MINOR PROJECT 1

ODD SEM 2021

IOT DETECTOR

Smarter approach to Detect IoT

GROUP – 15

GROUP- 15

Group Members

Ansh Goyal (9919103056)
Rohan Singh (9919103038)
Raghav Singh (9919103091)

Project Mentor

Dr. Charu Gandhi

OBJECTIVES

01

Collection of pcap captures of an IoT Environment.

02

Filtering of Packets on the basis of IP address.

03

Extraction of features from the dataset

04

Selection of Features by analysis done using weka

05

Classification and build a sniffer for monitoring

06

Link our trained model and Sniffer with Flask

INTRODUCTION

As the number of low-cost Internet-of-Things (IoT) devices increases dramatically in recent years, they have become ideal targets for flooding the network with junk traffic. Mostly every IoT Device is shipped to the market with default IP passwords. Operating systems of many IoT devices are often outdated or not well-configured. These device have very low vendor support so they rarely get software update or patches and they become vulnerable with the time because of outdated software. As a result these devices can be used by the advers adversary to flood the network with runt packets and this will lead to degradation of the network which affect all the other devices that are connect to that network. Now the main issue for administrator is that how to identify that this traffic is generated by which device, whether it is a IoT Device or a normal device. To solve this issue we built a tool using machine learning, called IoT Detector (IoTD) to detect and classify the IoT Device on the network by analysing packet capture (pcap), this tool will also provide you a real time graphical representation of mainly three type of traffic named TCP (Transmission Control Protocol), UDP (User Datagram Protocol) & ICMP (Internet Control Message Protocol). Experimental results shows that IoTD can accurately detect and identify IoT Devices. The accuracy evaluation among 12 devices of this study shows that 8 IoT Devices are true positive and the network sniffer is perfectly working on capturing network traffic.

STEPS

- Import required libraries
- Filtering out Data
- Feature Extraction
- Building a Sniffer
- Model Training
- Building Model and Classification
- Flask Integration
- Result

STEP 1 : IMPORTING LIBRARIES

```
from warnings import simplefilter

import numpy as np
import pandas as pd
import sklearn
from numpy import genfromtxt
from sklearn import datasets, svm
from sklearn.datasets import fetch_mldata
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (accuracy_score, confusion_matrix, f1_score,
                             precision_score, recall_score)
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler

simplefilter(action='ignore', category=FutureWarning)
```

STEP 2 : FILTERING OUT DATA

```
ip_filter = {}
ip_filter['TCP_Mobile'] = "'tcp && (ip.src==192.168.1.45)'"
ip_filter['TCP_Outlet'] = "'tcp && (ip.src==192.168.1.222) || (ip.src==192.168.1.67)'"
ip_filter['TCP_Assistant'] = "'tcp && (ip.src==192.168.1.111) || (ip.src==192.168.1.30)\
        || (ip.src==192.168.1.42) || (ip.src==192.168.1.59) || (ip.src==192.168.1.70)'"
ip_filter['TCP_Camera'] = "'tcp && (ip.src==192.168.1.128) || (ip.src==192.168.1.145) \
       || (ip.src==192.168.1.78)'"
ip_filter['TCP_Miscellaneous'] = "'tcp && (ip.src==192.168.1.216) || (ip.src==192.168.1.46)\
        || (ip.src==192.168.1.84) || (ip.src==192.168.1.91)'"
```

Packets are distributed on the basis of the ip address

All the files with their corresponding packets are saved in "filtered_pcap" directory

```
for oriPcapFile in glob.glob('./original_pcap/*.pcap'):
    for k in ip_filter.keys():
        os.system("tshark -r" + oriPcapFile + " -w- -Y " + \
        ip_filter[k] + ">> ./filtered_pcap/" + k+".pcap")
```

STEP 3 : FEATURE EXTRACTION

```
for ftdPcapFile in glob.glob('./filtered_pcap/*.pcap'):
    filename = ftdPcapFile.split('/')[-1]
    label = filename.replace('.pcap','')
    tsharkCommand = "tshark -r " + ftdPcapFile + " -T fields -e ip.len -e ip.hdr_len \
            -e ip.flags -e ip.ttl -e ip.proto -e ip.id -e ip.checksum -e tcp.srcport \
            -e tcp.dstport -e tcp.seq -e tcp.ack -e tcp.window_size_value -e tcp.hdr_len\
            -e tcp.flags -e tcp.len -e tcp.checksum -e tcp.stream -e tcp.urgent_pointer"
    allFeatures = str(os.popen(tsharkCommand).read())
    allFeatures = allFeatures.replace('\t',',')
    allFeatures = allFeatures.splitlines()
    for features in allFeatures:
        labelFeature.writelines(label+","+features+"\n")
```

