IoT Device Classification from Network traffic log using Machine Learning

Dr Gaurav Singal

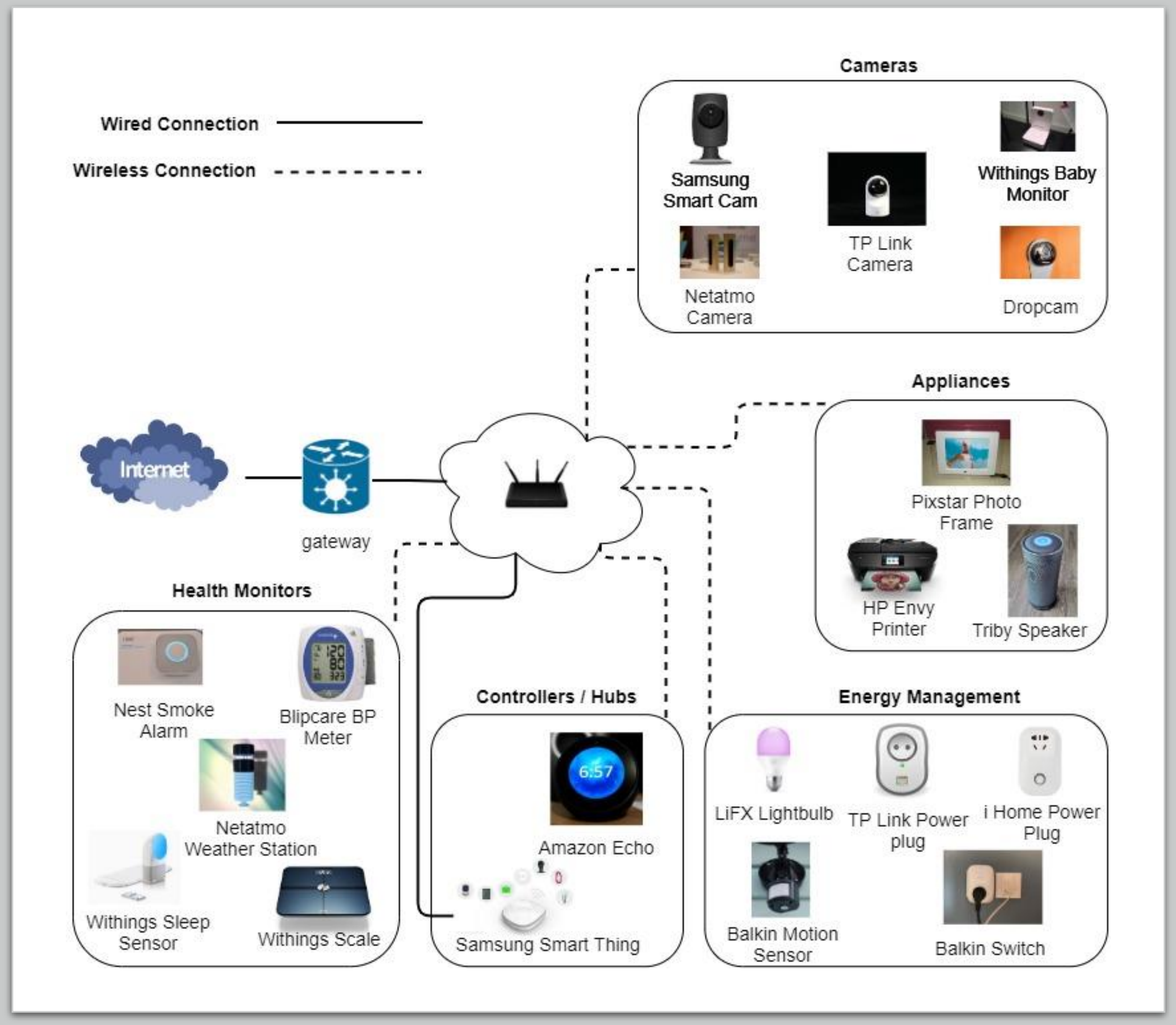
Netaji Subhas University of Technology, Delhi


07/09/2018 Fri 16:01:56

IoT Security

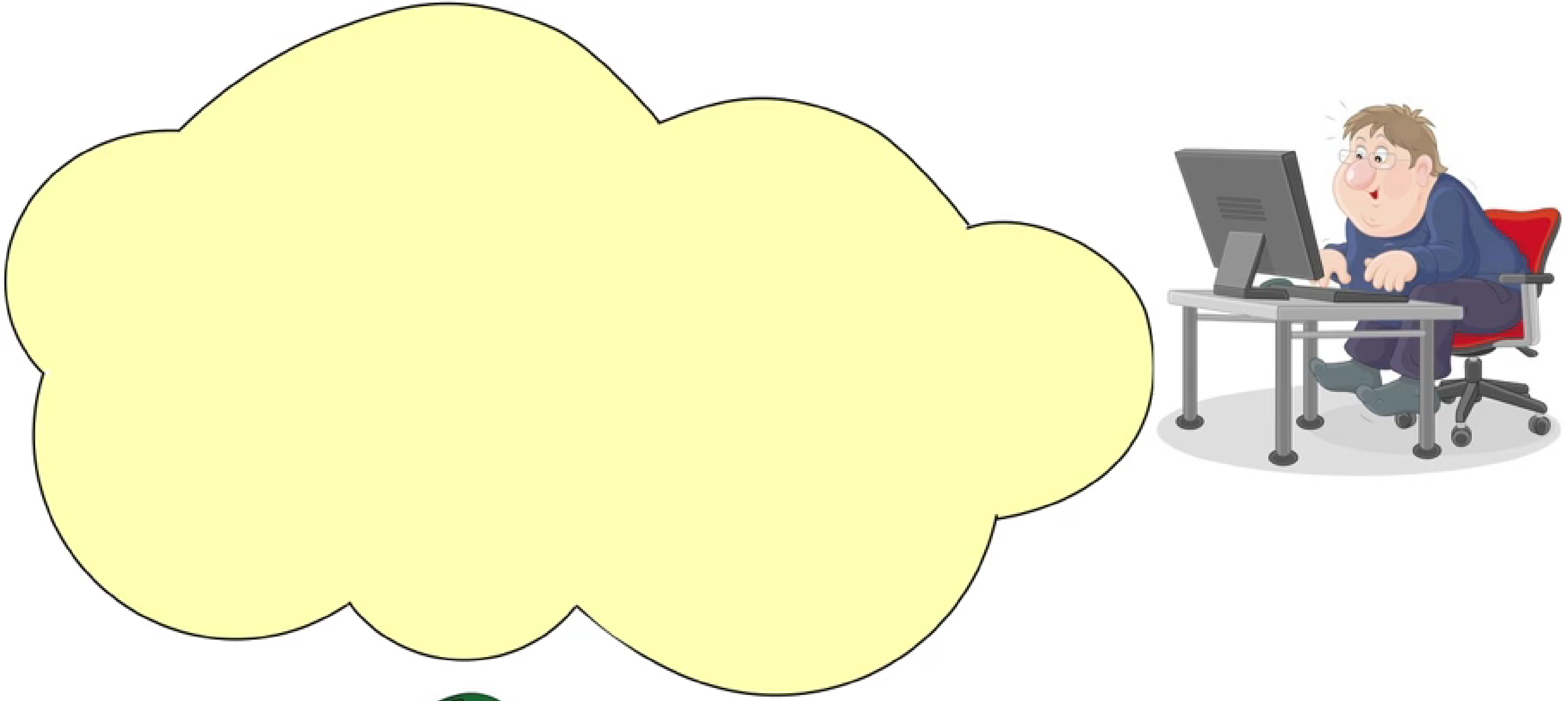
IoT network traffic classification

- Mechanism to **categorizes** the embedded **devices** connected to the internet and security **attacks** in the network.
- Beneficial to ensuring security, reliability, quality of services (QoS) and complete working of IoT devices.



http://gauravsingal.in/dsci_project.html

Traffic analysis



Motivation

- Easy to hack, can easy to compromise and become a part of botnet[3,4].
- Need to classify the IoT devices[5].

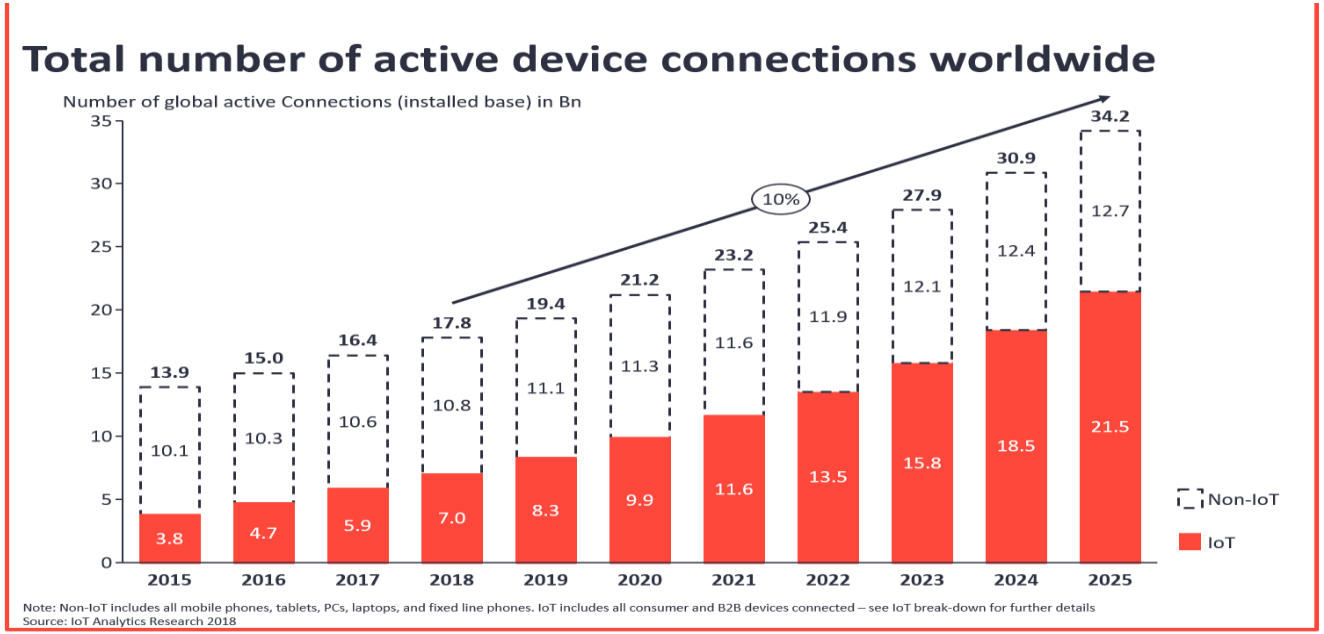


Fig: Number of IoT devices connected worldwide

Figure courtesy : https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/

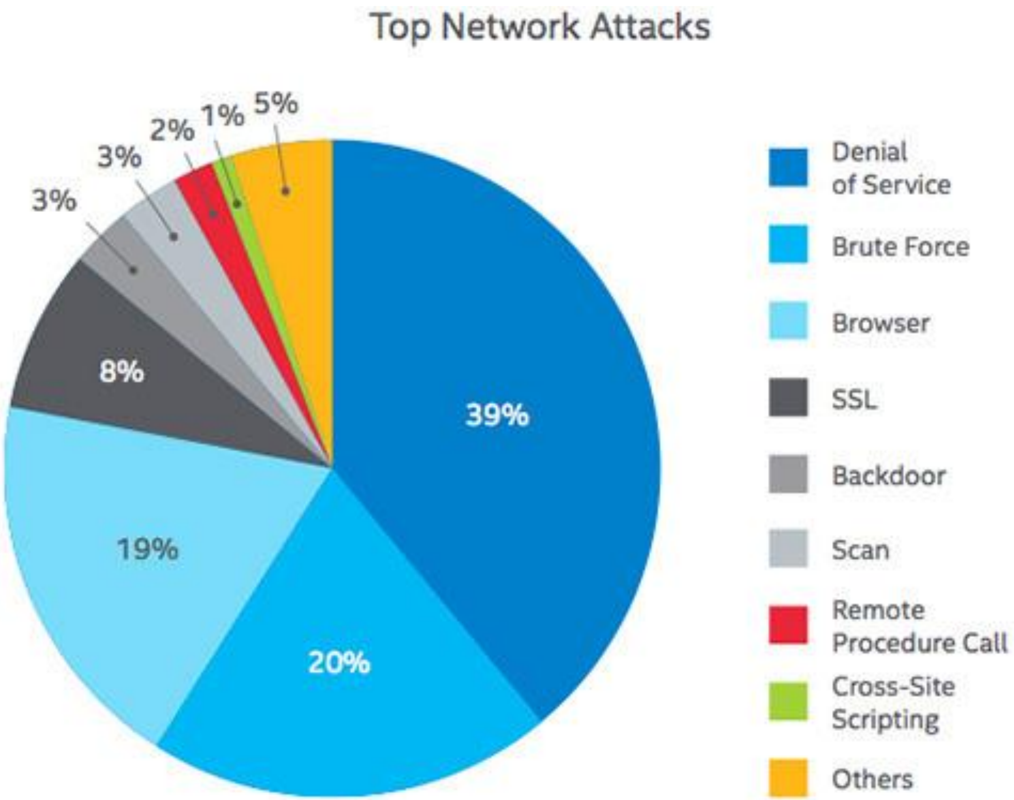


Fig.: IoT Attacks Statistics

Figure courtesy : https://www.cisecurity.org/blog/top-10-malware-july-2020/

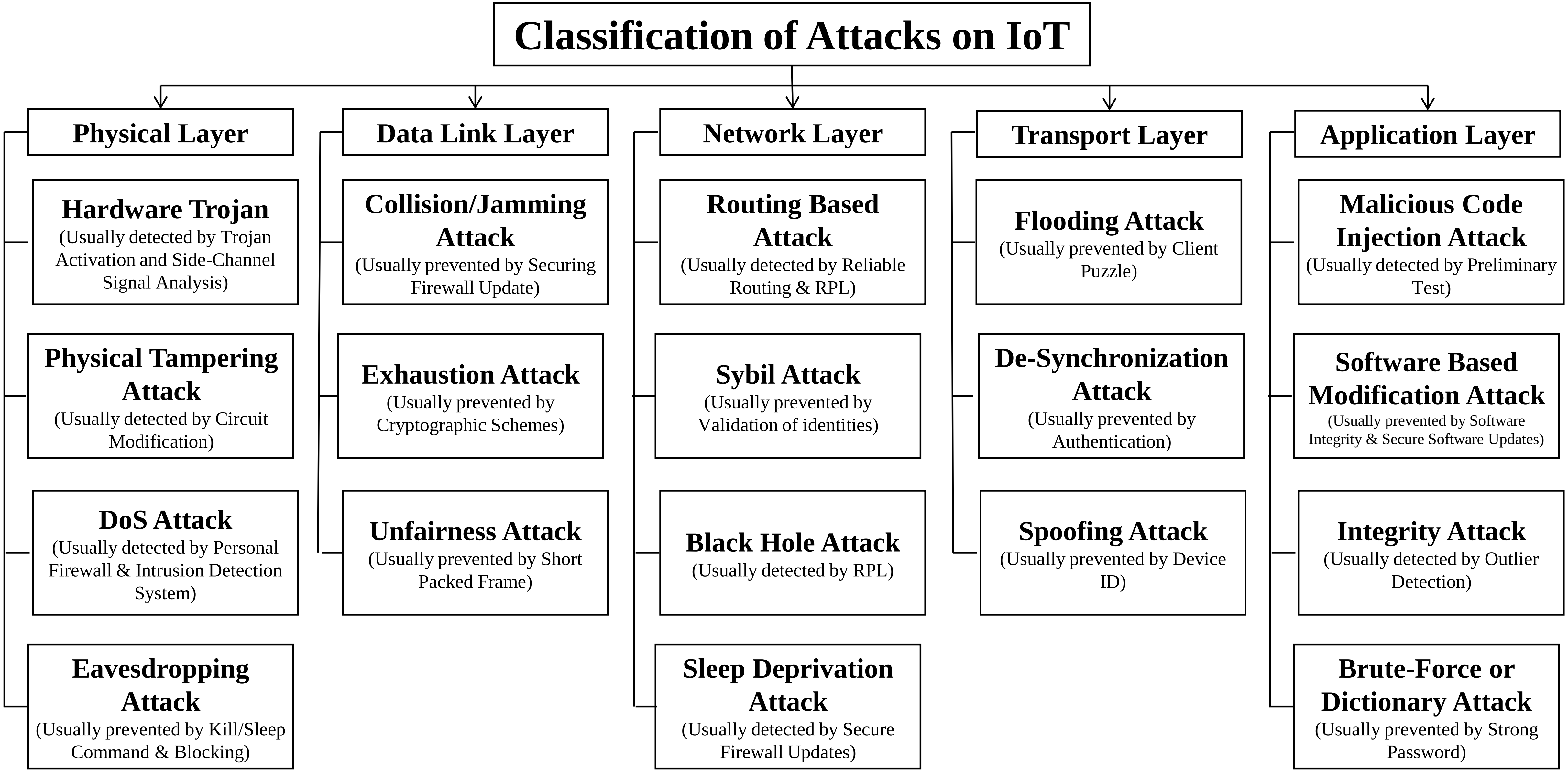


Fig.: Layer-wise IoT Attacks [6] [7] [8]

How DDoS is working?

Issues in IoT Networks Traffic classification

- **Variety** and **limited** number of IoT devices for classification.
- **Overlapping** instances problem increases as traffic increases from IoT devices.
- Issues in IoT devices classification due to periodic **updates**.
- Limited number of large **datasets** available publicly.
- **User** security and **privacy** Issues by data breaching.
- **Unbalanced** traffic from IoT devices (biased).
- **Unknown** (new) device and attacks in IoT network traffic.

