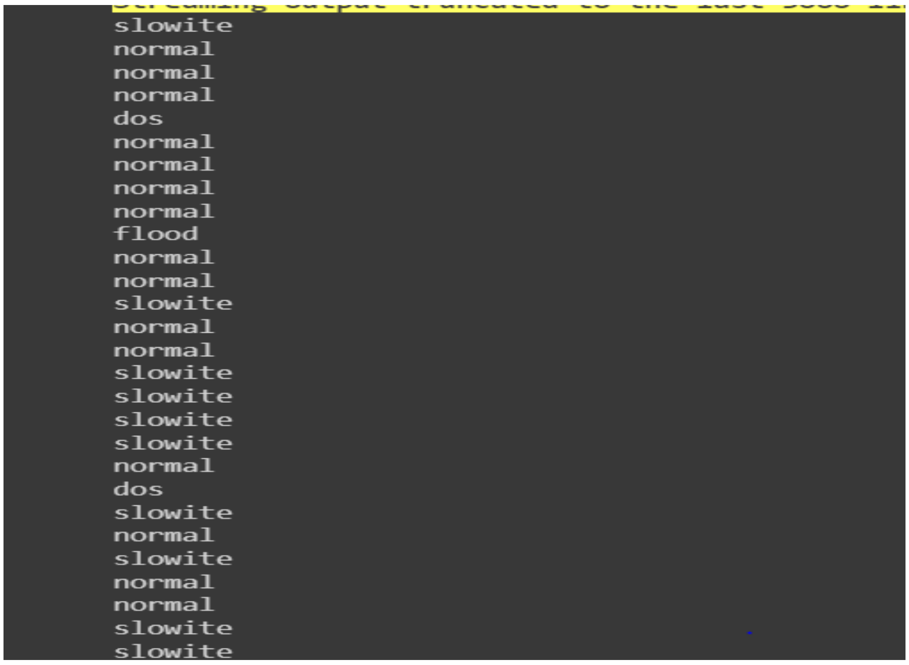
RESULTS AND DISCUSSIONS

1D-CNN is used for the detection of network intrusion. 1D-CNN is designed specifically to operate in 1-dimensional data. We divided the dataset into two sets, i.e., 70 and 30% for training and testing purposes, respectively.



```
Epoch 35/200
186/186 - 1s - loss: 0.2069 - accuracy: 0.9209 - val_loss: 0.2157 - val_accuracy: 0.9136 - 923ms/epoch - 5ms/step
Epoch 36/200
186/186 - 1s - loss: 0.2057 - accuracy: 0.9218 - val_loss: 0.2088 - val_accuracy: 0.9243 - 946ms/epoch - 5ms/step
Epoch 37/200
186/186 - 1s - loss: 0.2039 - accuracy: 0.9224 - val_loss: 0.2111 - val_accuracy: 0.9222 - 939ms/epoch - 5ms/step
Epoch 37: early stopping
Training time: 35.72204852104187
2489/2489 [==============================] - 3s 1ms/step
Test time: 11.069616794586182
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 50)                1700

 dense_1 (Dense)             (None, 30)                1530

 dense_2 (Dense)             (None, 20)                620

 dense_3 (Dense)             (None, 6)                 126

=================================================================
Total params: 3,976
Trainable params: 3,976
Non-trainable params: 0
```

Figure 6.1 CNN Model Summary



```
slowite
normal
normal
normal
dos
normal
normal
normal
normal
flood
normal
normal
slowite
normal
normal
slowite
slowite
slowite
slowite
normal
dos
slowite
normal
slowite
normal
normal
slowite
slowite
```

Figure 6.2 Classification Result

We evaluated the performance of the 1D-CNN classifier using different performance metrics such as accuracy andF1 score.

Accuracy = (TP + TN) ÷ Total samples

F1 Score = (T2 × Precision × Recall) ÷ (Precision + Recall).