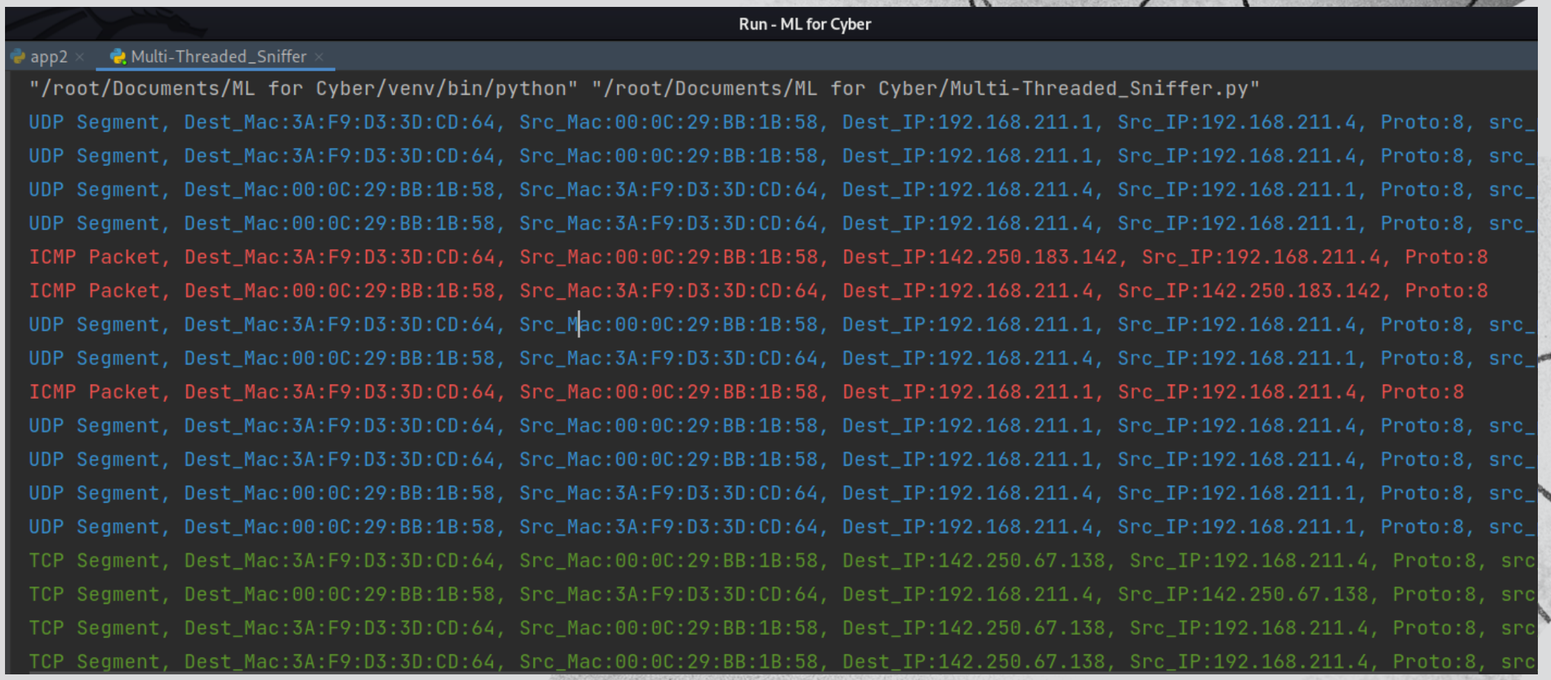
Feature are extracted from the packets stored in pcap file using tshark