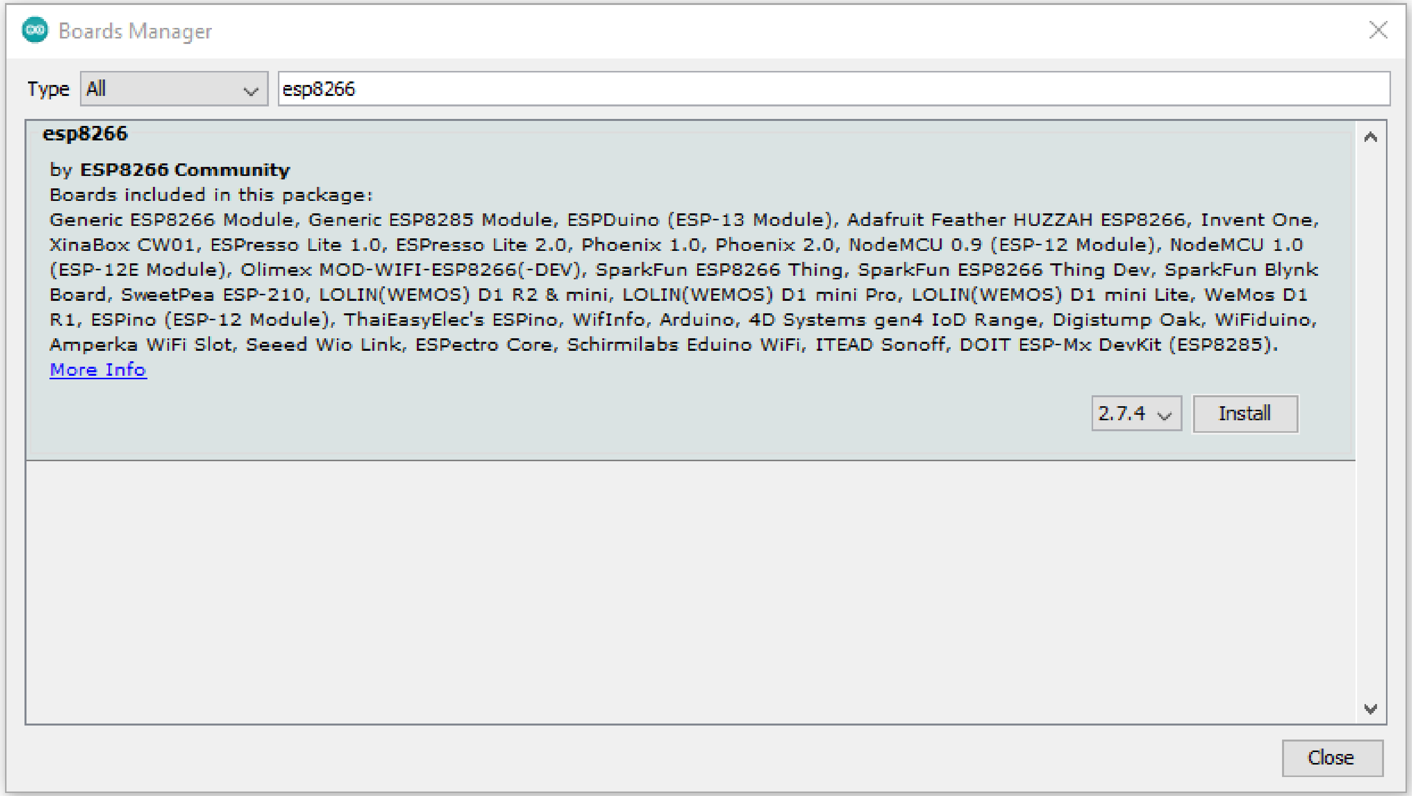
Capturing Packets though MQTT Protocol

Generating Dataset

Setup Installation

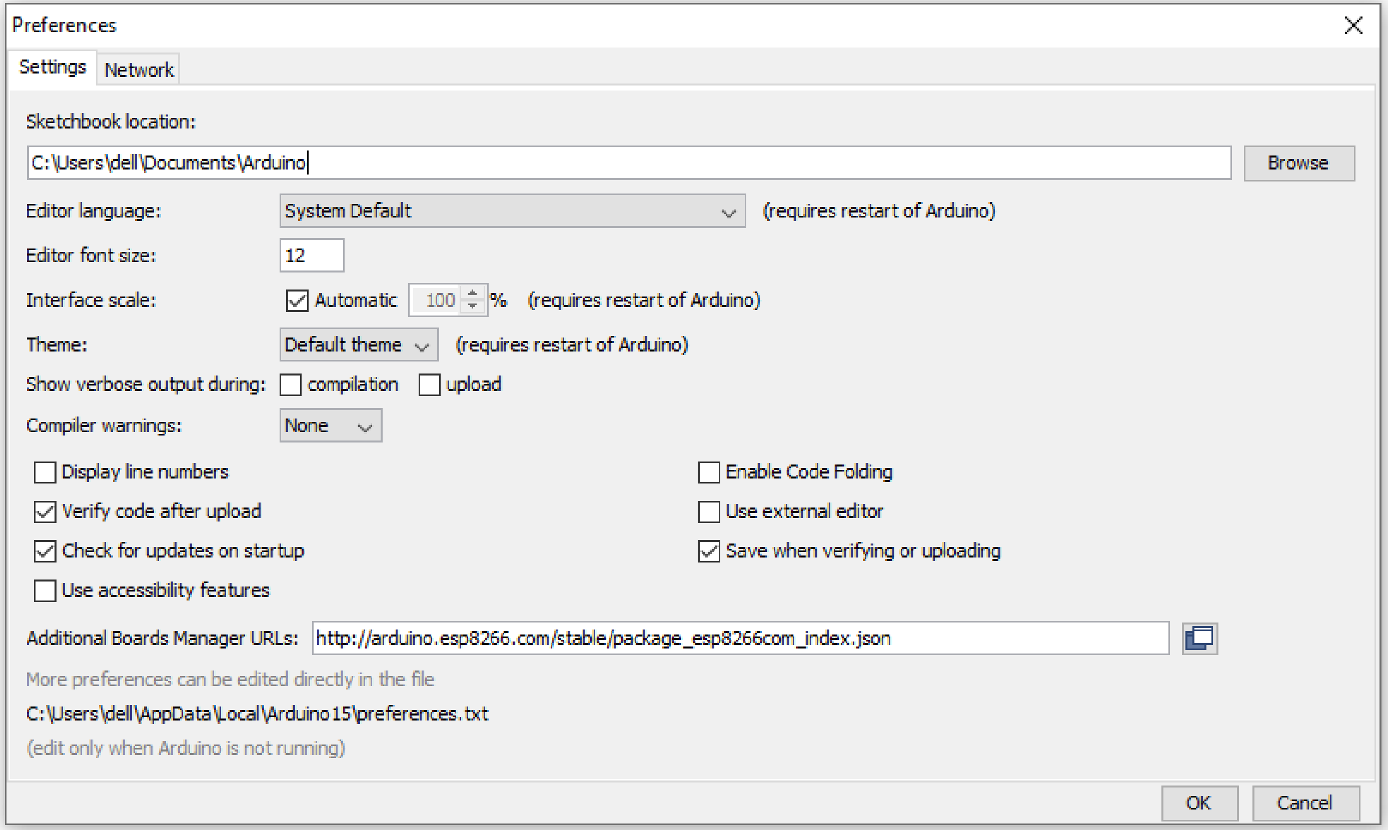
- Install Arduino IDE :

- Go to **File** --> **Preference** --> paste the URL in additional board manager URL -->

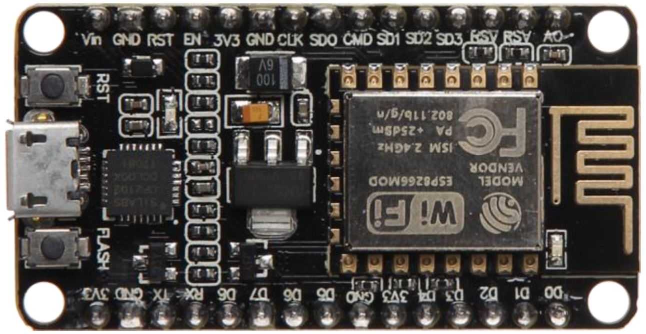


Continue

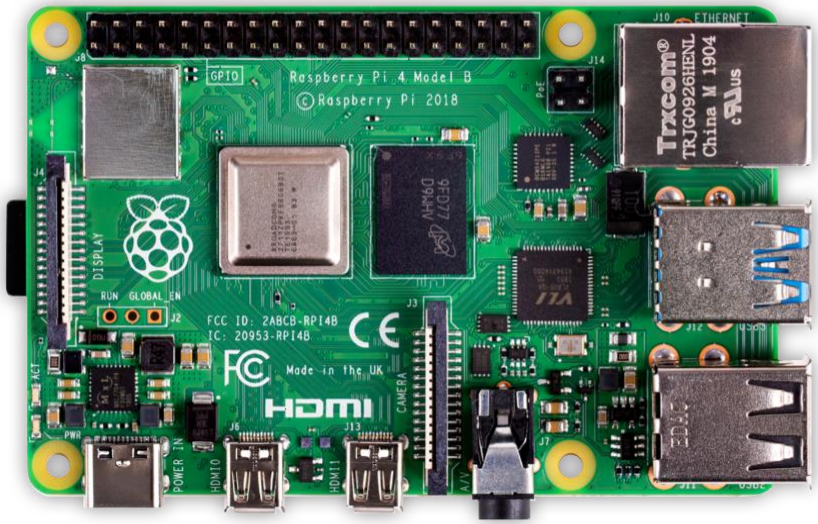
- Go to **Tools --> Board --> Board Manager -->** Search for Esp8266 and install library
- **Go to Sketch** --> **Include Library** --> **Manage Library -->** Search MQTT and download **Adafruit MQTT Library, EspMQTTClient**
- Install Wireshark:
  - sudo apt-get update
  - sudo apt-get upgrade
  - sudo apt-get install wireshark



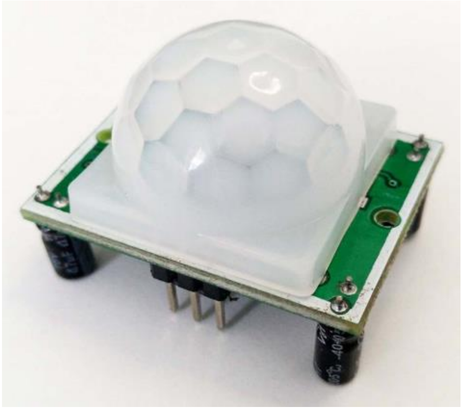
Devices Used



NodeMCU (ESP8266)

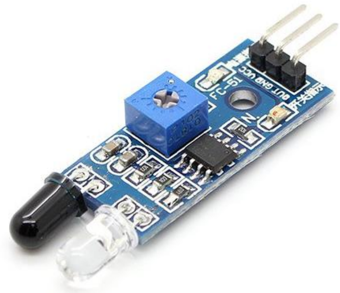


Raspberry Pi 4



PIR Sensor

Ultrasonic Sensor

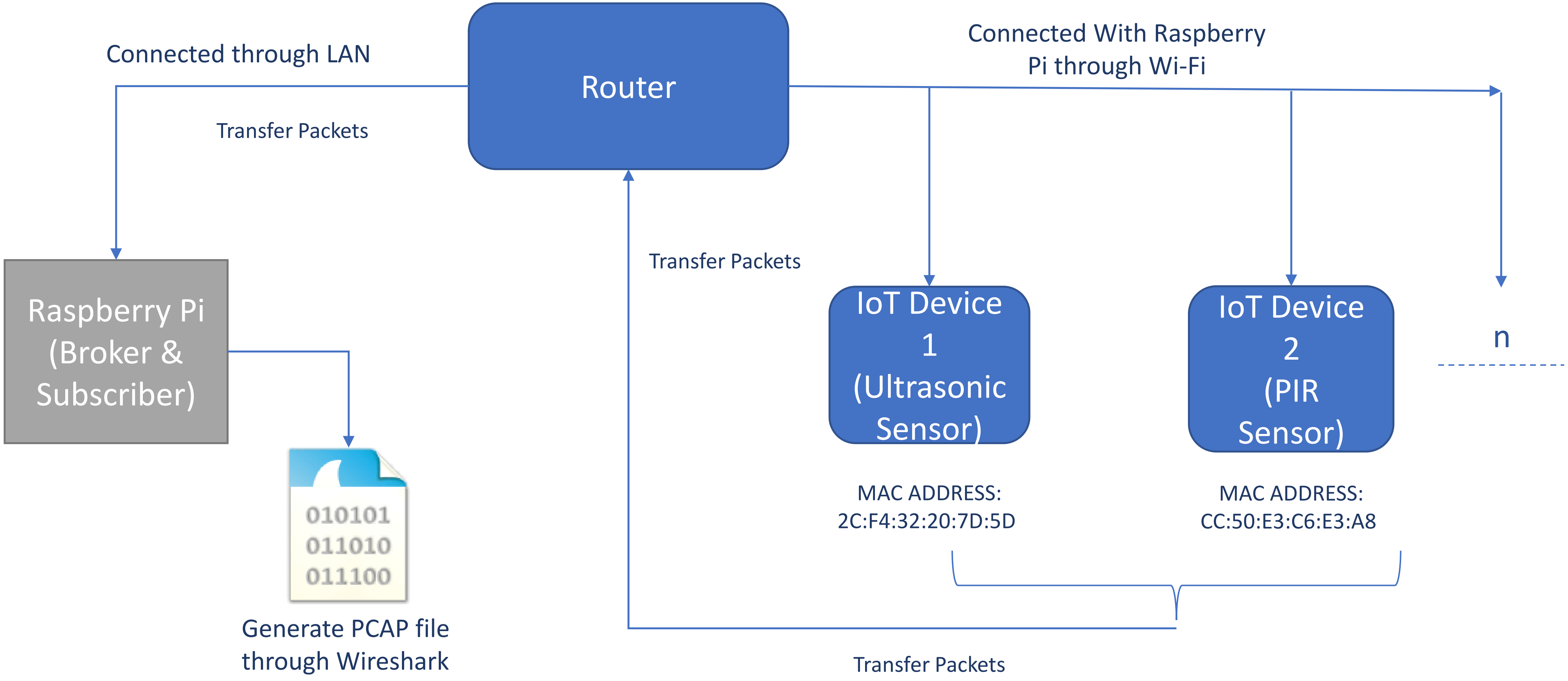


IR Sensor

About MQTT

- Message Queue Telemetry Transport Protocol
- It has "Publish/Subscriber architecture. Device can publish any topics and can also subscribe for any updates
- It runs over TCP
- Message size is Small
- MQTT session divided into four stages:
    - Connection
    - Authentication
    - Communication
    - Termination
- It is many-to-many communication protocol for passing messages between multiple clients through a central broker.
- Message format is binary with 2 Byte header

Workflow

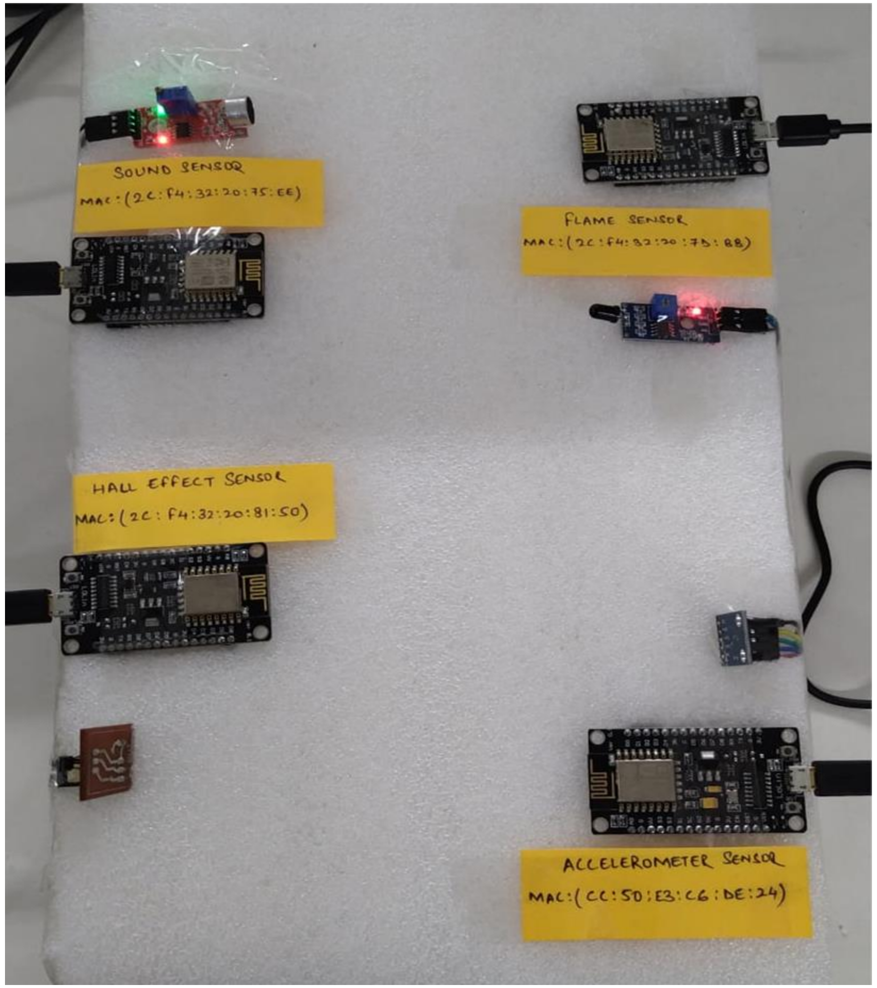

Connected through LAN
Router
Connected With Raspberry Pi through Wi-Fi
Transfer Packets
Transfer Packets
Raspberry Pi (Broker & Subscriber)
IoT Device 1 (Ultrasonic Sensor)
IoT Device 2 (PIR Sensor)
n
010101
011010
011100
MAC ADDRESS: 2C:F4:32:20:7D:5D
MAC ADDRESS: CC:50:E3:C6:E3:A8
Generate PCAP file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

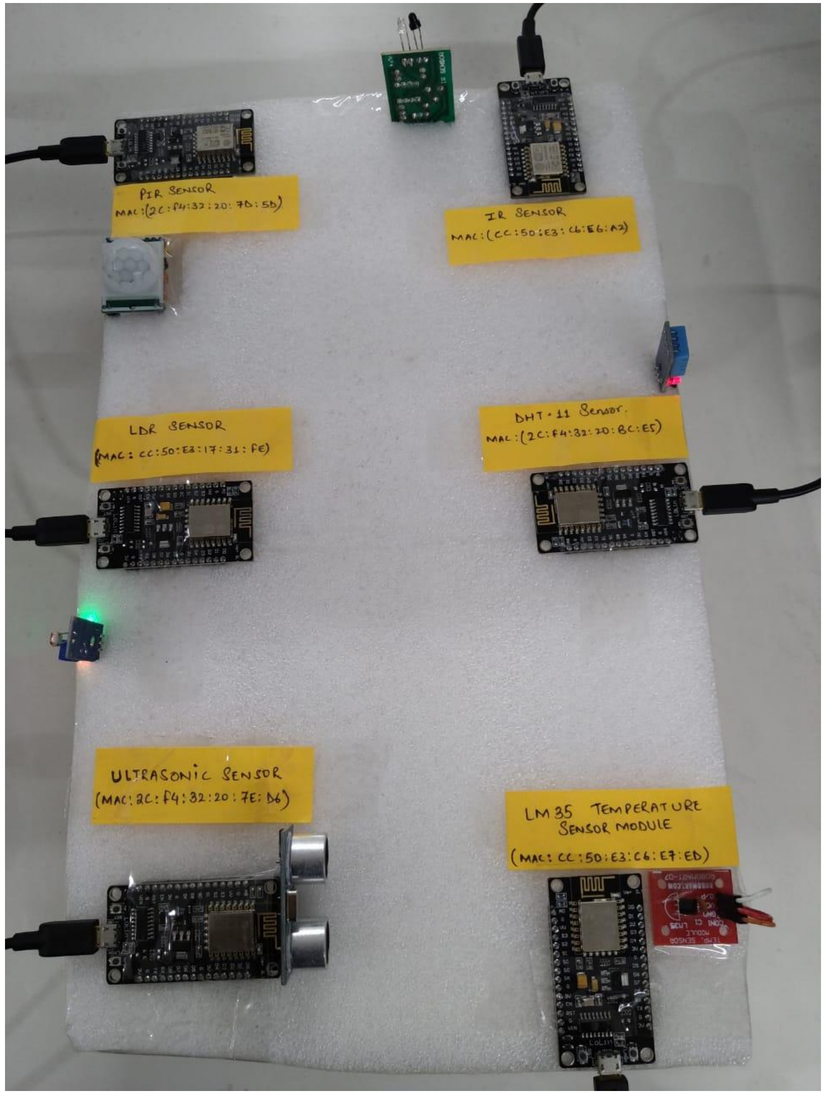
Description

MQTT Broker Setup

- Install MQTT broker
  - sudo apt-get install mosquitto
- Install command line clients in case for debugging
  - sudo apt-get install mosquitto-clients –y
- Open the Mosquitto MQTT broker configuration
  - sudo nano /etc/mosquitto/mosquitto.conf
- Create new user with username and password
  - sudo mosquitto_passwd -c /etc/mosquitto/pwfile username
  - sudo mosquitto_passwd -c /etc/mosquitto/pwfile password
- See current status of MQTT broker
  - sudo systemctl status mosquitto
- Stop Mosquitto:
  - sudo systemctl stop mosquitto

Experiment Setup





S.NO.	IOT DEVICE NAME	MAC ADDRESS	PROTOCOLS	APPLICATION AREA
1	Ultrasonic Sensor 1	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	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

File Edit Tabs Help

```
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.
```