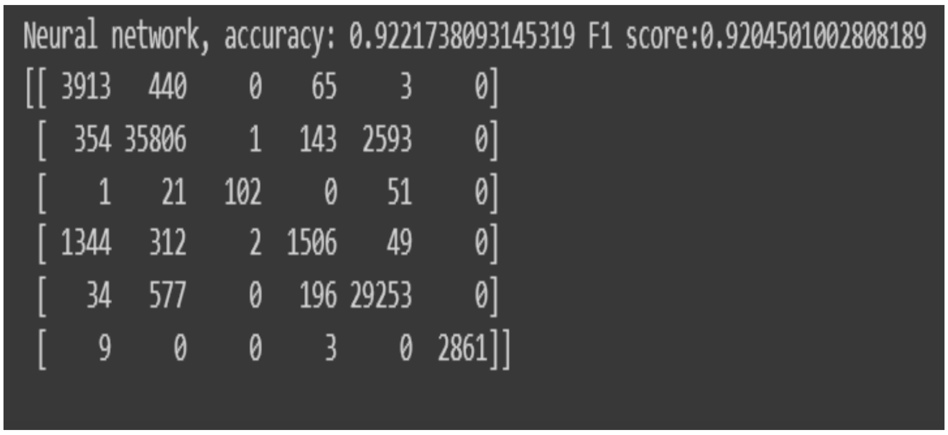
where,

• TP represents the number of attacks that are correctly classified as attacks

• FP represents the normal data classified as an attack

• TN represents the normal data classified as normal

• FN represents the attack as normal.



```
Neural network, accuracy: 0.9221738093145319 F1 score:0.9204501002808189
[[ 3913   440     0    65     3     0]
 [  354 35806     1   143  2593     0]
 [    1    21   102     0    51     0]
 [ 1344   312     2  1506    49     0]
 [   34   577     0   196 29253     0]
 [    9     0     0     3     0  2861]]
```

Figure 6.3 Performance metrics

CONCLUSION

CHAPTER 7

CONCLUSION

The recently reported vulnerabilities of MQTT were categorized into five types. The proposed Intrusion detection model for IoT networks used a convolutional neural network to detect and classify multiclass attacks. Model was built based on One-Dimensional Convolution Neural Network (1D-CNN) methodology for the detection of normal and five different types of MQTT based malicious network traffic. The model make use of the machine learning methodology by employing a 1D-CNN based architecture and achieved very promising results while performing multiclass classification. The proposed model architecture is simple and less computationally expensive. We achieved an overall accuracy of 92.25% with this approach. The proposed multiclass classification models showed high accuracy, precision, recall, and F1 score compared to existing classification strategies and recent machine learning implementations. This study findings indicate that the suggested model will aid in the development of an efficient Intrusion Detection System for IoT networks that has both a high detection rate and a low false alarm rate.

FUTURE WORK

CHAPTER 8

FUTURE WORK

The module II of future model includes building application model with different IoT sensors. Intruding the MQTT broker network traffic in an application using network traffic analysis tools like Nmap, IoT Flock. Collecting the network traffic using Wireshark as pcap files and converting it into csv file that the format suitable for detecting Intrusion in a built model from Module I. The collected data should be pre-processed and required features should be selected by removing unwanted features. Now the dataset obtained will be tested for Intrusion. If there is no attack the packets will be forwarded else discarded.

MODULE II

- Building application using different IoT sensors
- Intruding the MQTT broker
- Data collection using network analysis tools
- Collected network packets are converted into csv file
- Data preprocessing
- Feature selection
- Classify the data using the built model(Module I)
- Forward only the legitimate packet