Label along with data is stored in a CSV file which will be used by our model for training

```
Label,IPLength,IPHeaderLength,IPFlags,TTL,Protocol,IPID,IPchecksum
TCP_Mobile,64,20,0x00004000,64,6,0x00000000,0x0000fcf3,52368,443,0
TCP_Mobile,52,20,0x00004000,64,6,0x00000000,0x0000fcff,52368,443,1
TCP_Mobile,292,20,0x00004000,64,6,0x00000000,0x0000fc0f,52368,443,
TCP_Mobile,52,20,0x00004000,64,6,0x00000000,0x0000fcff,52368,443,2
TCP_Mobile,52,20,0x00004000,64,6,0x00000000,0x0000fcff,52368,443,2
TCP_Mobile,178,20,0x00004000,64,6,0x00000000,0x0000fc81,52368,443,
TCP_Mobile,52,20,0x00004000,64,6,0x00000000,0x0000fcff,52368,443,3
TCP_Mobile,381,20,0x00004000,64,6,0x00000000,0x0000fbb6,52368,443,
TCP_Mobile,52,20,0x00004000,64,6,0x00000000,0x0000fcff,52368,443,6
TCP_Mobile,52,20,0x00004000,64,6,0x00000000,0x0000fcff,52368,443,6
TCP_Mobile,52,20,0x00004000,64,6,0x00000000,0x0000fcff,52368,443,6
```

STEP 4 : BUILDING A SNIFFER

Packet sniffer is a computer program or computer hardware such as a packet capture appliance, that can intercept and log traffic that passes over a computer network or part of a network. Packet capture is the process of intercepting and logging traffic



STEP 5 : CLASSIFICATION

```
def print_stats_metrics(y_test, y_pred):
    print('Accuracy: %.2f' % accuracy_score(y_test,y_pred) )
    confmat = confusion_matrix(y_true=y_test, y_pred=y_pred)
    print ("confusion matrix")
    print(confmat)
    print (pd.crosstab(y_test, y_pred, rownames=['True'], colnames=['Predicted'], margins=True))
    print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred))
    print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred))
    print('F1-measure: %.3f' % f1_score(y_true=y_test, y_pred=y_pred))
```

Function to print the Accuracy, Confusion, Precision, Recall, F1-Measure

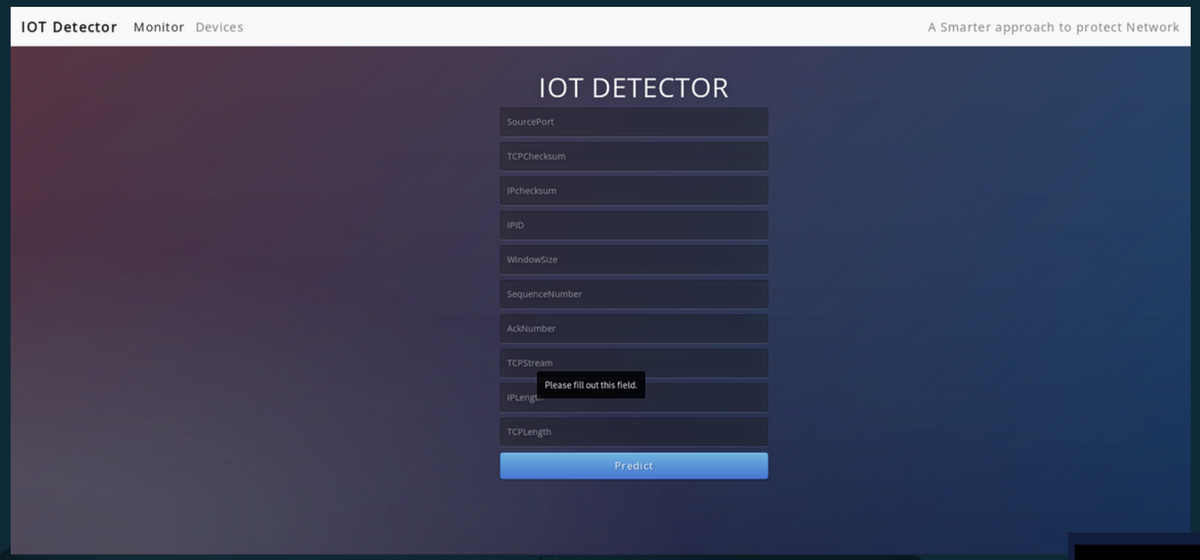
use the Logistic Regression to test the model

```
########################Logistic Regression###############################
print("########################Logistic Regression###############################")
clfLog = LogisticRegression()
clfLog.fit(x_train,y_train)
predictions = clfLog.predict(x_test)
print_stats_metrics(y_test, predictions)
```

```
########################Naive Bayes########################
print("########################Naive Bayes########################")
clfNB = GaussianNB()
clfNB.fit(x_train,y_train)
predictions = clfNB.predict(x_test)
print_stats_metrics(y_test, predictions)
```