Output of Broker

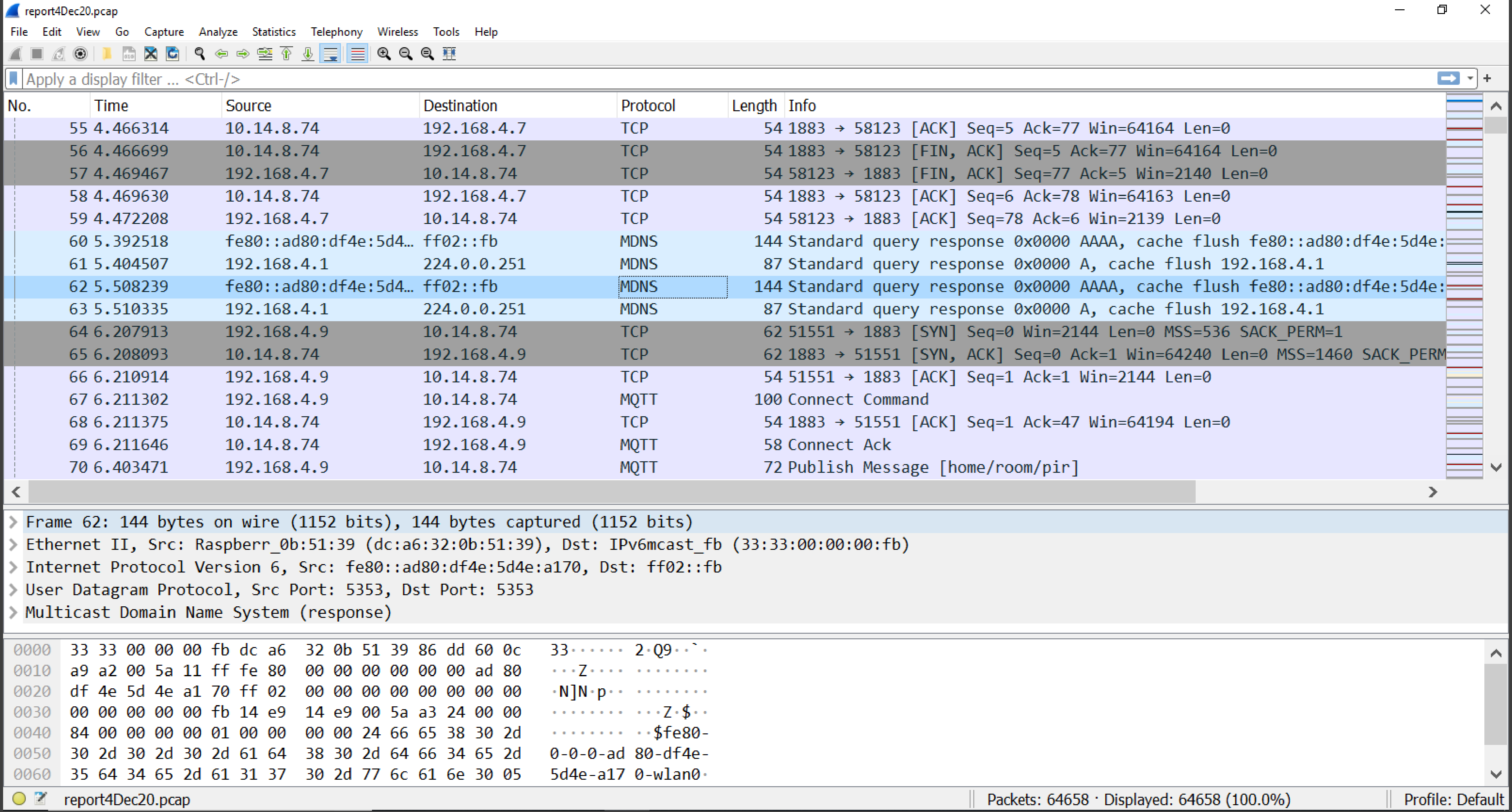
pi@raspberrypi: ~

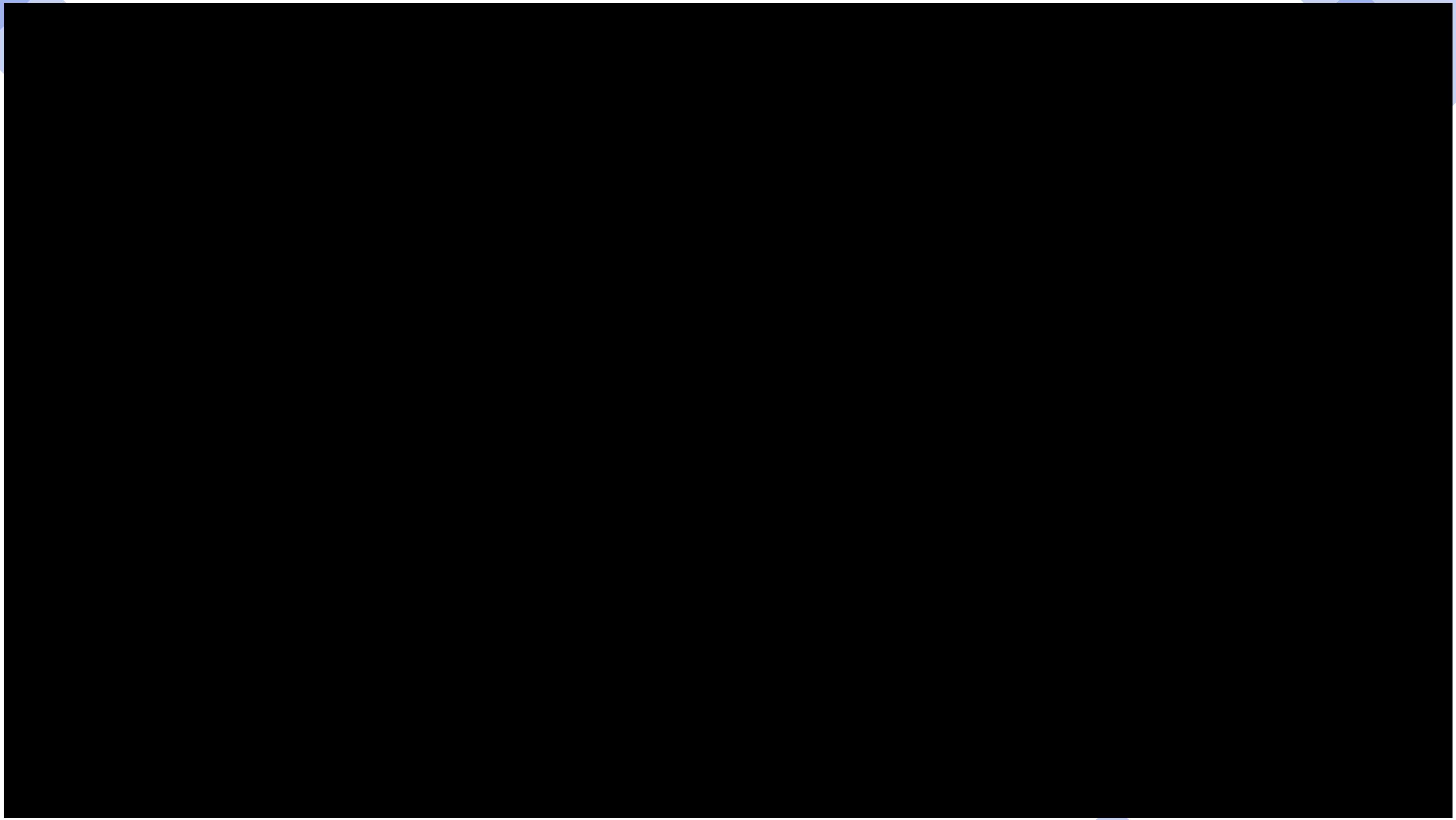
File Edit Tabs Help

```
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
```

Output of Publishers

Wireshark Report


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·
report4Dec20.pcap
Packets: 64658 · Displayed: 64658 (100.0%)
Profile: Default



IoT Traffic Classification

Demo

Machine Learning in IoT

- Machine learning in IoT [22] [23] [24] as follows:

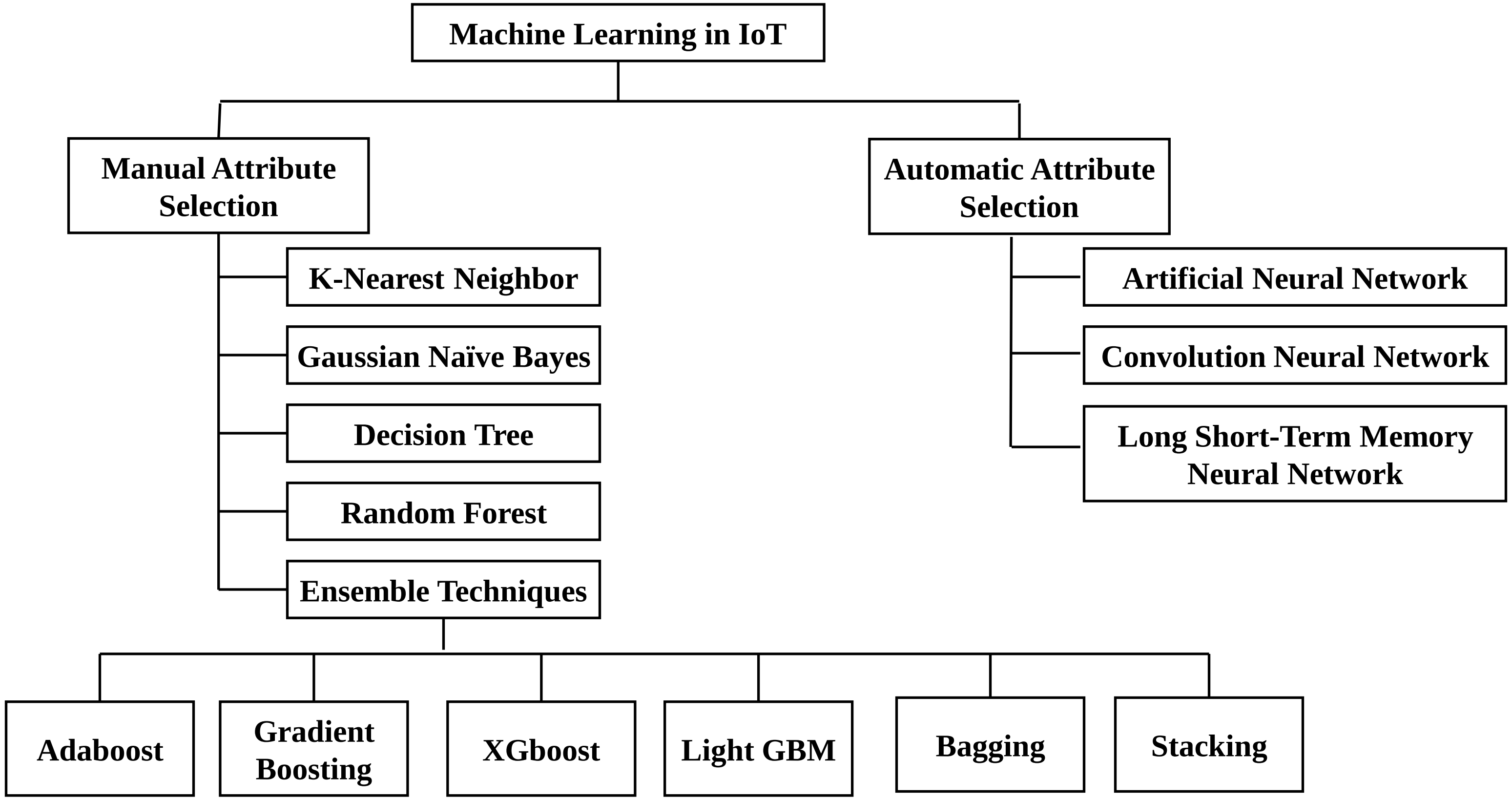


Fig. 5: Machine Learning Techniques used for IoT Classification

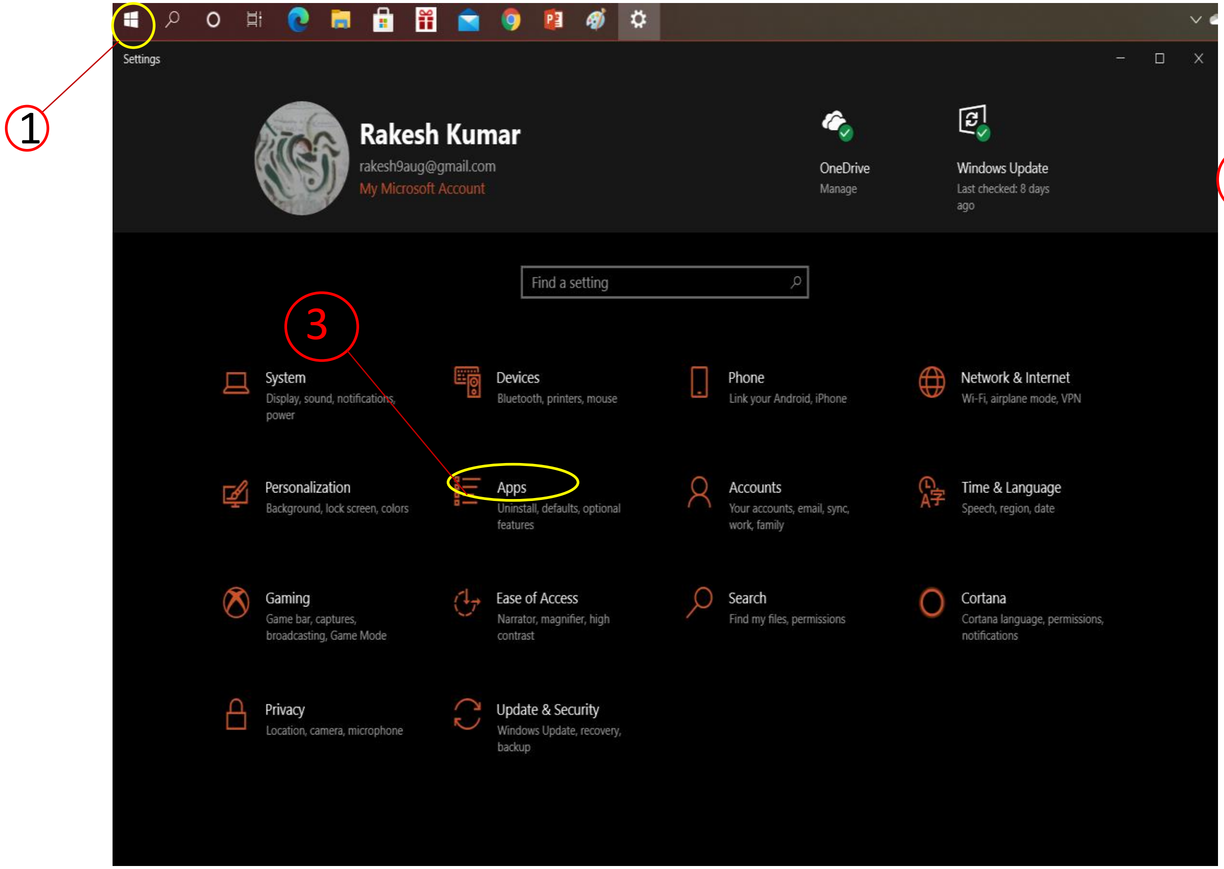
[5]. Sivanathan, Arunan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. "Classifying IoT devices in smart environments using network traffic characteristics." IEEE Transactions on Mobile Computing, vol. 18, pp. 1745-1759, 2019

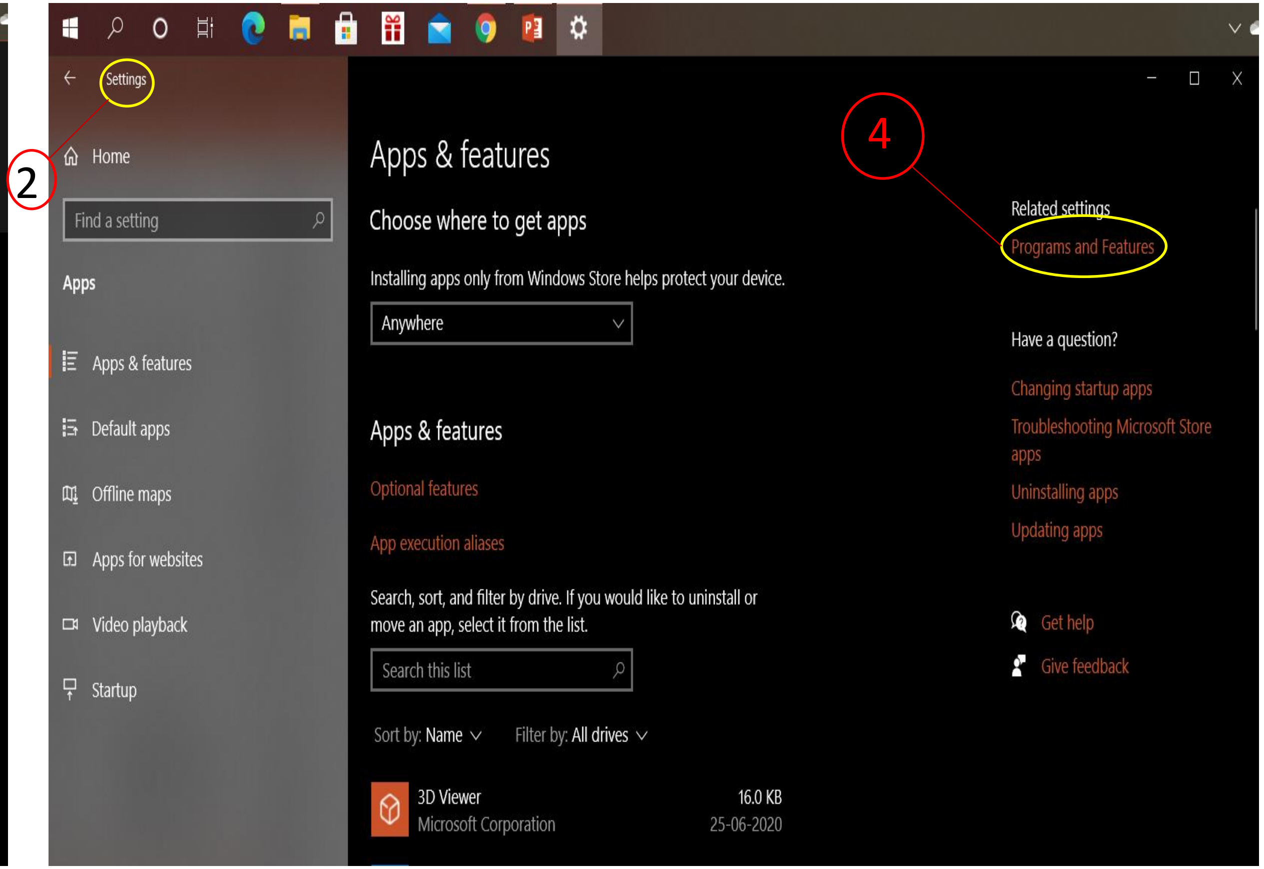
[23]. Pinheiro, Antônio J., Jeandro de M. Bezerra, Caio AP Burgardt, and Divanilson R. Campelo, "Identifying IoT devices and events based on packet length from encrypted traffic" Computer Communications, vol. 144, pp. 8-17, 2019.

[24]. A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, “Managing iotcyber-security using programmable telemetry and machine learning,”IEEE Transactions on Network and Service Management, vol. 17, pp.60–74, 2020

Installation of Kali Linux on Windows (WSL) GUI

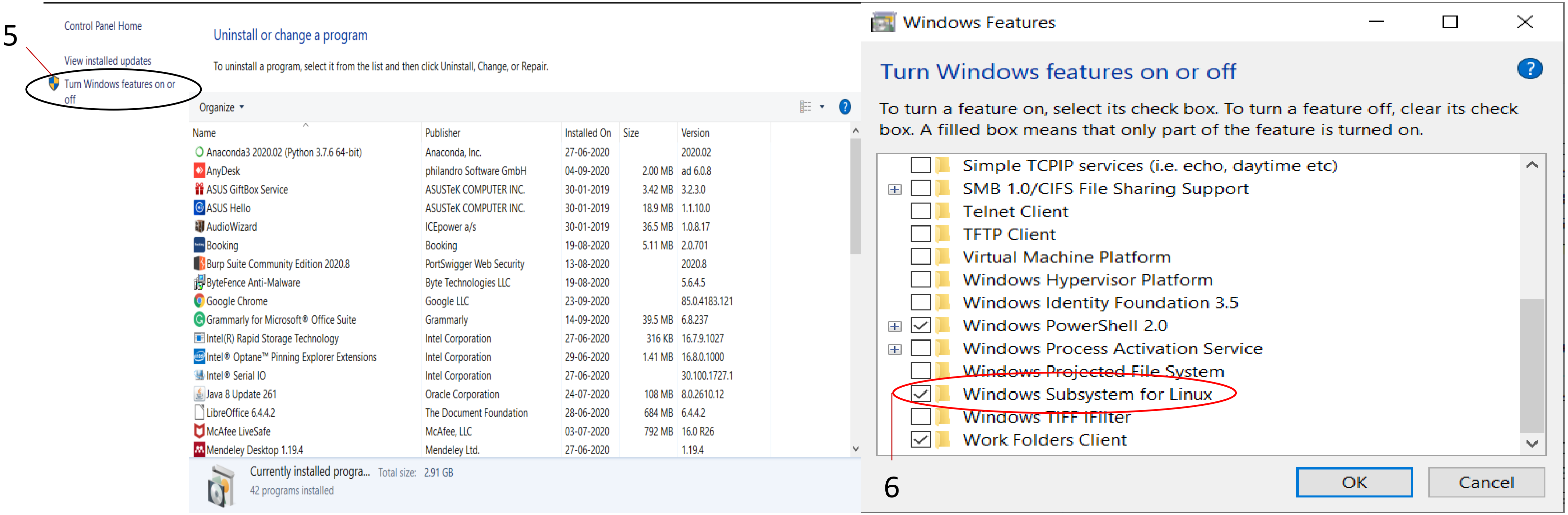
- Installation Steps as Follows:
  - Step1 : 1 Click on Windows Start Button > 2 Click Settings > 3 Click Apps > 4 Click Programs & Features (Top Right Corner) > 5 Click Turn Windows Features on or off > 6 Tick (✓) on Window Subsystem for Linux