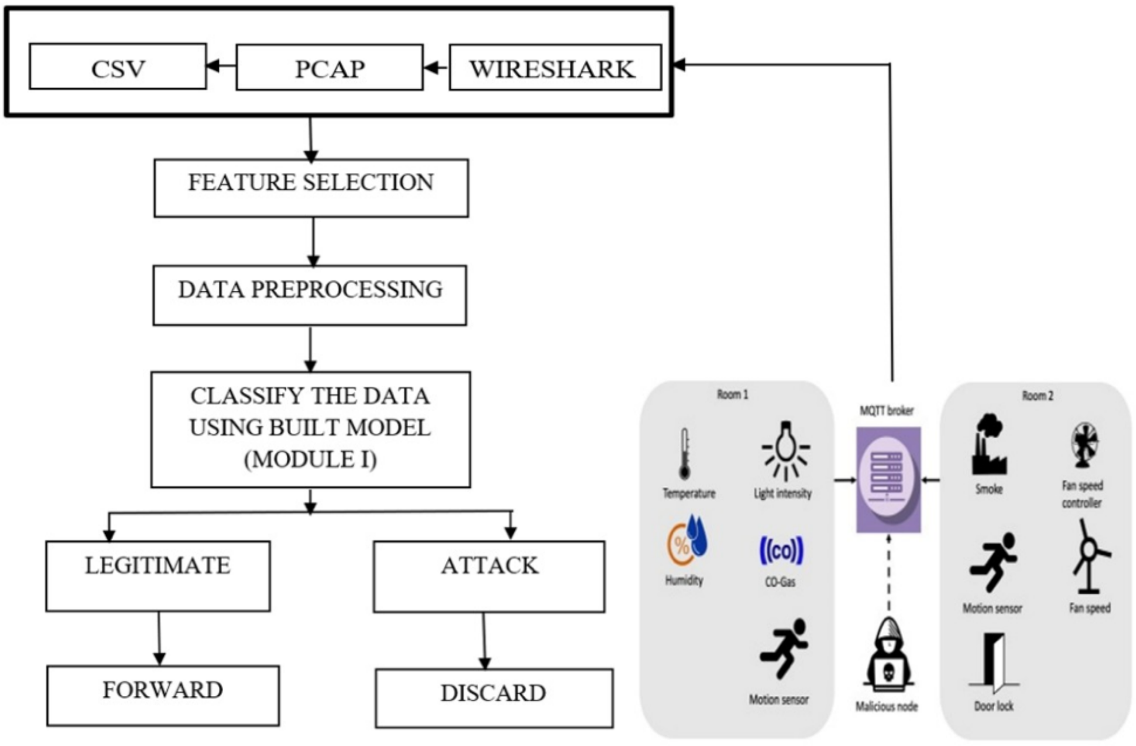


Figure 8.1 Block diagram for Module II

Based on the obtained results, a future work will be focused on the tuning of the hyper-parameters of the CNN algorithm in order to ensure the accuracy and F1 score about the balanced dataset and to define the best characteristics of the algorithm. The model can be extended and can be used to detect attacks against MQTT in an industrial scenario, such as a smart building or an industrial IoT network. The existing model is limited to the MQTT protocol so in future a similar analysis could be done for other IoT application layer protocols to analyse the major causes of the vulnerabilities reported in those protocols. The model can also be extended to detect even more types of network attacks on industrial medical application of IoT devices with less features.

REFERENCES

CHAPTER 9

REFERENCES

[1] Deepa Bura, Meeta Singh and Poonam Nandal "Predicting Secure and Safe Route for Women using Google Maps," In 2019 International Conference on Machine Learning,India, 14th -16th Feb 2019

[2] Amar Shukla, Avita Katal and Saurav Raghuvanshi "Criminal Combat: Crime Analysis and Prediction Using Machine Learning," In 2021 International Conference on Intelligent Technologies, Karnataka, India. June 25-27, 2021

[3] Edge, D., Walker, T., Meacock, R., Wilson, H., McNair, L., Shaw, J., ... and Sutton, M. "Secure pathways for women in the UK: lessons from the women's enhanced medium secure services (WEMSS) pilots," The Journal of Forensic Psychiatry & Psychology, 28(2), 206-225,2017

[4] Yu-Yueh Huang, Cheng-Te Li and Shyh-Kang Jeng, "Mining Location-based Social Networks for Criminal Activity Prediction," Proceedings of 24th IEEE International Conference on Wireless and Optical Communication, pp. 185-190, 2015

[5] Colleen McCue, "Data Mining and Predictive Analysis: Intelligence Gathering and Crime Analysis," ButterworthHeinemann, 2014.

[6] Arunima S. Kumar and Raju K. Gopal, "Data Mining based Crime Investigation Systems: Taxonomy and Relevance," Proceedings of Global Conference on IEEE Communication Technologies, pp. 850-853, 2015.

[7] Syed Naeem Firdous, Zubair Baig, Craig Valli and Ahmed Ibrahim "Modeling and Evaluation of Malicious Attacks against the IoT MQTT Protocol", 2017 IEEE International Conference on Internet of Things, 2017.

[8] Ahmad W. Atamli and Andrew Martin,“Threat-based Security

Analysis for the Internet of Things”, International Workshop on Secure Internet of Things, 2014.

[9] Syaiful Andy, Budi Rahardjo and Bagus Hanindhito, “Attack Scenarios and Security Analysis of MQTT Communication Protocol in IoT System”, Proc. EECSI 2017, Yogyakarta, Indonesia, 19-21 September 2017.

[10] Robert Moskowitz, Network Intrusion: Methods of Attack, [Online], Available: http://www.rsaconference.com/industry- topics/blog/network-intrusion-methods-of-attack

[11] Haripriya A. P and Kulothungan K, “Secure-MQTT: an efficient fuzzy logic-based approach to detect DoS attack in MQTT protocol for internet of things”, A. P. and K. EURASIP Journal on Wireless Communications and Networking, 2019.

[12] https://www.kaggle.com/datasets/cnrieiit/mqttset