use the Naive Bayes class to test the model

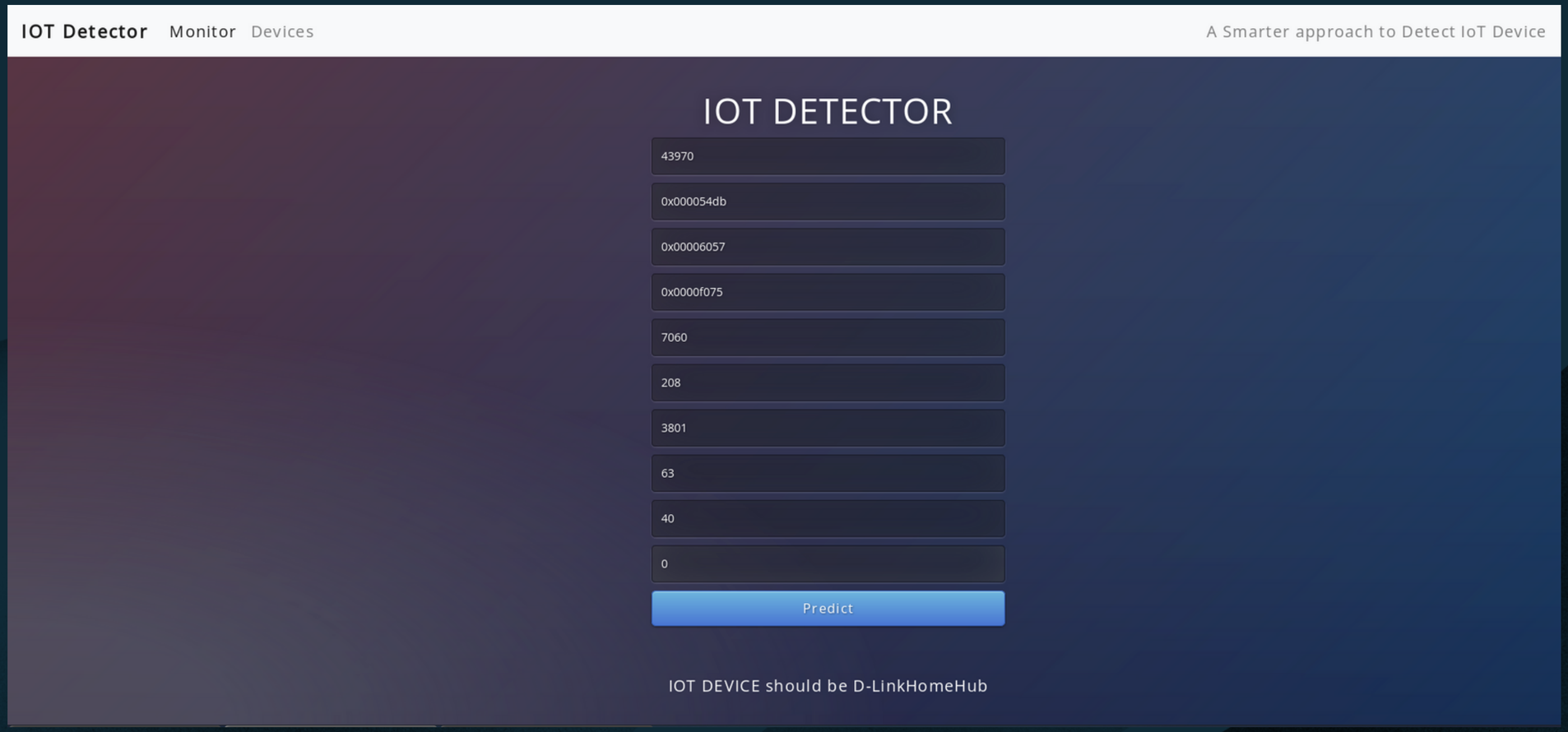
STEP 6 : FLASK INTEGRATION



Flask application is build and Prediction on given input is taken care by our trained model

Packet Sniffer, this will get update in every 4 sec and plot a real time graph of different type of packets

STEP 6 :RESULT



For the given input values our model will predict the IoT Device

THANK YOU !!!!