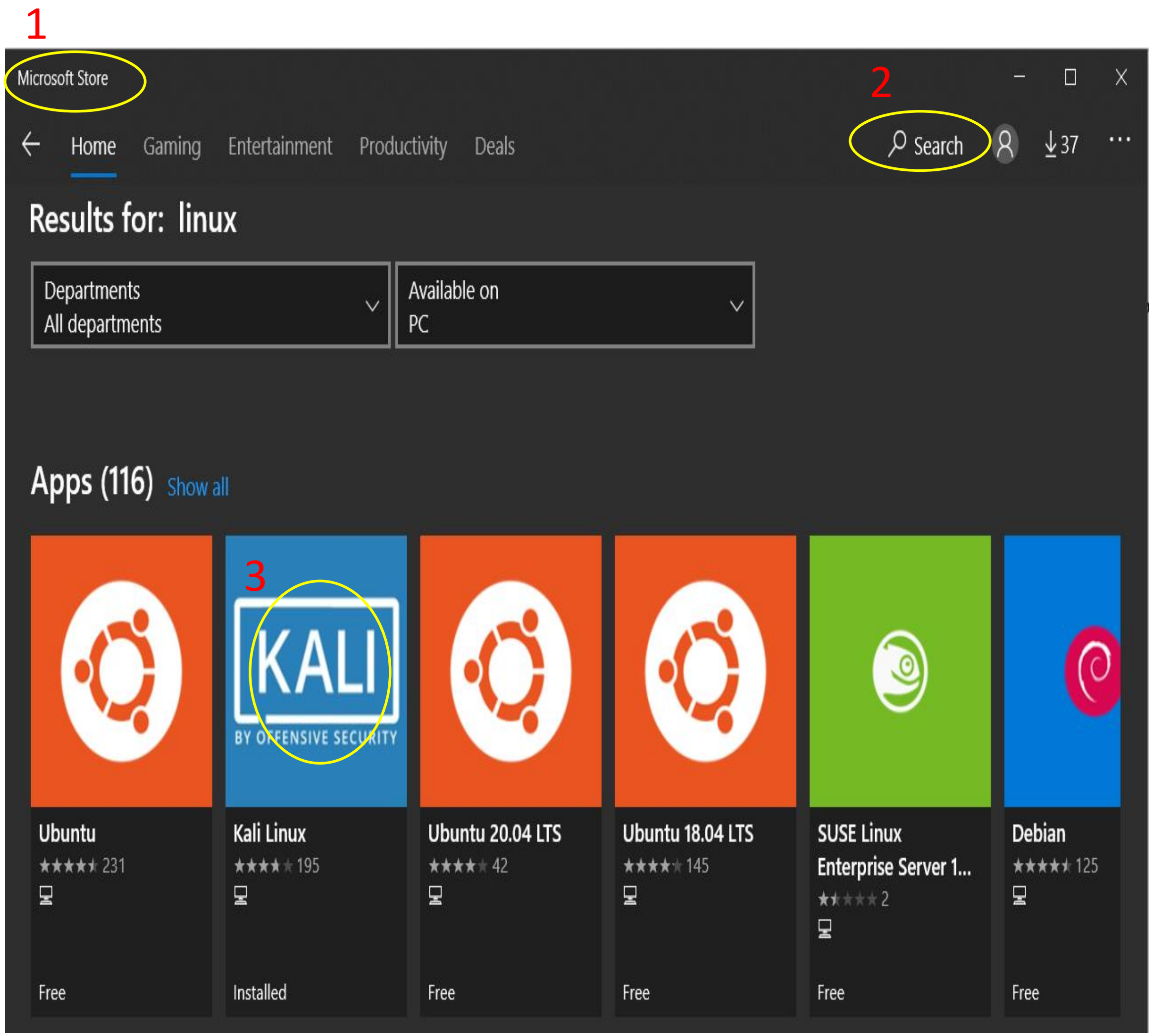
Installation of Kali Linux on Windows (WSL) GUI

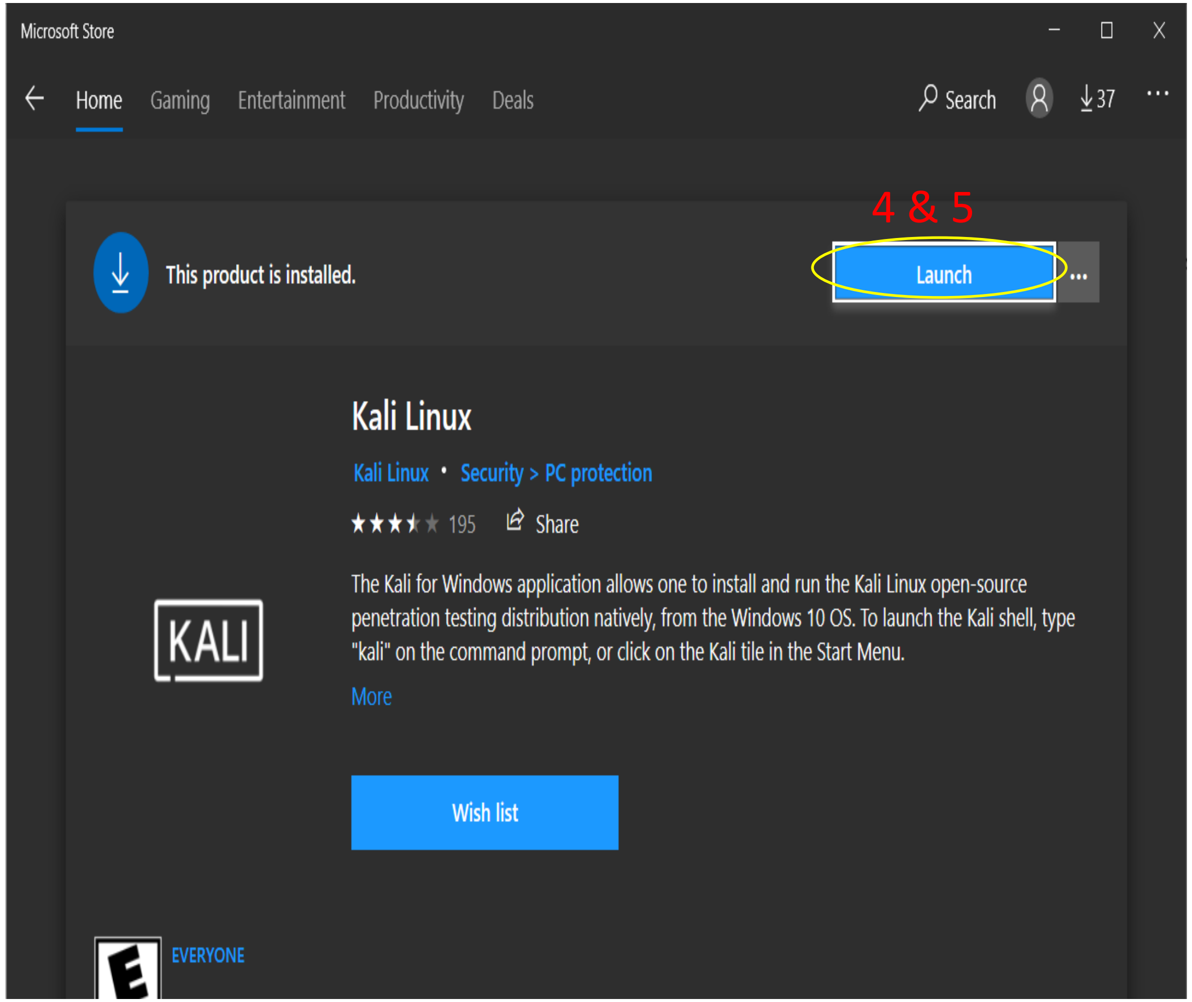
- Installation Steps as Follows:
  - Step1 : 1 Click on Windows Start Button > 2 Click Settings > 3 Click Apps > 4 Click Programs & Features (Top Right Corner) > 5 Click Turn Windows Features on or off > 6 Tick (✓) on Window Subsystem for Linux



Installation of Kali Linux on Windows (WSL) GUI

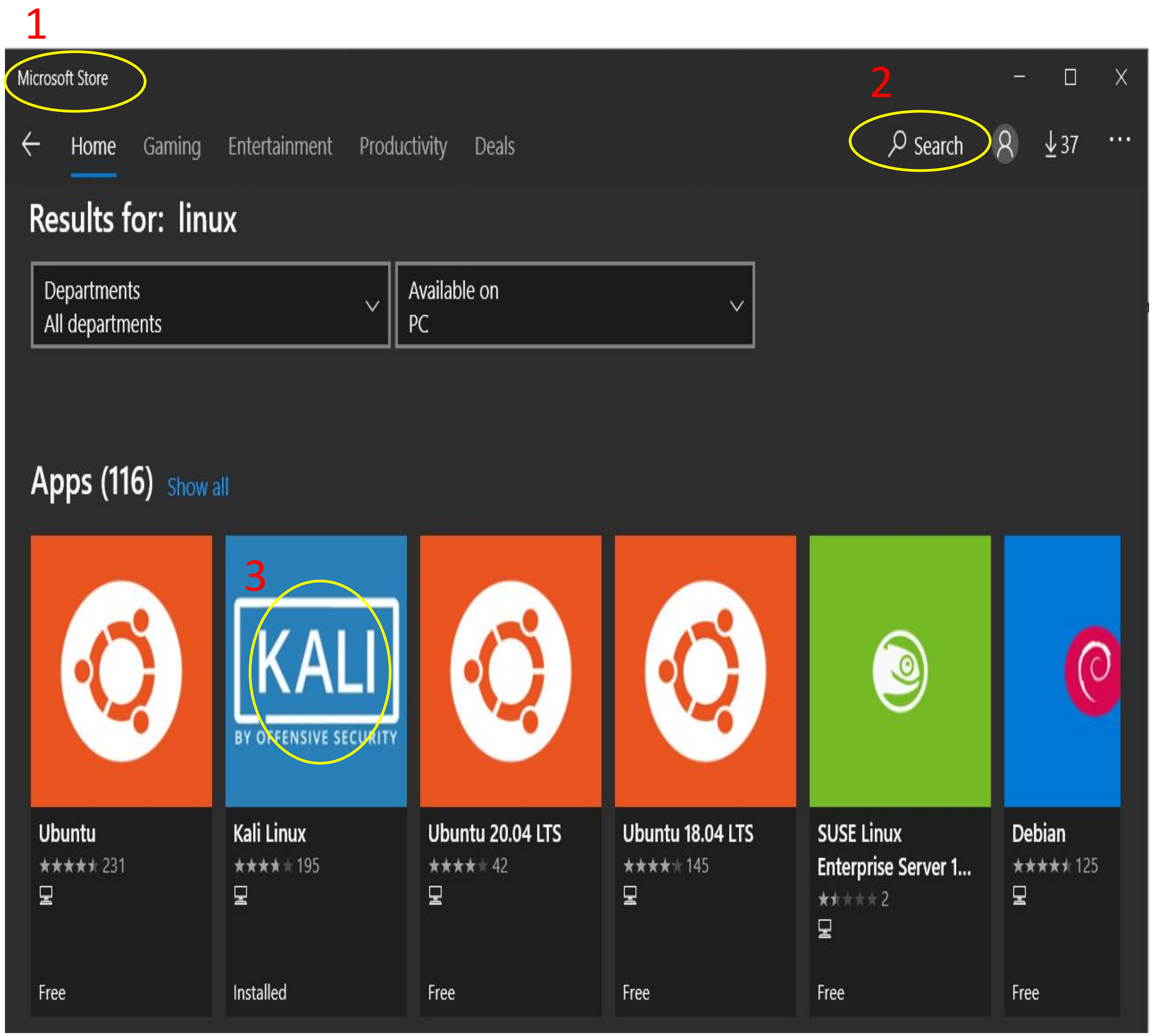
- Installation Steps as Follows:
  - Step2 : 1 Click on Microsoft Store > 2 Search Linux > 3 Click on Kali Linux App > 4 Click on Get for Download App > 5 Click on Launch

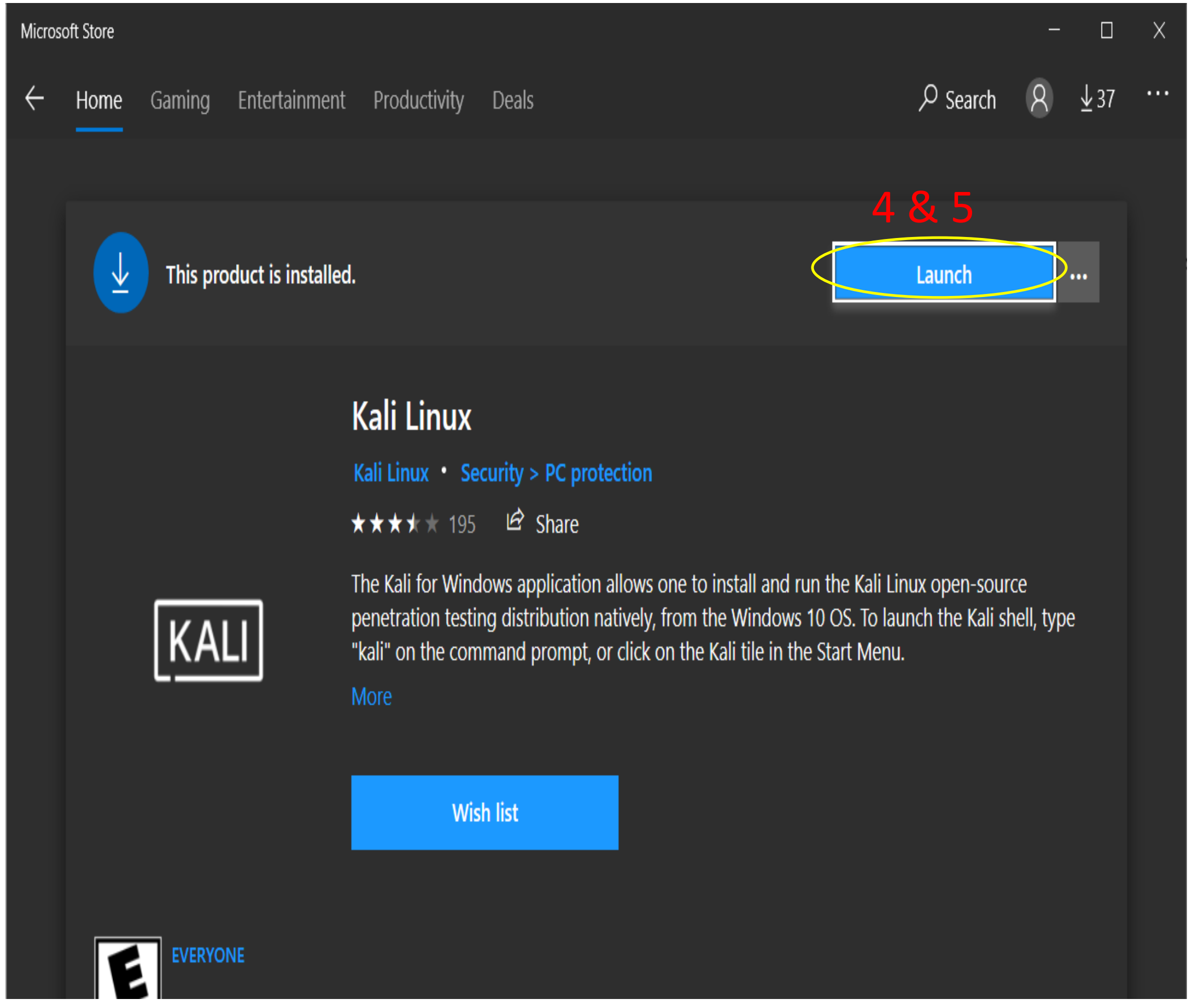




Installation of Kali Linux on Windows (WSL) GUI

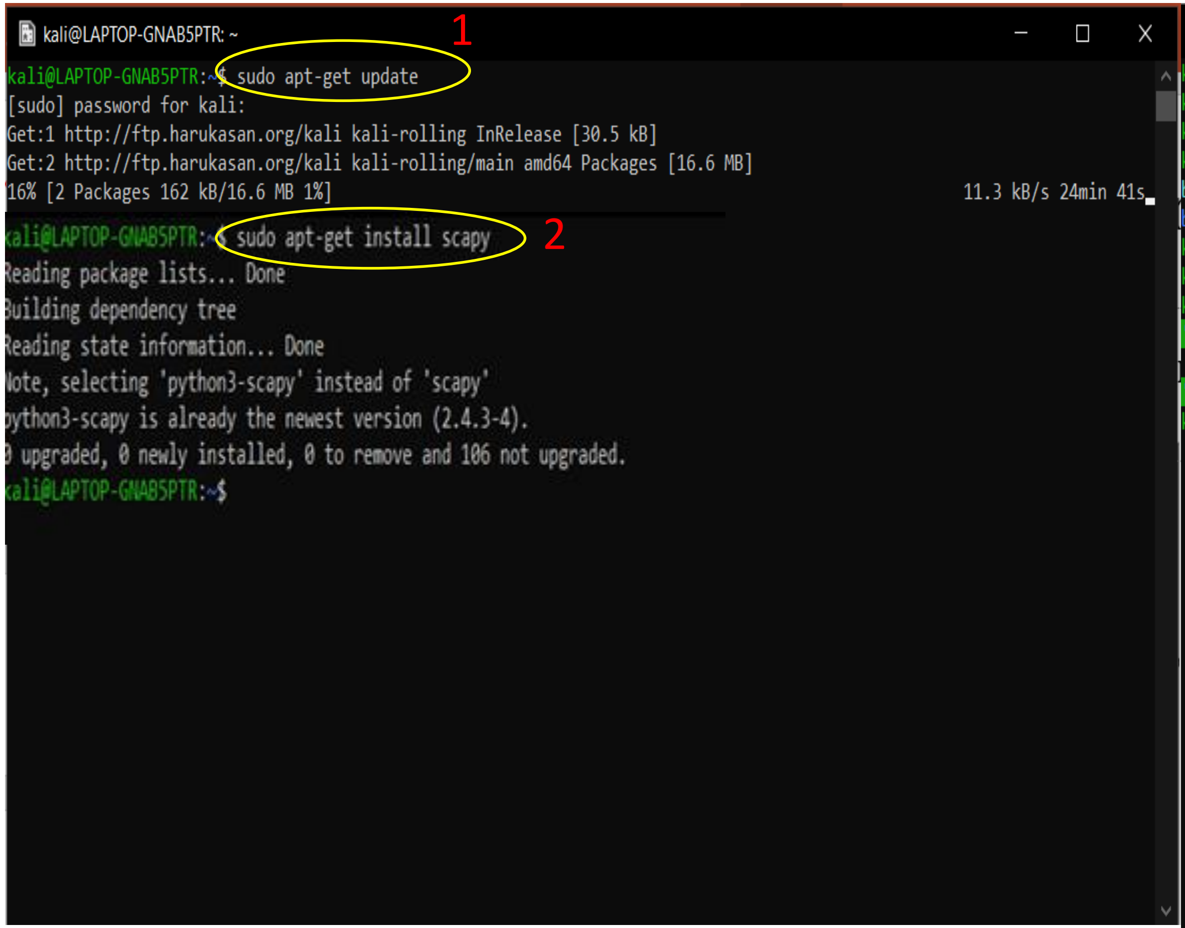
- Installation Steps as Follows:
  - ➢ Step2 : 1 Click on Microsoft Store > 2 Search Linux > 3 Click on Kali Linux App > 4 Click on Get for Download App > 5 Click on Launch

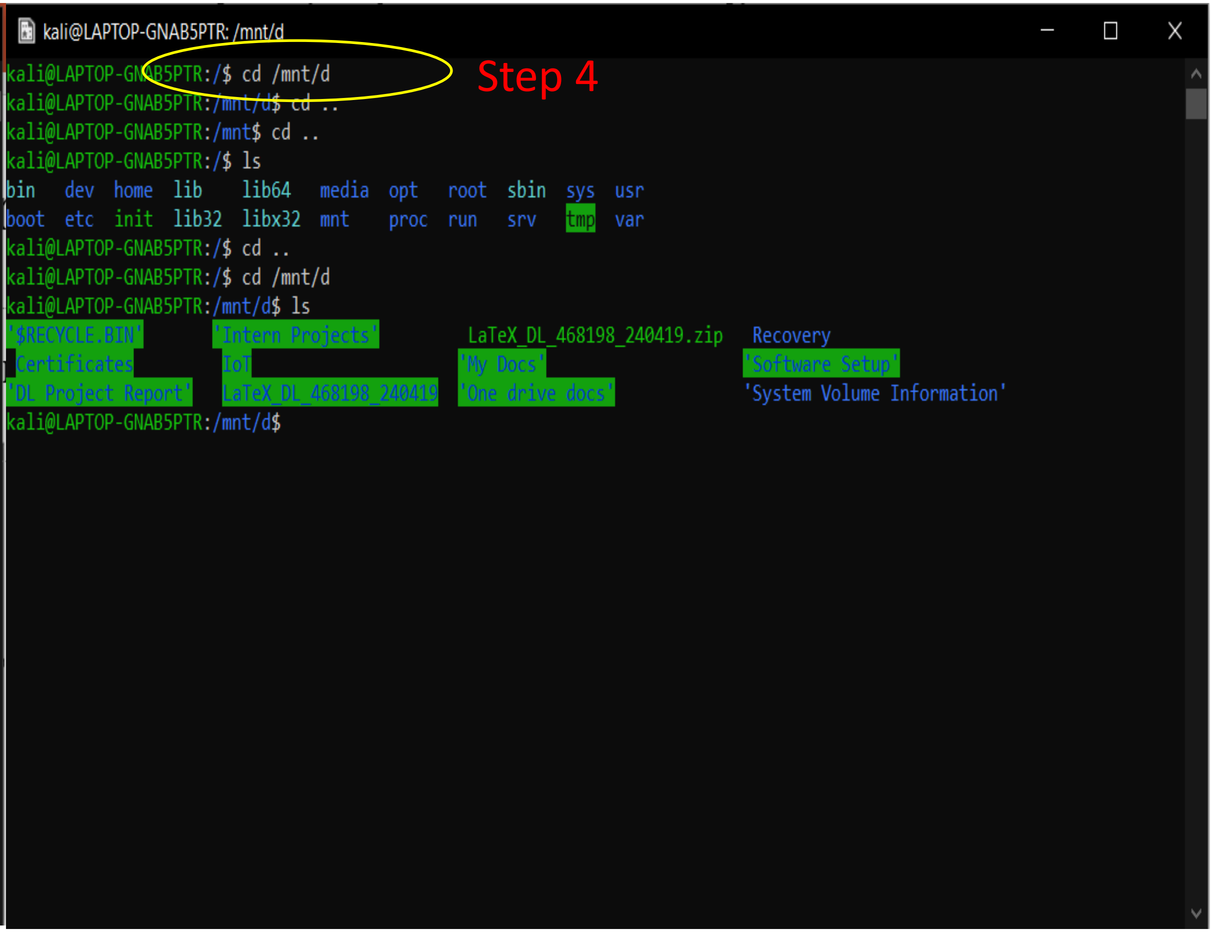




Installation of Kali Linux on Windows (WSL) GUI

- Installation Steps as Follows:
  - Step3 : 1 Update your Kali Linux > 2 Install Scapy & Use installed software
  - Step4: Mount local drives





Pre-processing of IoT Traffic

- Network Traffic Capturing: Using Wireshark/TCPdump capturing the IoT network traffic.
- Splitting: Separate the IoT devices traffic from whole network traffic traces.
- Flow Construction: Construction of two types flows such as TCP & UDP from IoT traffic.
- Feature Extraction: Extract the three types of features such as packet level, flow level & behavior level.

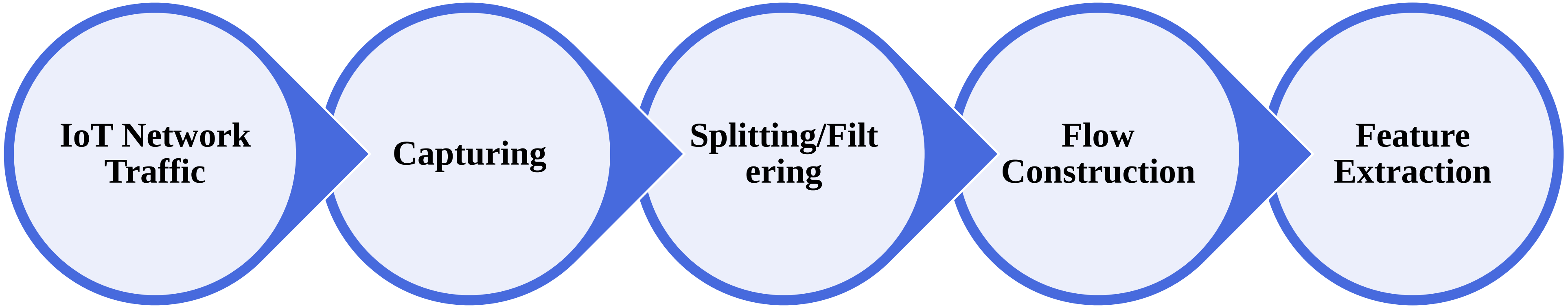


Fig. 6: Preprocessing Steps for IoT Traffic Classification

IoT Network Traffic Capturing

- Network Traffic Capturing: Using Wireshark/TCPdump capturing the IoT network traffic.

  - Download Wireshark for Windows, Linux, MAC OS :
    https://www.wireshark.org/download.html
  - Wireshark
    - ✓ Free & open-source packet analyzer
    - ✓ Network troubleshooter
    - ✓ Analysis
    - ✓ Software and communications protocol development
    - ✓ Education

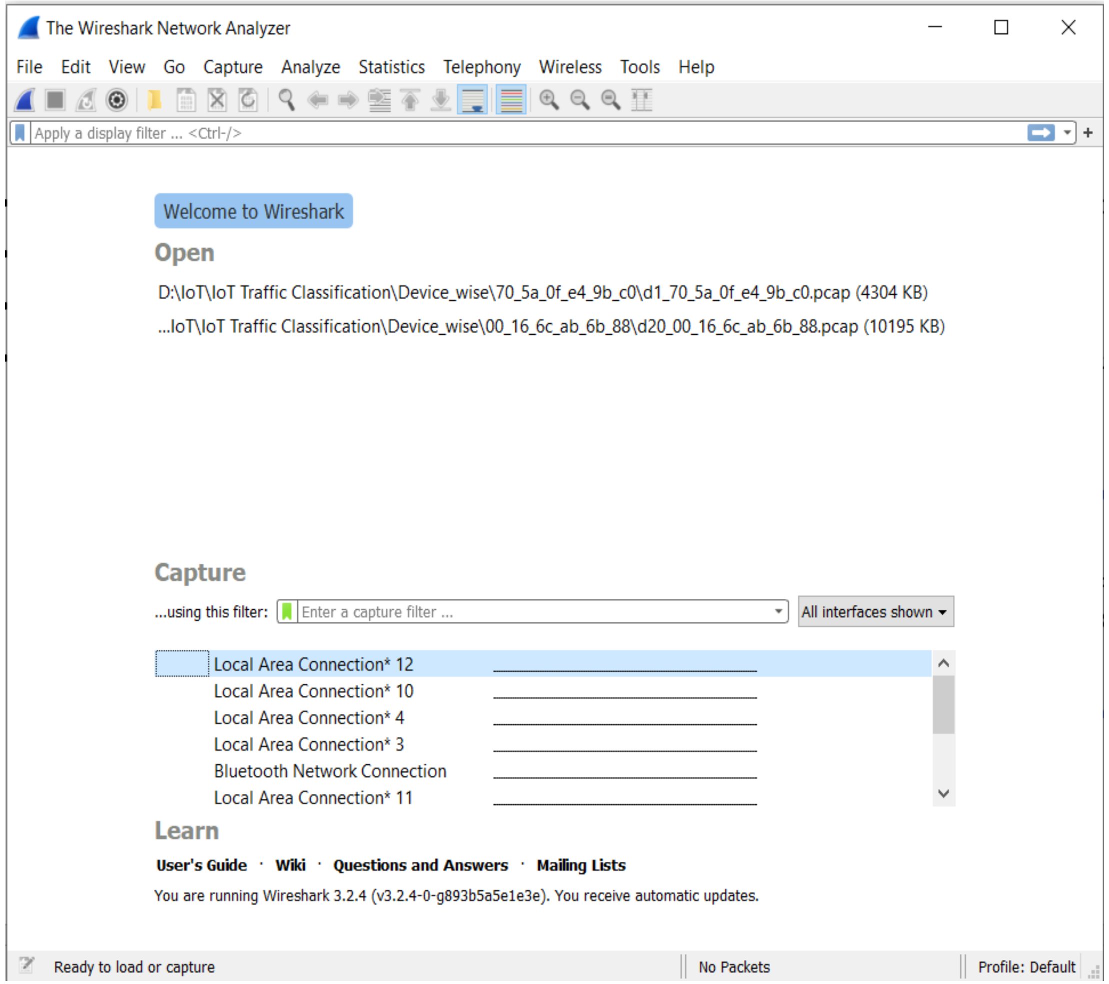


Fig. 7: IoT Traffic Capturing

IoT Network Traffic Capturing

- Network Traffic Capturing: Using Wireshark/TCPdump capturing the IoT network traffic.

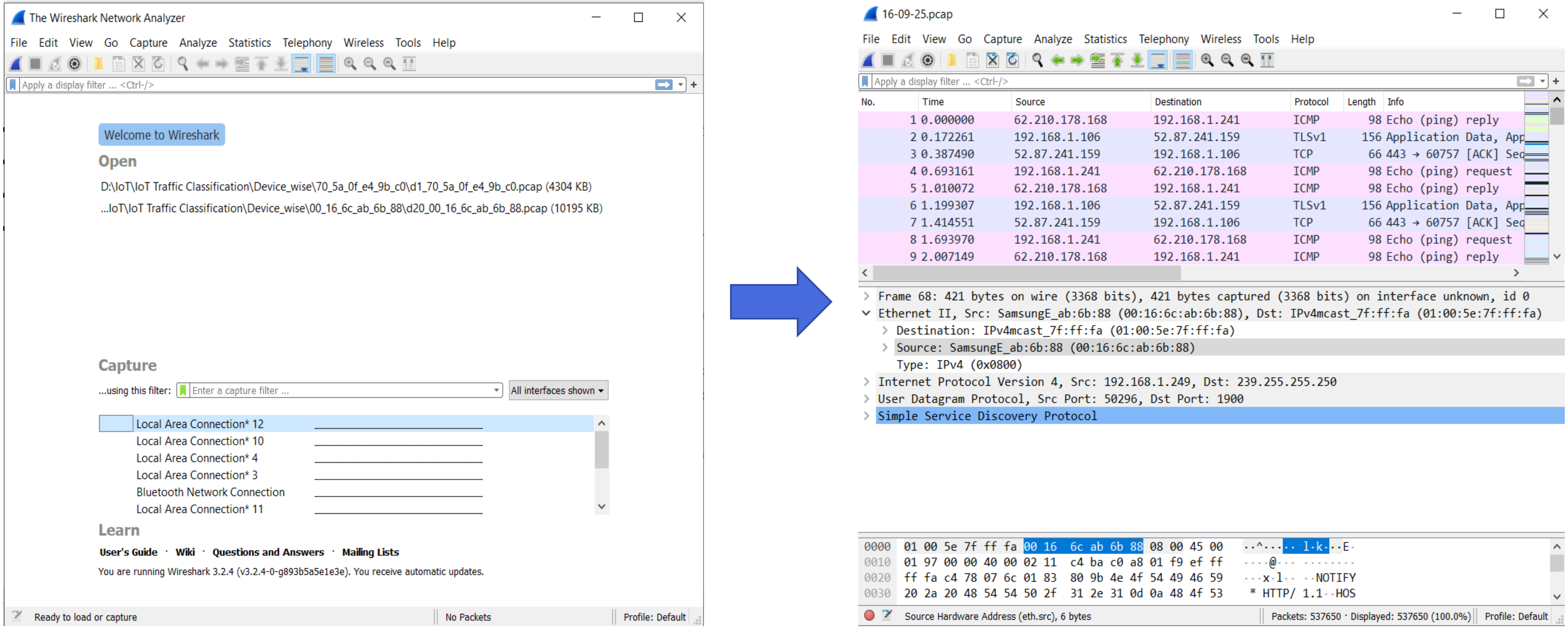


Fig. 7: IoT Traffic Capturing

IoT Network Traffic Splitting/Filtering

- Splitting/Filtering: Separate the IoT devices traffic from whole network traffic traces.
  - ➢ Method 1: Using Wireshark

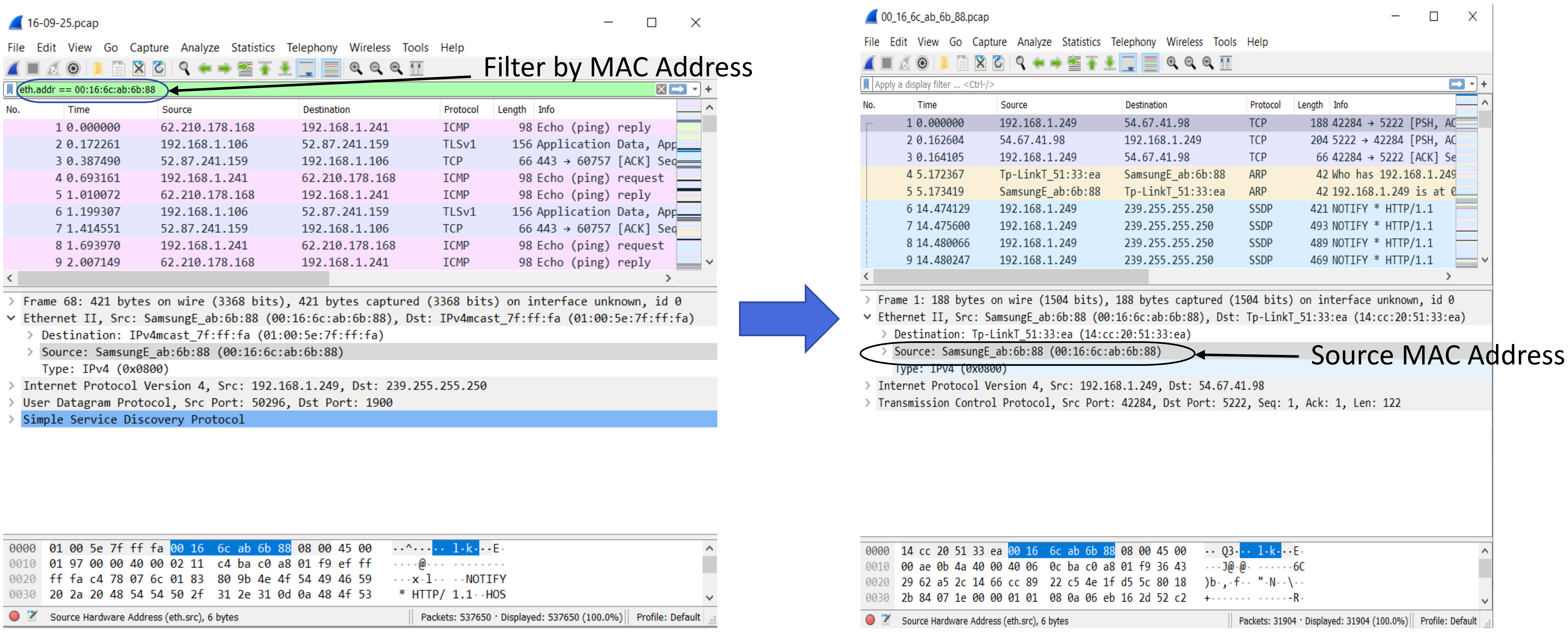


Fig. 8: IoT Traffic Splitting/Filtering

IoT Network Traffic Splitting/Filtering

- Splitting/Filtering: Separate the IoT devices traffic from whole network traffic traces.
  - Method 2: Bash Script

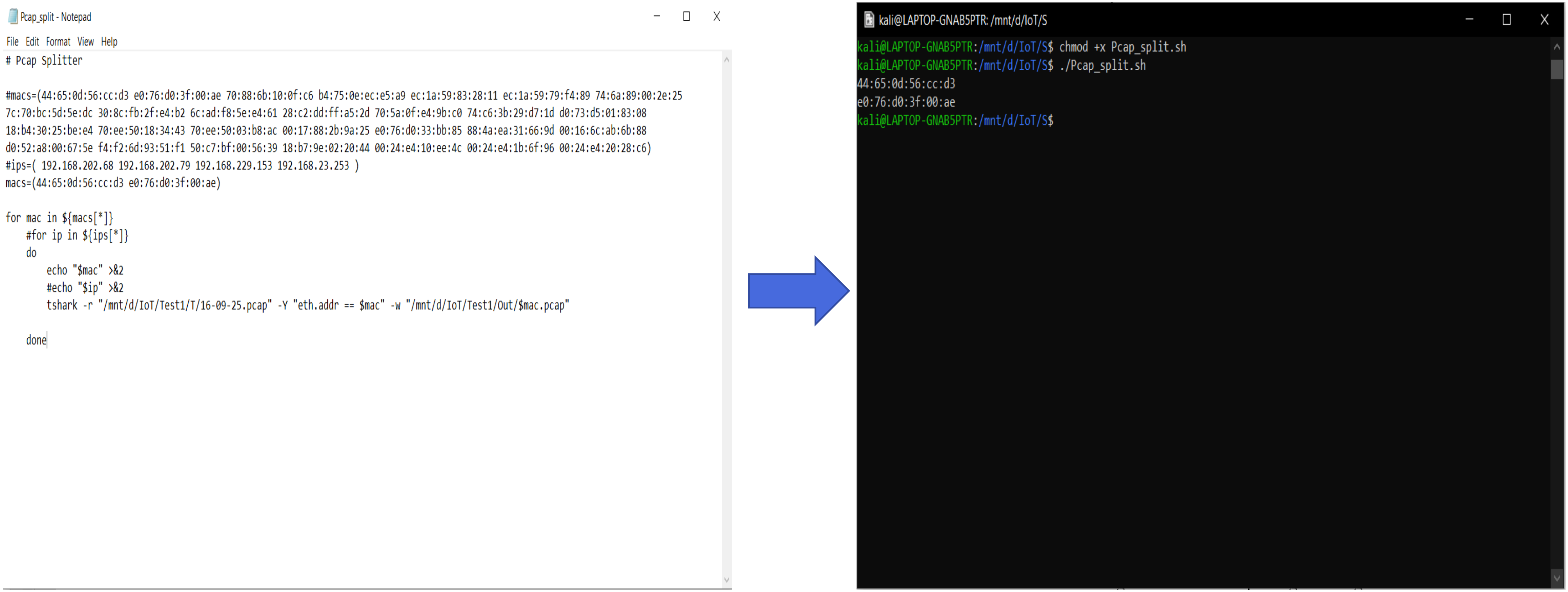


Fig. 8: IoT Traffic Splitting/Filtering

IoT Traffic Flow Construction

- Flow Construction: Construction of two types flows such as TCP & UDP from IoT traffic using scapy & python.

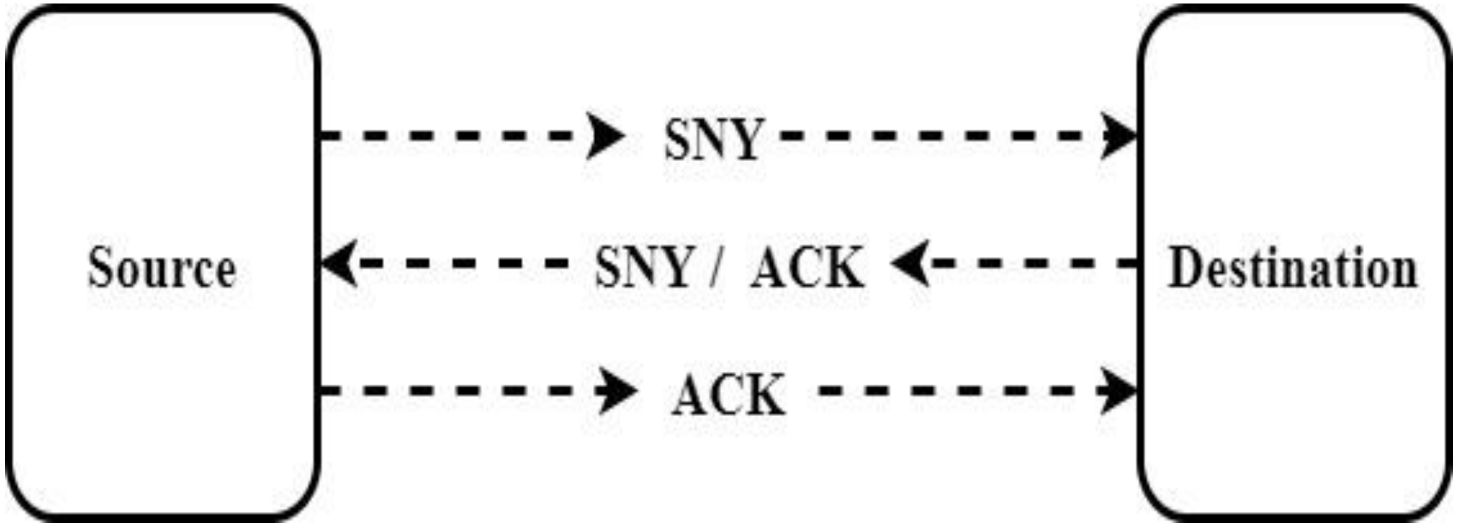


Fig. 9: IoT Traffic Flow Construction

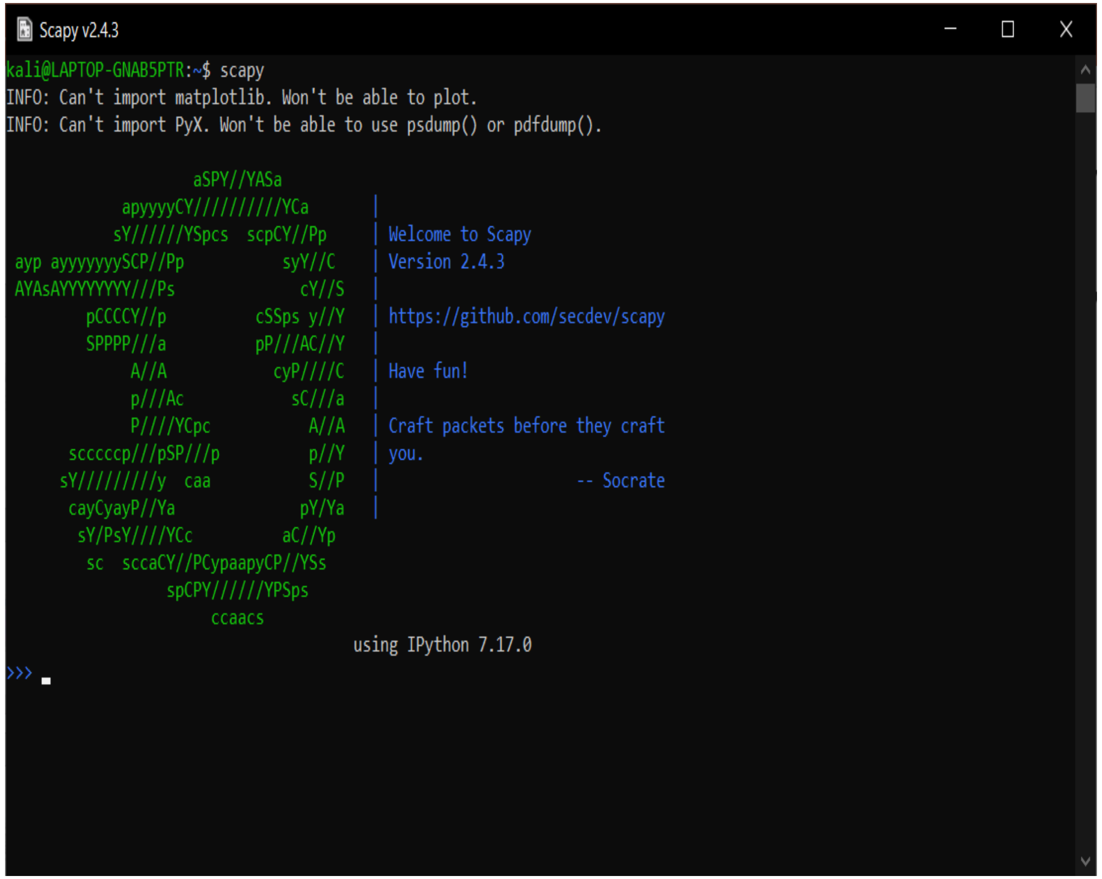
IoT Traffic Flow Construction

- Flow Construction: Construction of two types flows such as TCP & UDP from IoT traffic using scapy & python.

  - ➢ Download Scapy :

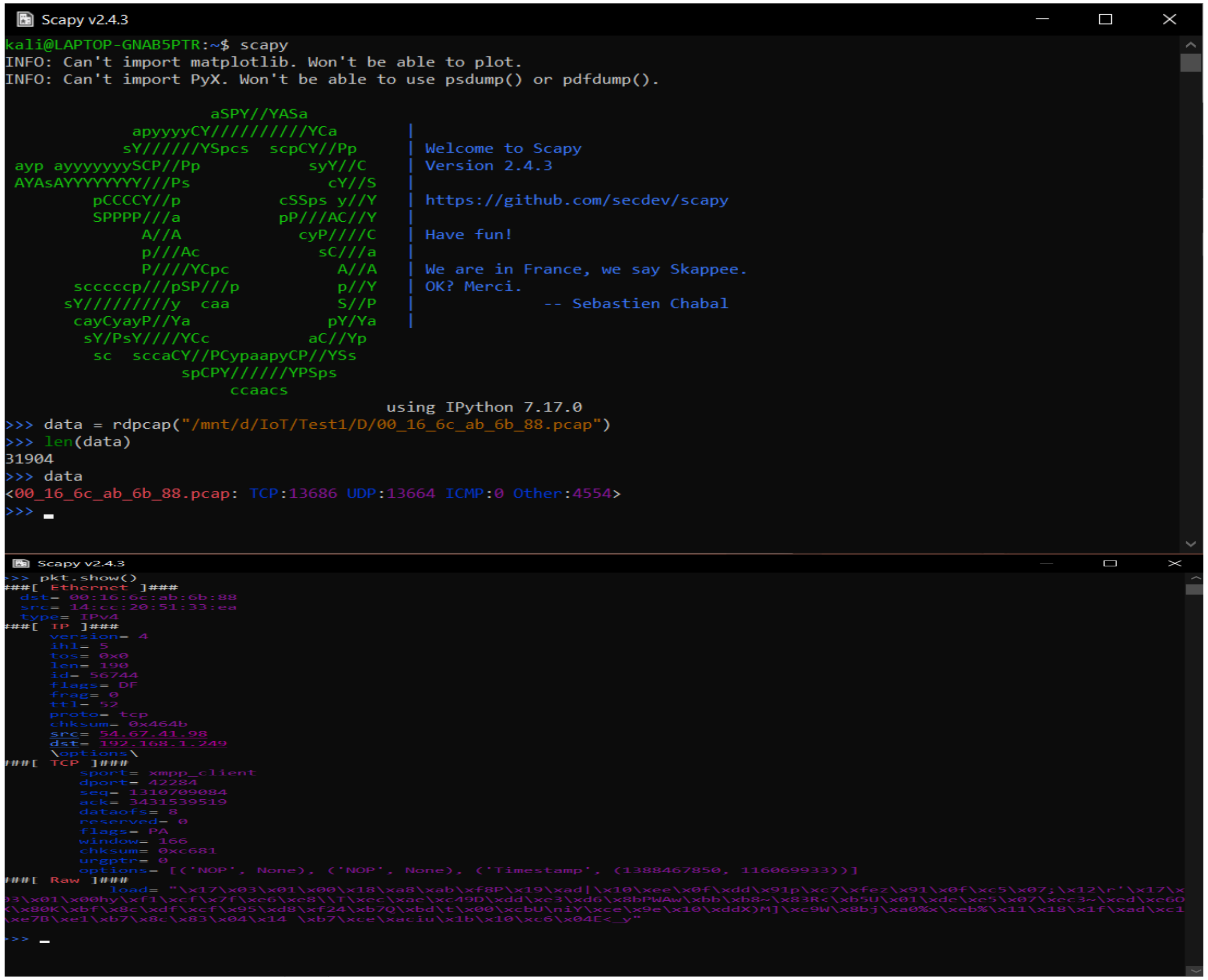
    https://scapy.readthedocs.io/en/latest/installation.html

  - ➢ Scapy
    - ✓ Packet Manipulation Python Tool
    - ✓ Flow Construction
    - ✓ Forge or Decode Packets
    - ✓ Scanning
    - ✓ Tracerouting
    - ✓ Attacks
    - ✓ Network Discovery



IoT Traffic Flow Construction

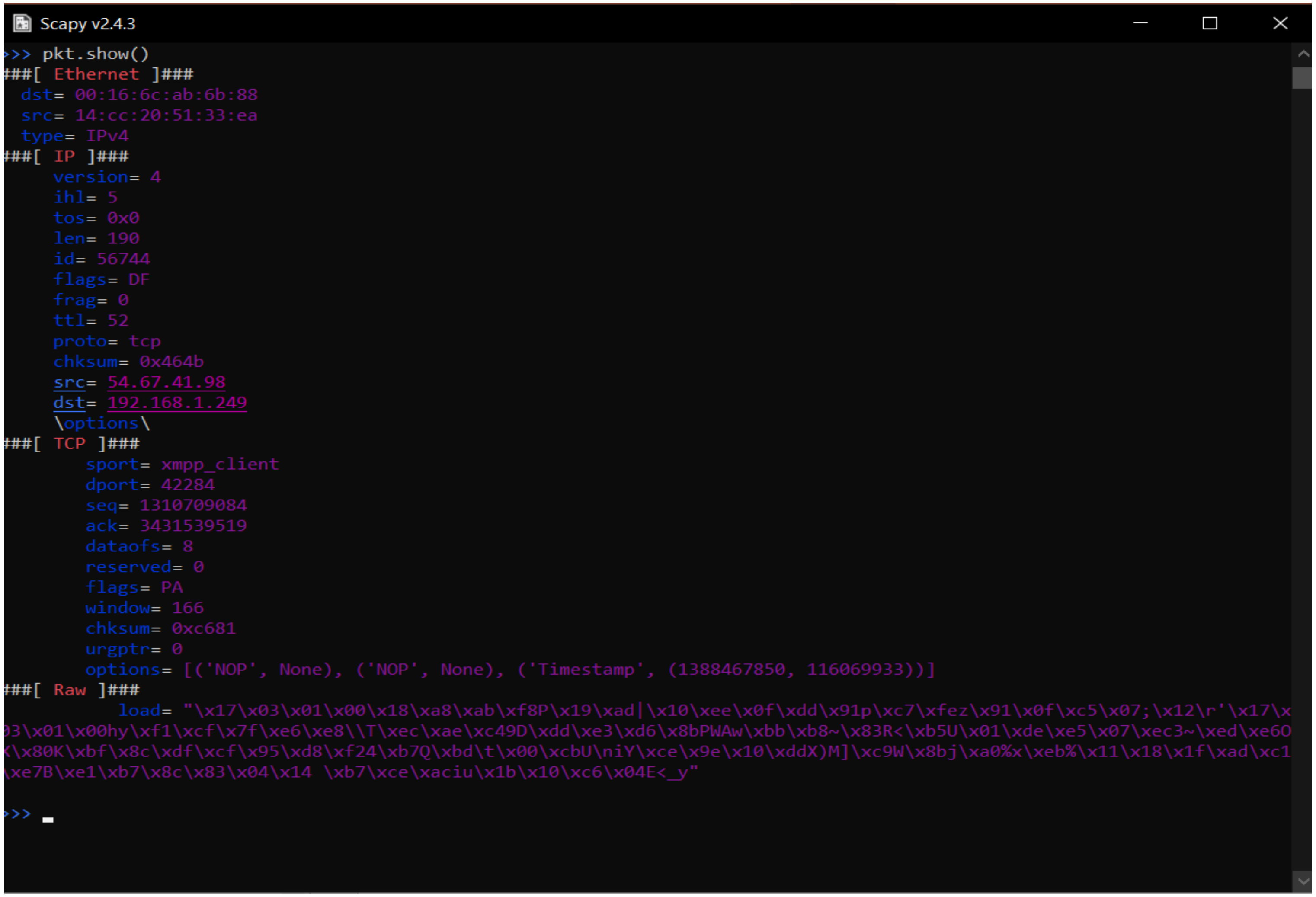
- Flow Construction: Construction of two types flows such as TCP & UDP from IoT traffic using scapy & python.
  - ➢ Working with Scapy :
    - ▪ Reading PCAP File
      - ✓ Open scapy on Terminal
      - ✓ Using rdpcap() for reading PCAP file
      - ✓ Check Number of packets in PCAP file



IoT Traffic Flow Construction

- Flow Construction: Construction of two types flows such as TCP & UDP from IoT traffic using scapy & python.

  - Working with Scapy :
    - Analyze Single Packet



IoT Traffic Flow Construction

- Flow Construction: Construction of two types flows such as TCP & UDP from IoT traffic using scapy & python.

➢ Working with Scapy :

  - Construction of Flow

Scapy v2.4.3

```
>>> sessions = data.sessions()
>>> for k,v in sessions.items():
...:     print(k,v)
...:
TCP 192.168.1.249:42284 > 54.67.41.98:5222 <PacketList: TCP:3491 UDP:0 ICMP:0 Other:0>
TCP 54.67.41.98:5222 > 192.168.1.249:42284 <PacketList: TCP:2253 UDP:0 ICMP:0 Other:0>
ARP 192.168.1.1 > 192.168.1.249 <PacketList: TCP:0 UDP:0 ICMP:0 Other:1173>
ARP 192.168.1.249 > 192.168.1.1 <PacketList: TCP:0 UDP:0 ICMP:0 Other:1173>
UDP 192.168.1.249:44976 > 239.255.255.250:1900 <PacketList: TCP:0 UDP:12159 ICMP:0 Other:0>
IPv6 fe80::216:6cff:feab:6b88 > ff02::1:ffab:6b88 nh=Hop-by-Hop Option Header <PacketList: TCP:0 UDP:0 ICMP:0 Other:414>
IPv6 fe80::216:6cff:feab:6b88 > ff02::fb nh=Hop-by-Hop Option Header <PacketList: TCP:0 UDP:0 ICMP:0 Other:414>
IP 192.168.1.249 > 239.255.255.250 proto=igmp <PacketList: TCP:0 UDP:0 ICMP:0 Other:413>
UDP 192.168.1.249:1900 > 192.168.1.193:3080 <PacketList: TCP:0 UDP:648 ICMP:0 Other:0>
TCP 192.168.1.193:3916 > 192.168.1.249:49152 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.193:3918 > 192.168.1.249:49152 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.249:49152 > 192.168.1.193:3916 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.249:49152 > 192.168.1.193:3918 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
IP 192.168.1.249 > 224.0.0.251 proto=igmp <PacketList: TCP:0 UDP:0 ICMP:0 Other:413>
TCP 192.168.1.193:4442 > 192.168.1.249:49152 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.249:49152 > 192.168.1.193:4442 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.193:4444 > 192.168.1.249:49152 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.249:49152 > 192.168.1.193:4444 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.193:4451 > 192.168.1.249:49152 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.249:49152 > 192.168.1.193:4451 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.193:4455 > 192.168.1.249:49152 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.249:49152 > 192.168.1.193:4455 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.193:4637 > 192.168.1.249:49152 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.193:4640 > 192.168.1.249:49152 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.249:49152 > 192.168.1.193:4637 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.249:49152 > 192.168.1.193:4640 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
Ethernet type=888e <PacketList: TCP:0 UDP:0 ICMP:0 Other:86>
TCP 192.168.1.193:4752 > 192.168.1.249:49152 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.249:49152 > 192.168.1.193:4752 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.193:4754 > 192.168.1.249:49152 <PacketList: TCP:6 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.249:49152 > 192.168.1.193:4754 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
TCP 192.168.1.193:4212 > 192.168.1.249:49152 <PacketList: TCP:5 UDP:0 ICMP:0 Other:0>
```

Feature Extraction from IoT Traffic

- Feature Extraction: Extract the three types of features such as packet level, flow level & behavior level.

Packet Level Attributes	**Flow Level Attributes**	**Behavior Level Attributes**
Packet Length	Flow Length	DNS Interval
Packet Source Port No.	Flow Duration	NTP Interval
Packet Destination Port No.	Flow Ratio	Cipher Suites
Packet Payload Length	Flow Payload Length	Domain Names

Training & Testing of IoT Traffic

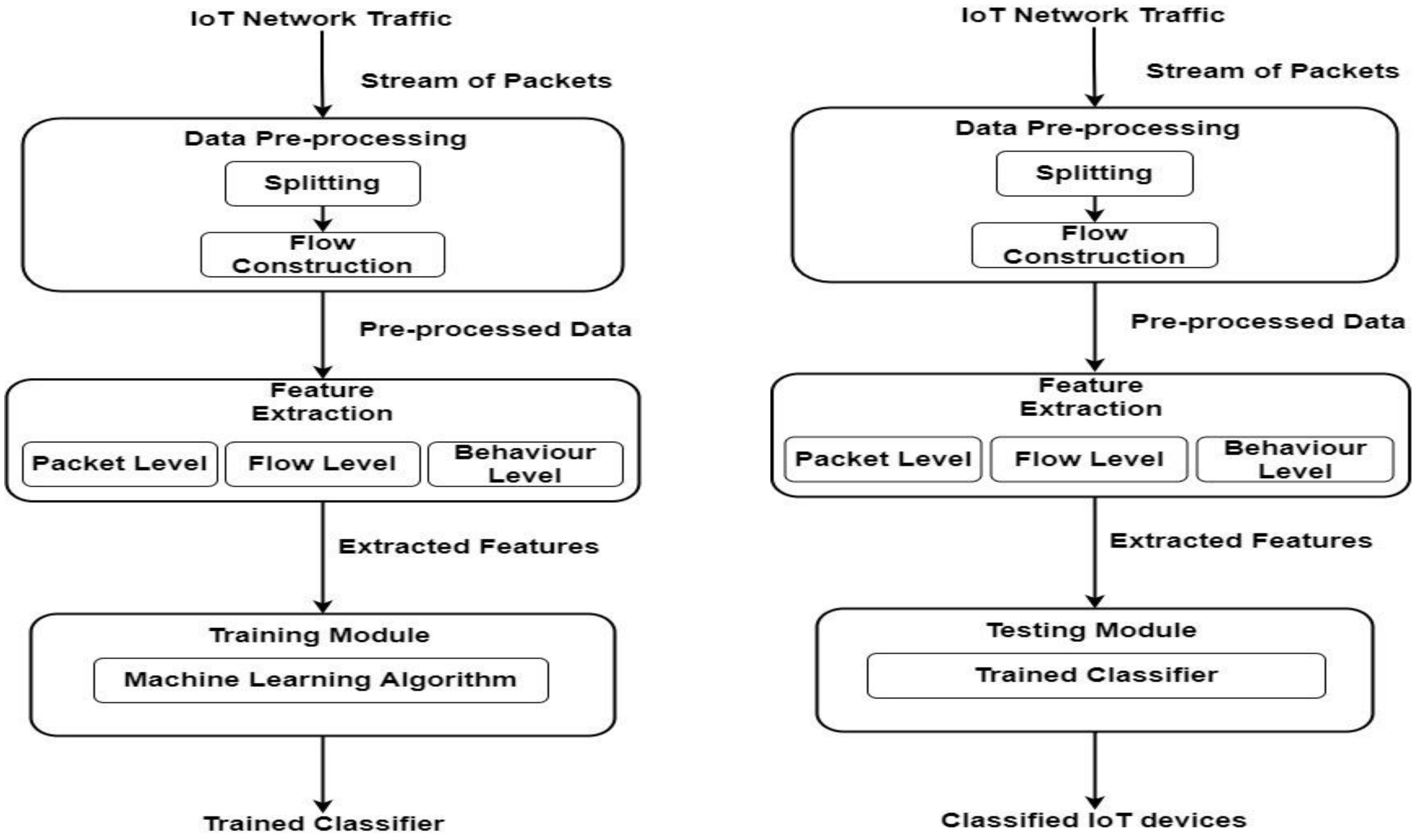
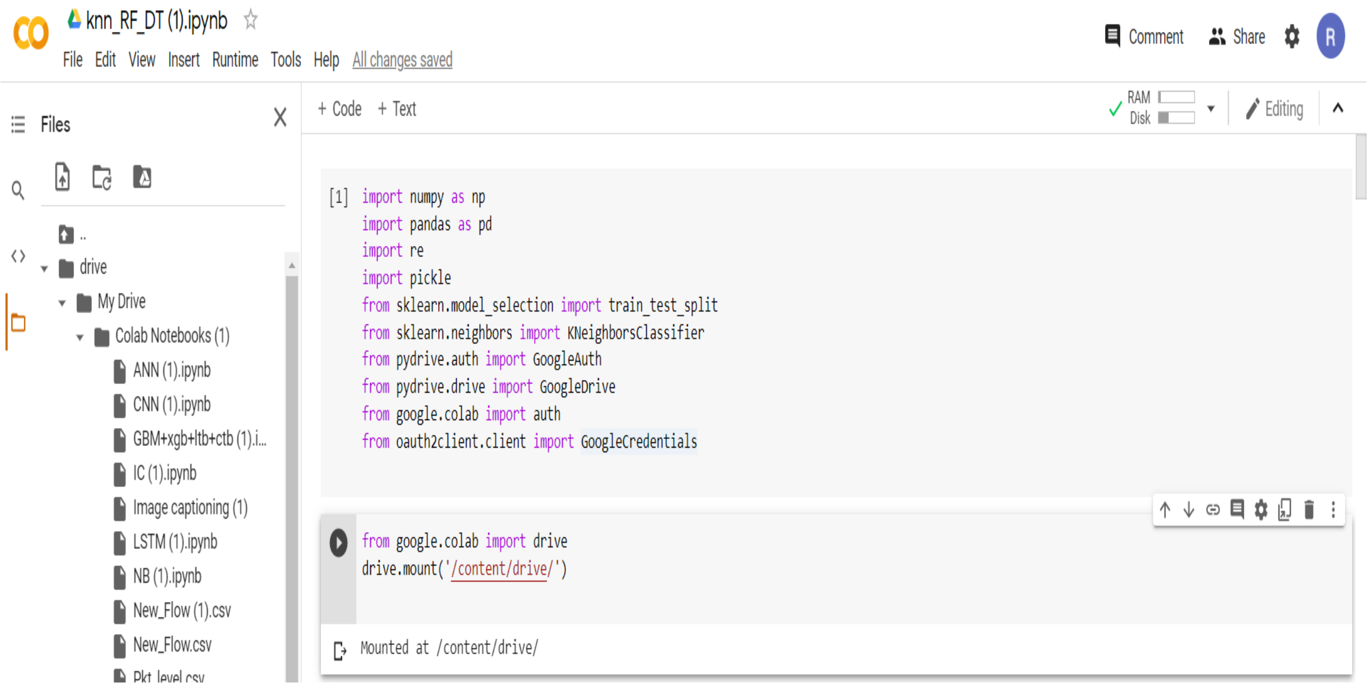


Fig. 10: Training Module & Testing Module

Training & Testing of IoT Traffic

- Training a machine learning classifier simply learning a certain type of patterns from a labeled input IoT traffic.

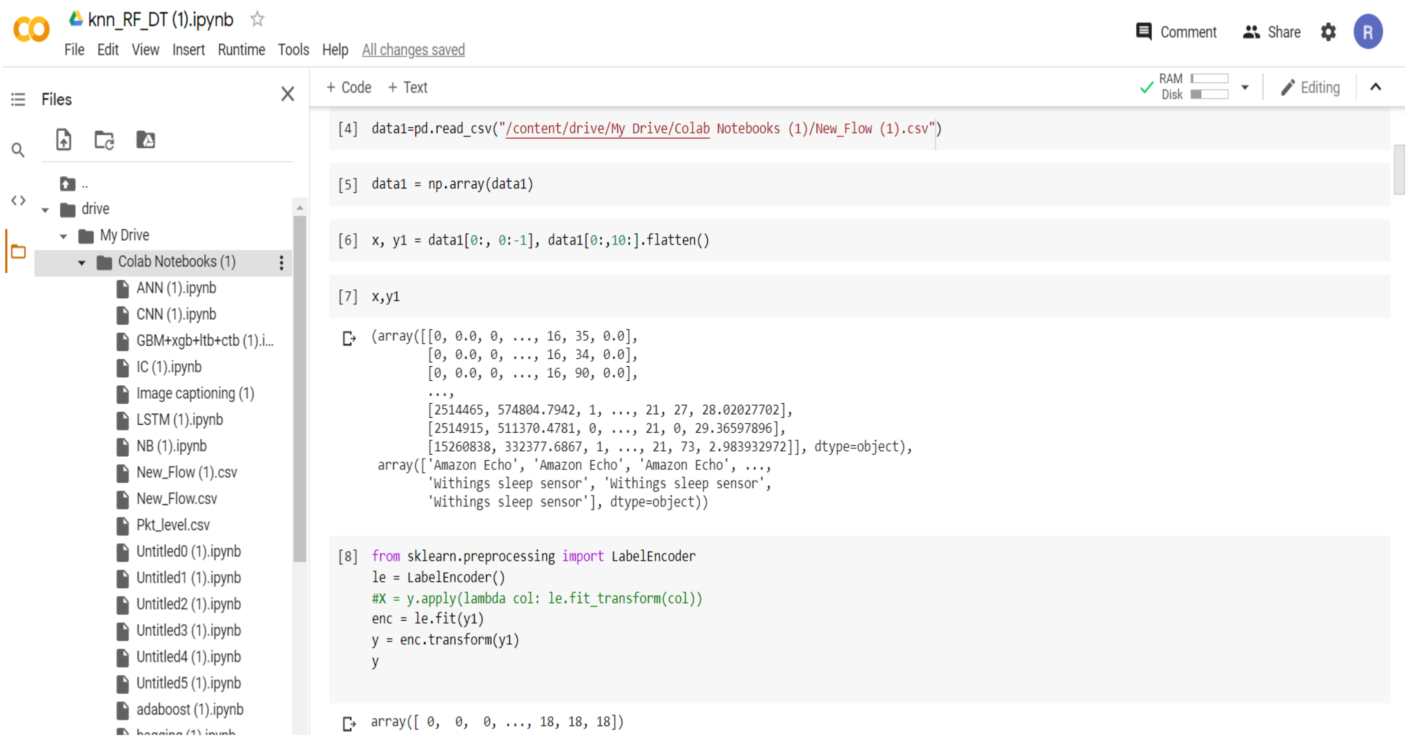
  - Steps to Train a Classifier:
    - Using Google Colab
      - ✓ Free online cloud-based Jupyter notebook environment
      - ✓ To train machine learning and deep learning models on CPUs, GPUs, and TPUs.
    - Importing Packages
    - Import Google Drive



Training & Testing of IoT Traffic

- Training a machine learning classifier simply learning a certain type of patterns from a labeled input IoT traffic.

  - ➢ Steps to Train a Classifier:
    - Using Google Colab
      - ✓ Free online cloud-based Jupyter notebook environment
      - ✓ To train machine learning and deep learning models on CPUs, GPUs, and TPUs.
    - Importing Packages
    - Import Google Drive
    - Data Type Selection and Upload
    - Data Mapping
    - Label Encoding (If needed)

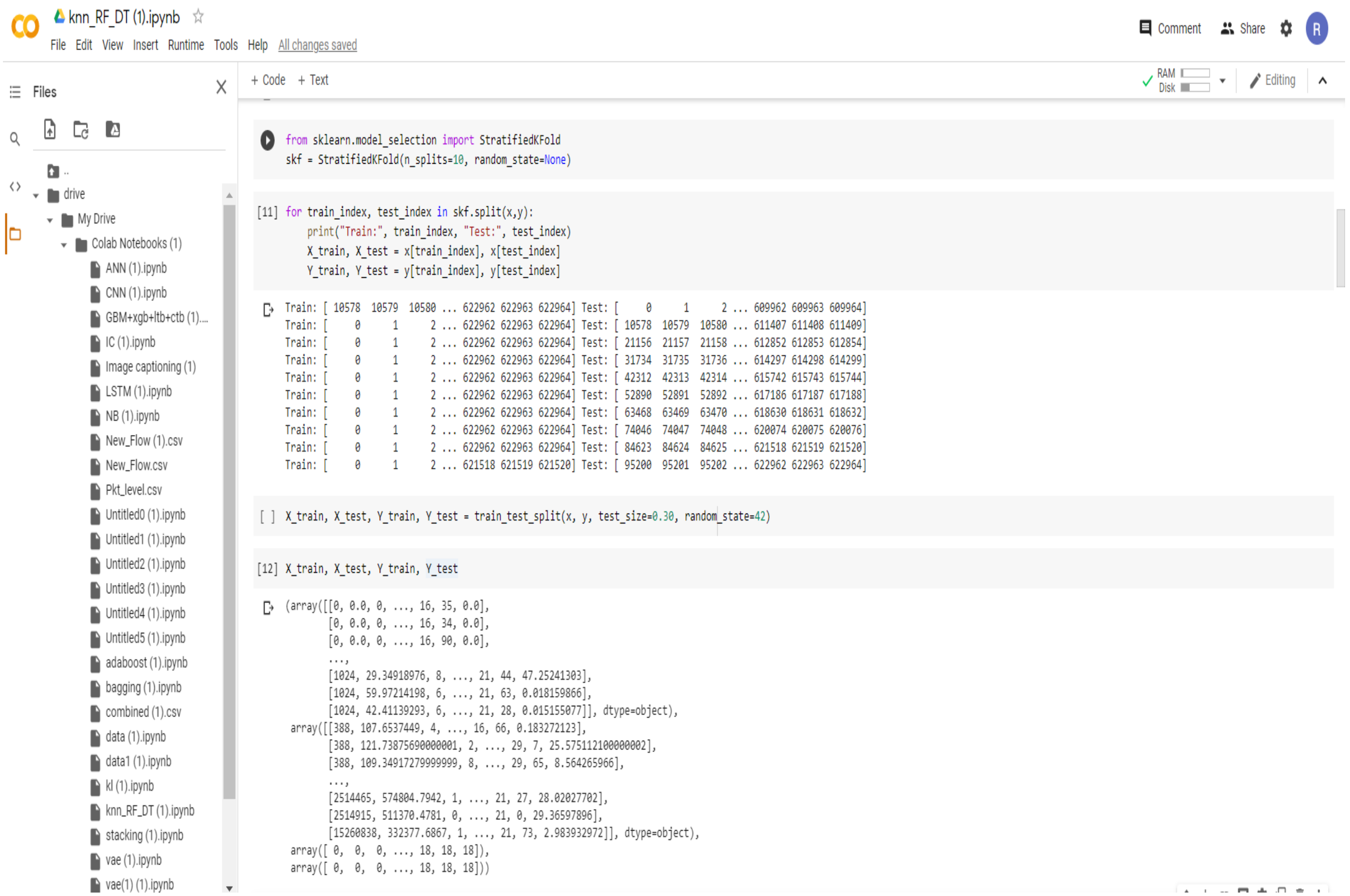


Training & Testing of IoT Traffic

- Training a machine learning classifier simply learning a certain type of patterns from a labeled input IoT traffic.

➢ Steps to Train a Classifier:

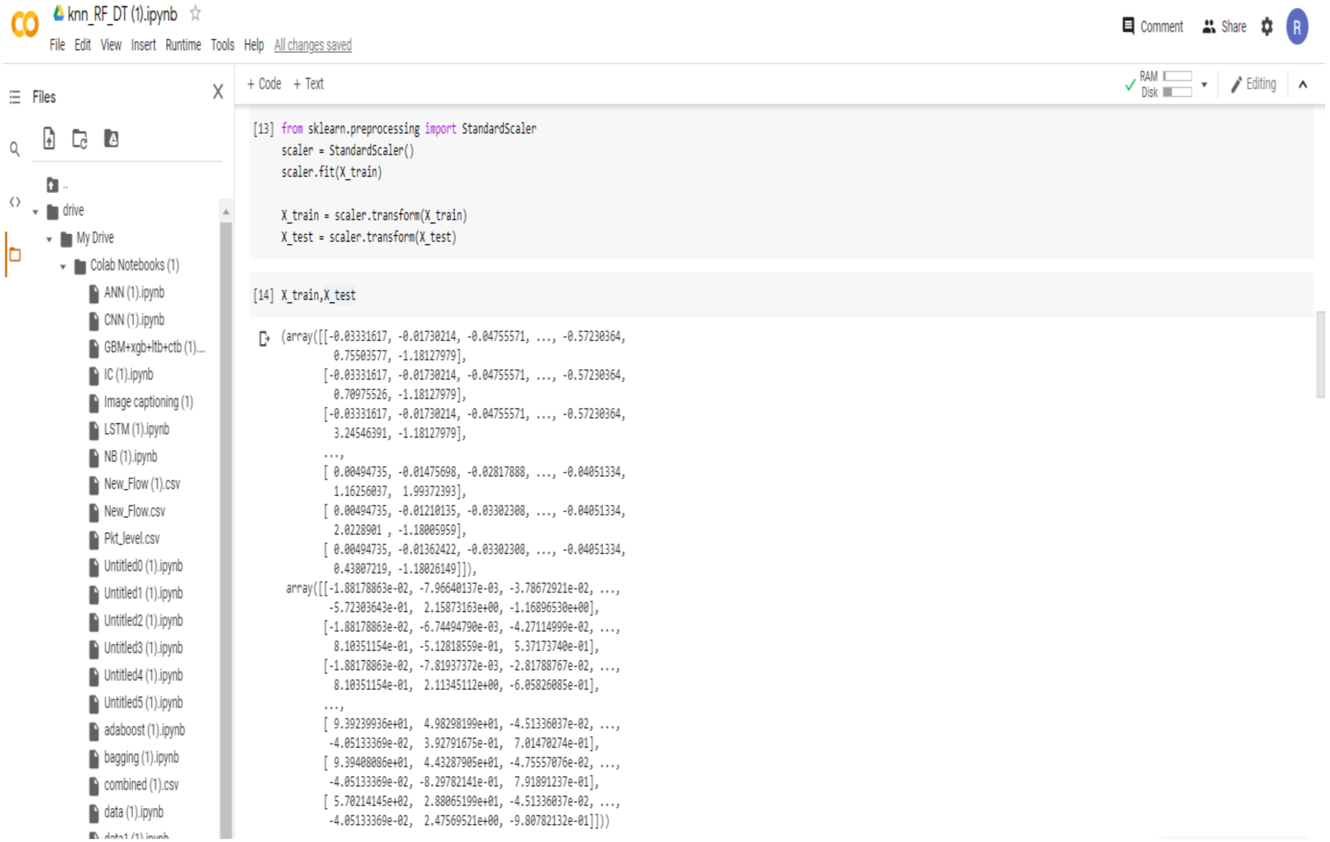
- Using Google Colab
  - ✓ Free online cloud-based Jupyter notebook environment
  - ✓ To train machine learning and deep learning models on CPUs, GPUs, and TPUs.
- Importing Packages
- Import Google Drive
- Data Type Selection and Upload
- Data Mapping
- Label Encoding (If needed)
- K-fold Cross validation or Splitting of Data (Choice)



Training & Testing of IoT Traffic

- Training a machine learning classifier simply learning a certain type of patterns from a labeled input IoT traffic.

  - Steps to Train a Classifier:
    - Using Google Colab
      - ✓ Free online cloud-based Jupyter notebook environment
      - ✓ To train machine learning and deep learning models on CPUs, GPUs, and TPUs.
    - Importing Packages
    - Import Google Drive
    - Data Type Selection and Upload
    - Data Mapping
    - Label Encoding (If needed)
    - K-fold Cross validation or Splitting of Data (Choice)
    - Normalization of Data

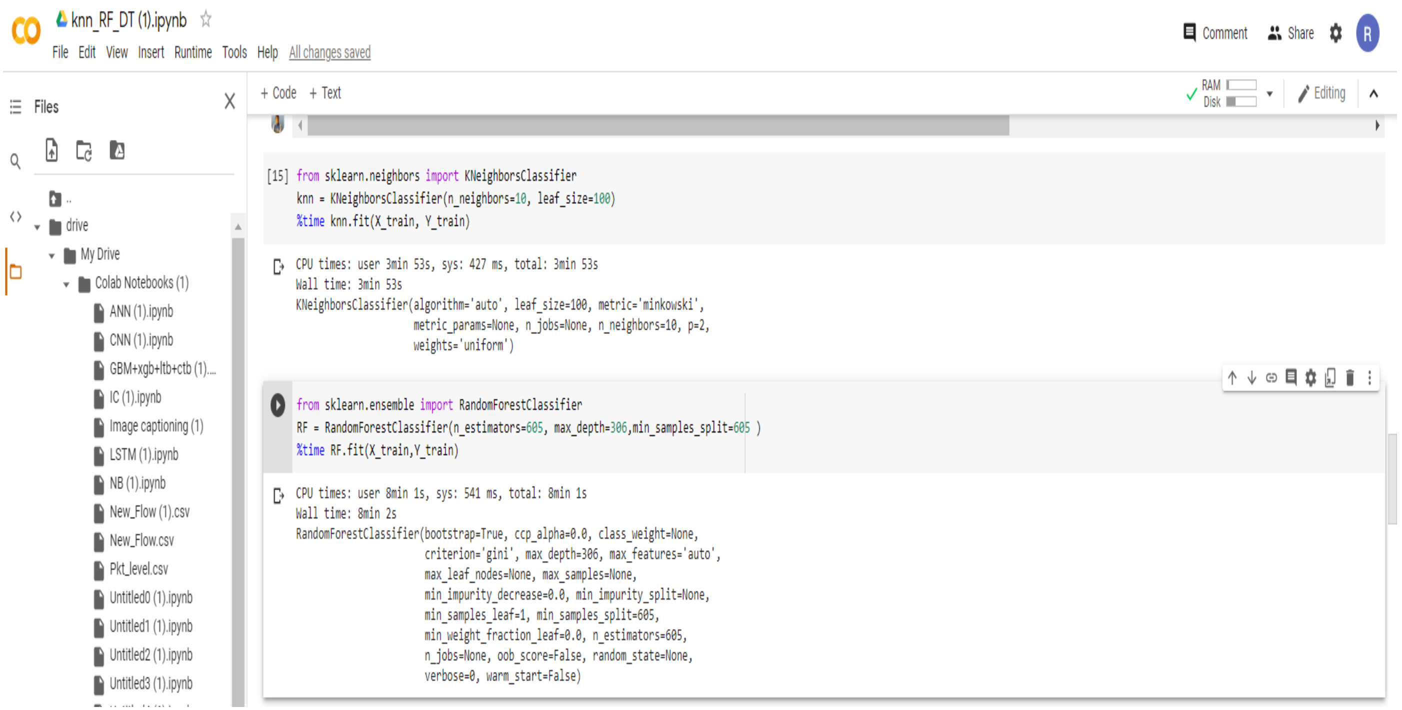


Training & Testing of IoT Traffic

- Training a machine learning classifier simply learning a certain type of patterns from a labeled input IoT traffic.

➢ Steps to Train a Classifier:

  - Using Google Colab
    - ✓ Free online cloud-based Jupyter notebook environment
    - ✓ To train machine learning and deep learning models on CPUs, GPUs, and TPUs.
  - Importing Packages
  - Import Google Drive
  - Data Type Selection and Upload
  - Data Mapping
  - Label Encoding (If needed)
  - K-fold Cross validation or Splitting of Data (Choice)
  - Normalization of Data
  - Training of Classifier

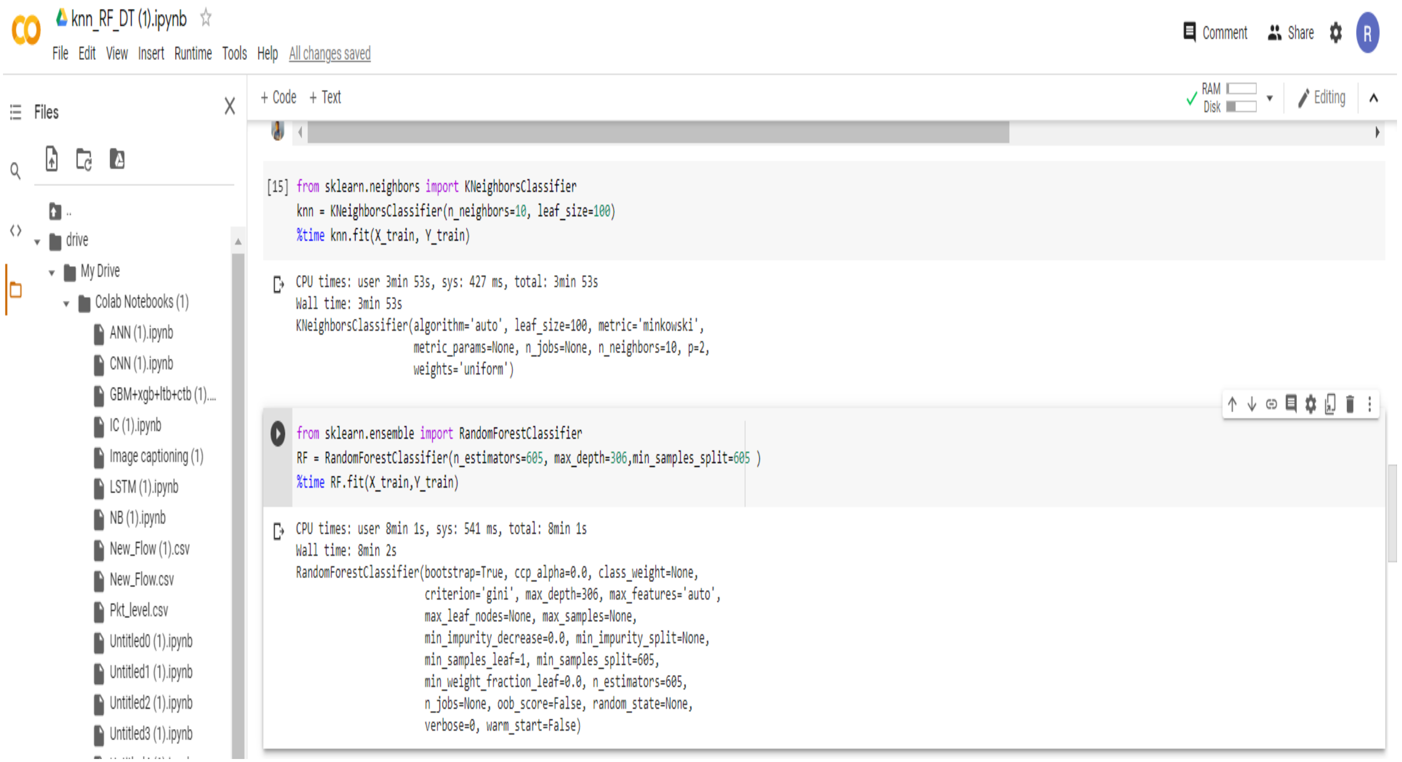


Training & Testing of IoT Traffic

- Training a machine learning classifier simply learning a certain type of patterns from a labeled input IoT traffic.

➢ Steps to Train a Classifier:

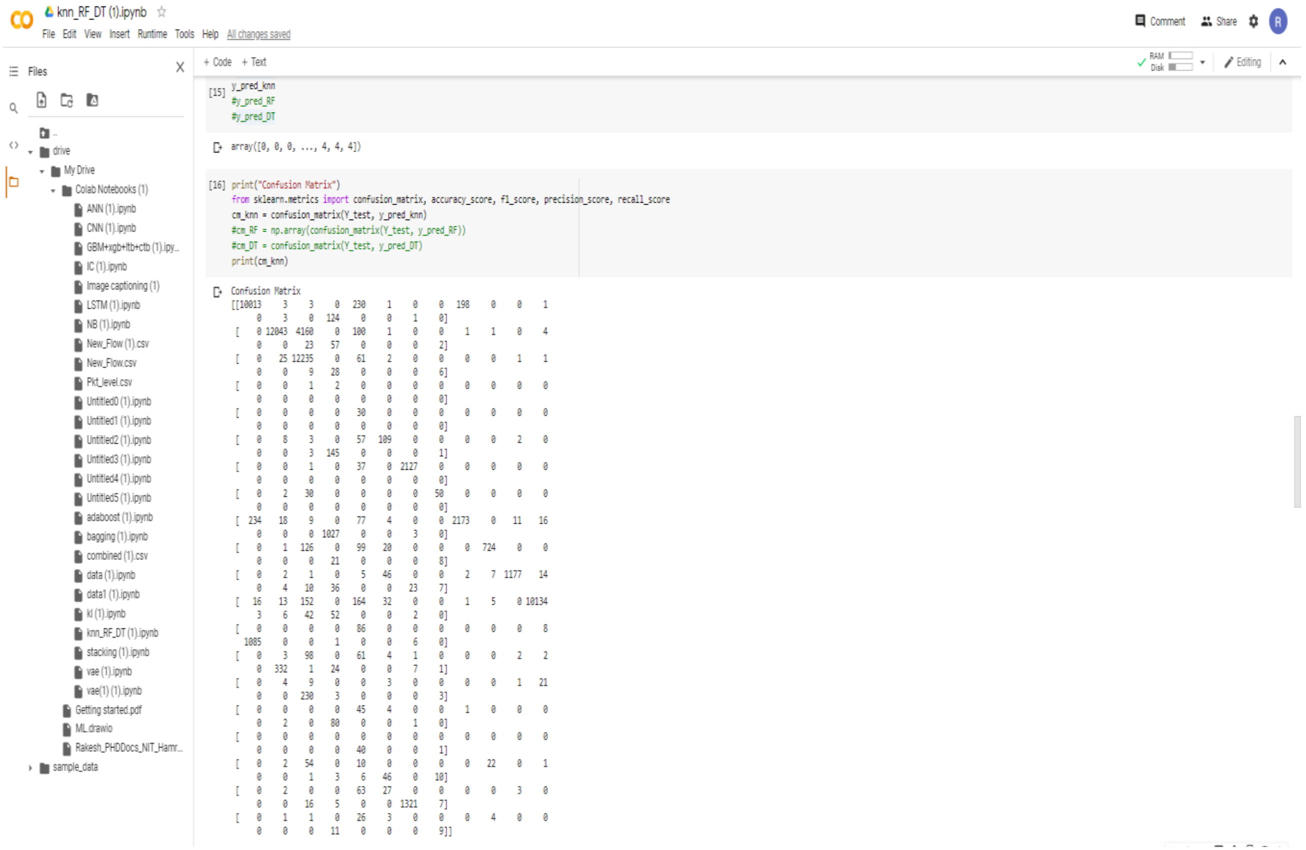
- Using Google Colab
  - ✓ Free online cloud-based Jupyter notebook environment
  - ✓ To train machine learning and deep learning models on CPUs, GPUs, and TPUs.
- Importing Packages
- Import Google Drive
- Data Type Selection and Upload
- Data Mapping
- Label Encoding (If needed)
- K-fold Cross validation or Splitting of Data (Choice)
- Normalization of Data
- Training of Classifier



Training & Testing of IoT Traffic

- Testing a machine learning classifier for algorithmic correctness and assuring the quality of newly build model.

  - ➢ Steps to Testing a Classifier:
    - Testing Classifier with test dataset
    - Making Confusion Matrix
      - ✓ Easy way to measure the performance of Classifier

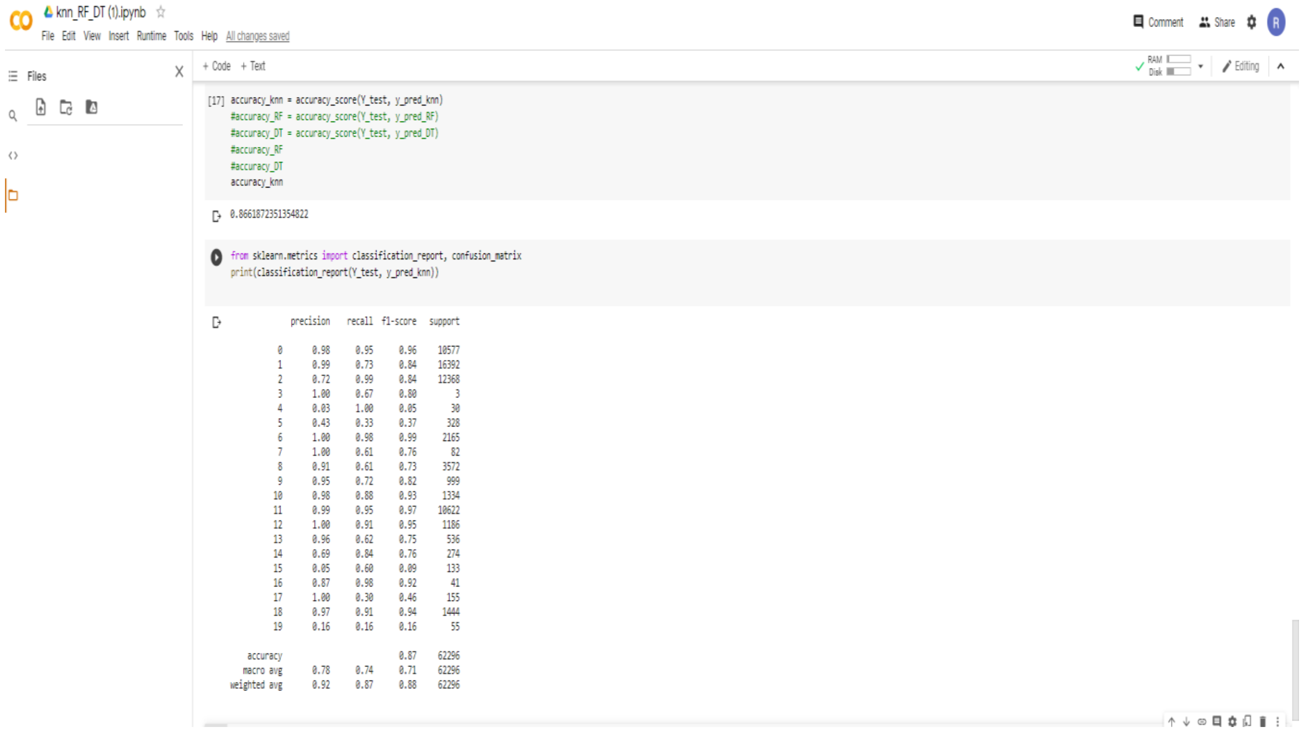


Training & Testing of IoT Traffic

- Testing a machine learning classifier for algorithmic correctness and assuring the quality of newly build model.

➢ Steps to Testing a Classifier:

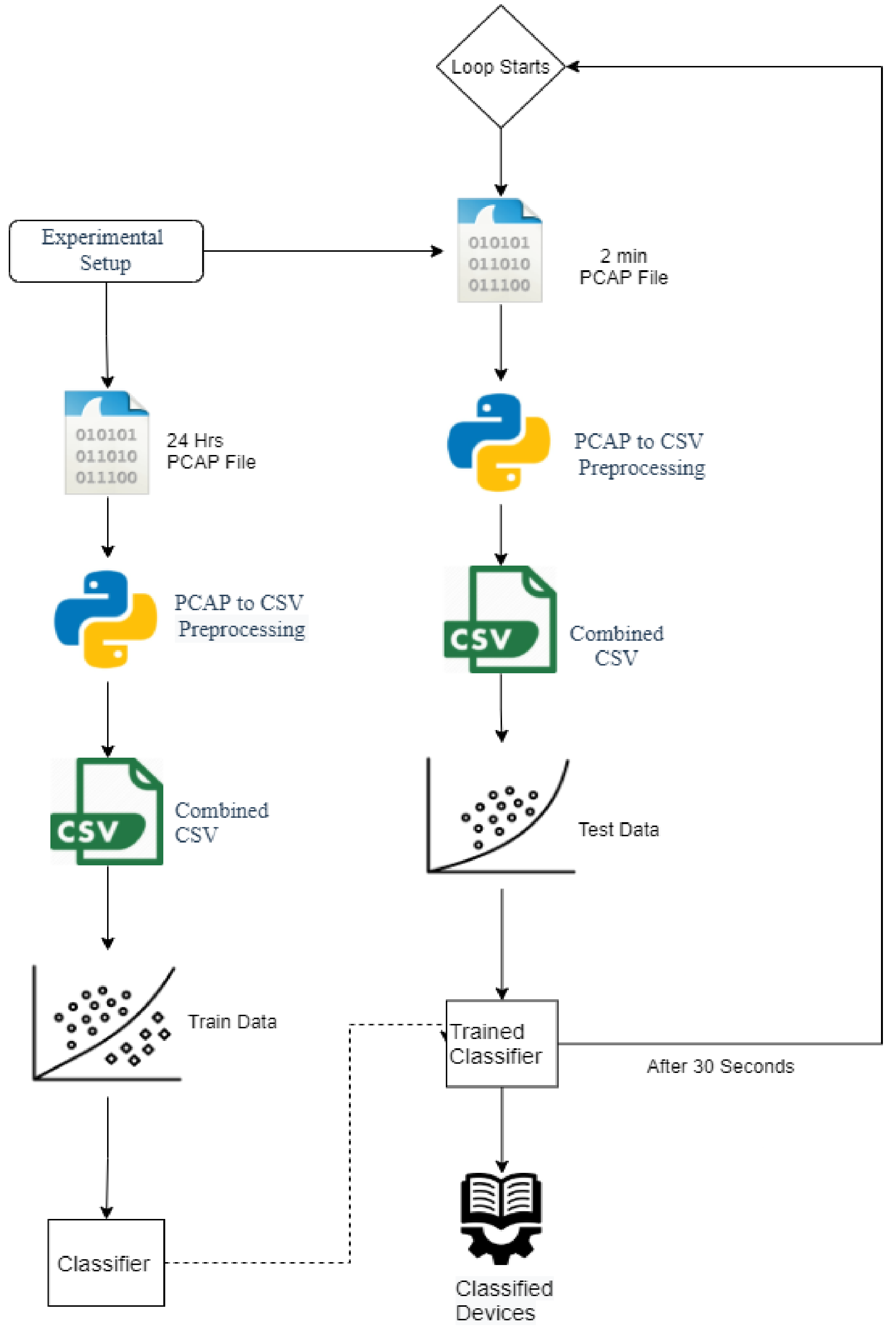
- Testing Classifier with test dataset
- Making Confusion Matrix
  - ✓ Easy way to measure the performance of Classifier
- Accuracy
- Evaluation Metrics
  - ✓ Precession
  - ✓ Recall
  - ✓ F1 Score



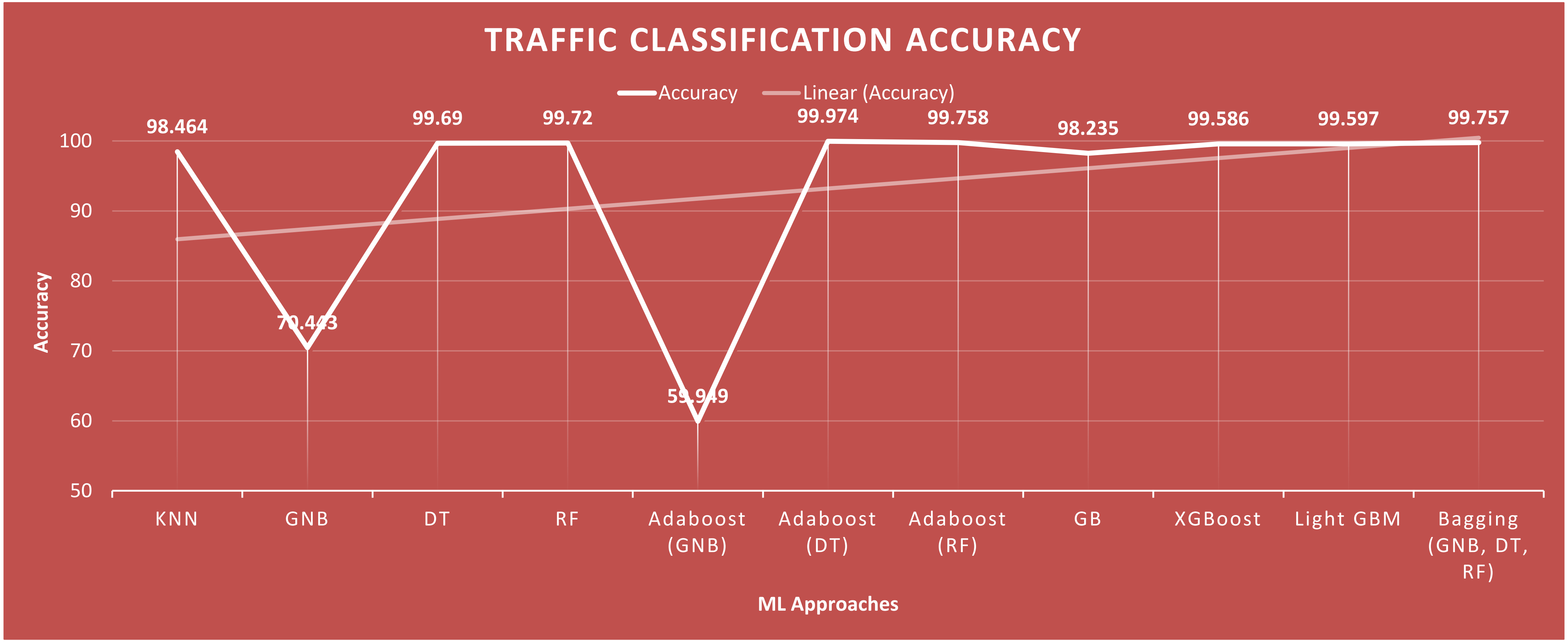
Live Testing Procedure

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

Live Testing Flowchart



Trained Model Accuracy for Device classification



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/

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.

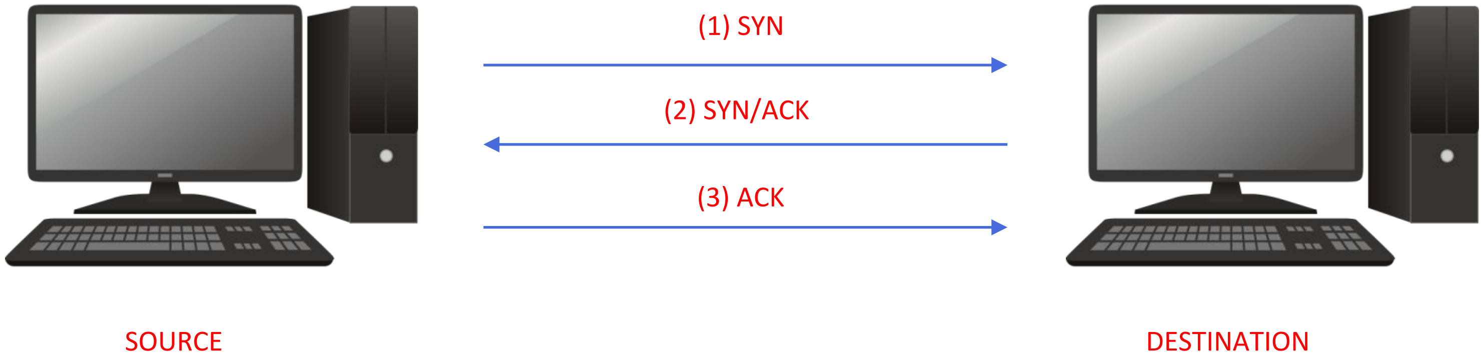
Steps of SYN Flood

- 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)

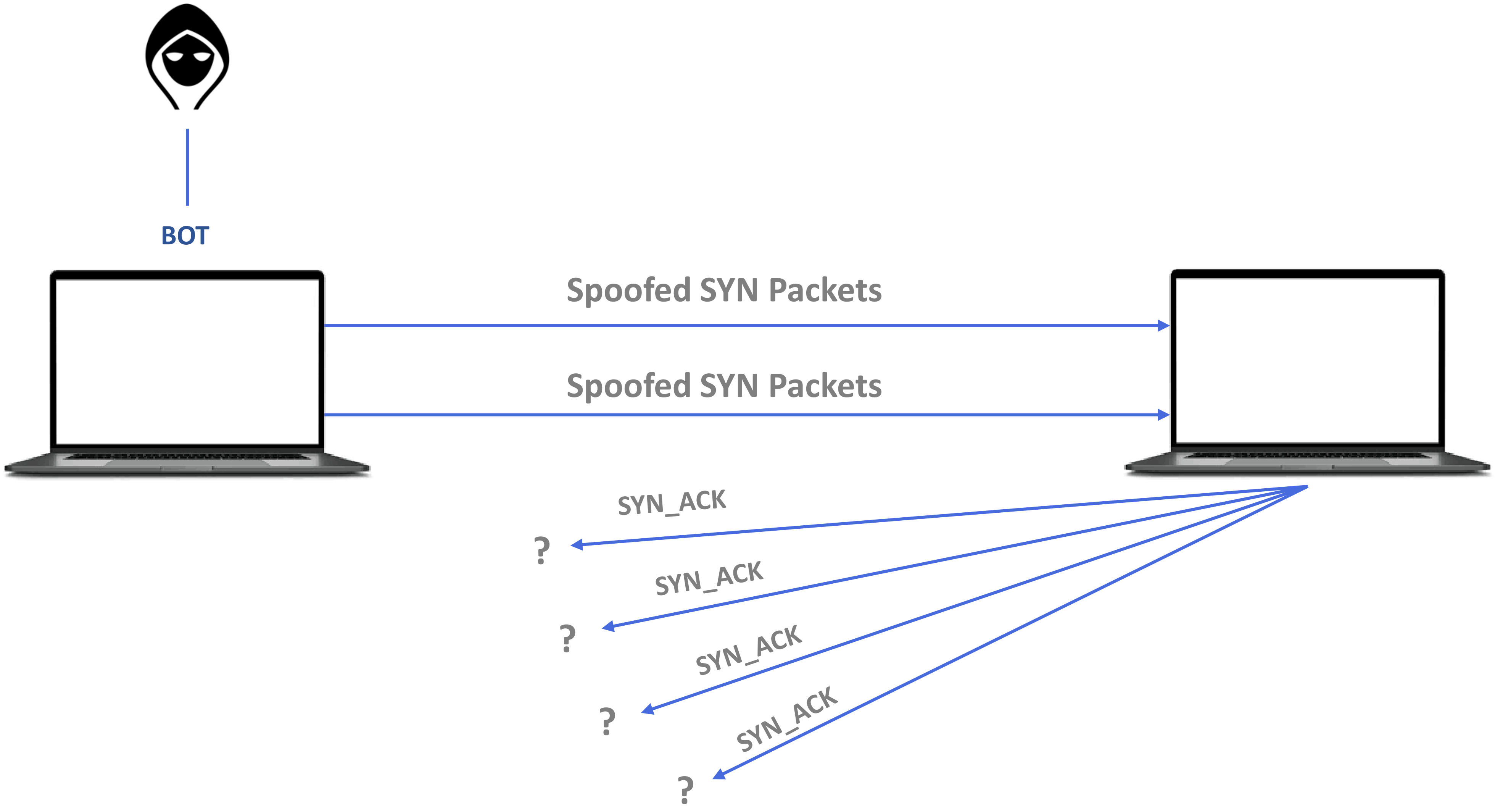
SYN = SYNCHRONIZATION

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
?

ARP Protocol

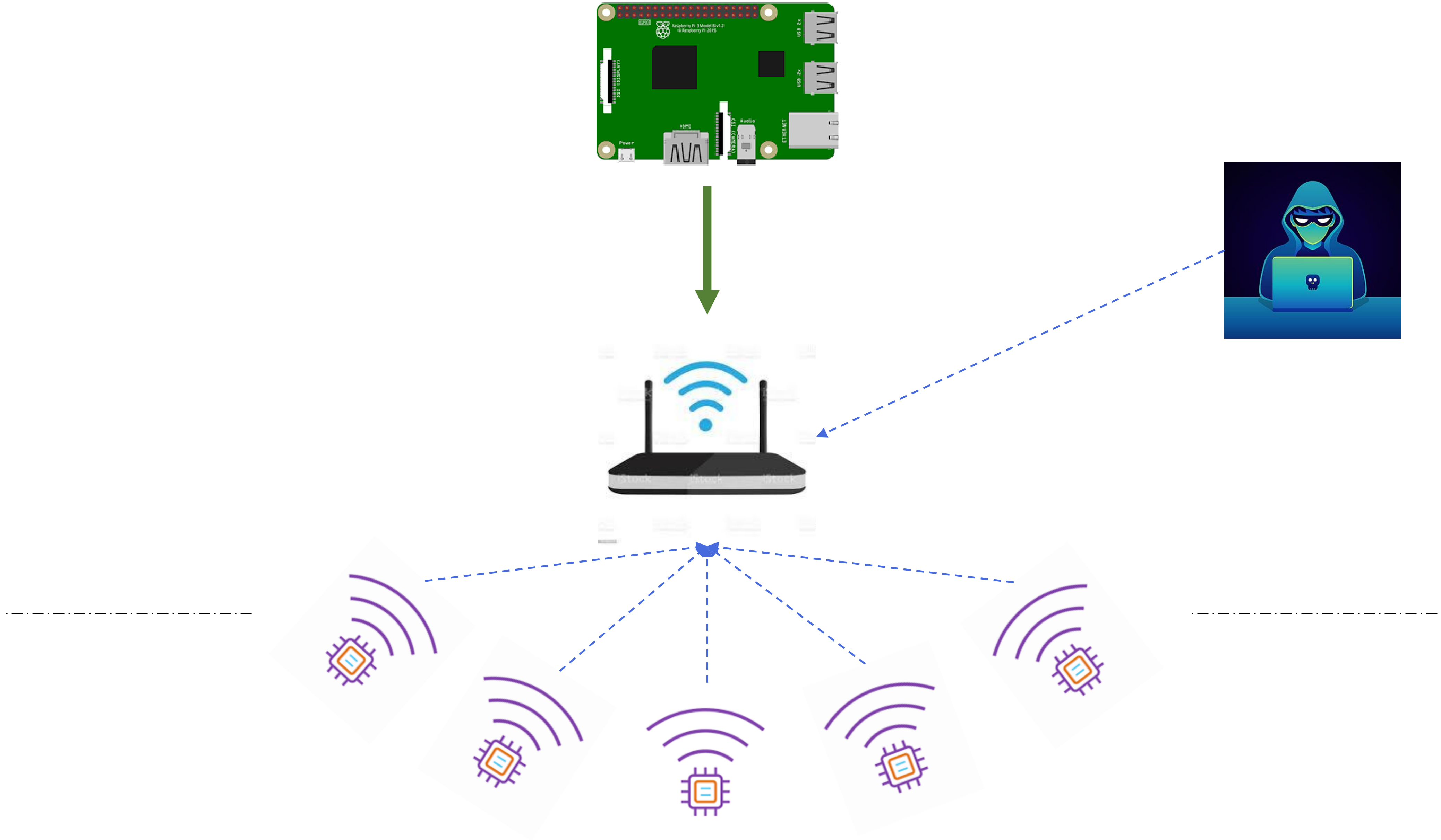
- Address Resolution Protocol (ARP) is a protocol that enables network communications to reach a specific device on the network.
- ARP translates Internet Protocol (IP) addresses to a Media Access Control (MAC) address, and vice versa.
- Most commonly, devices use ARP to contact the router or gateway 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

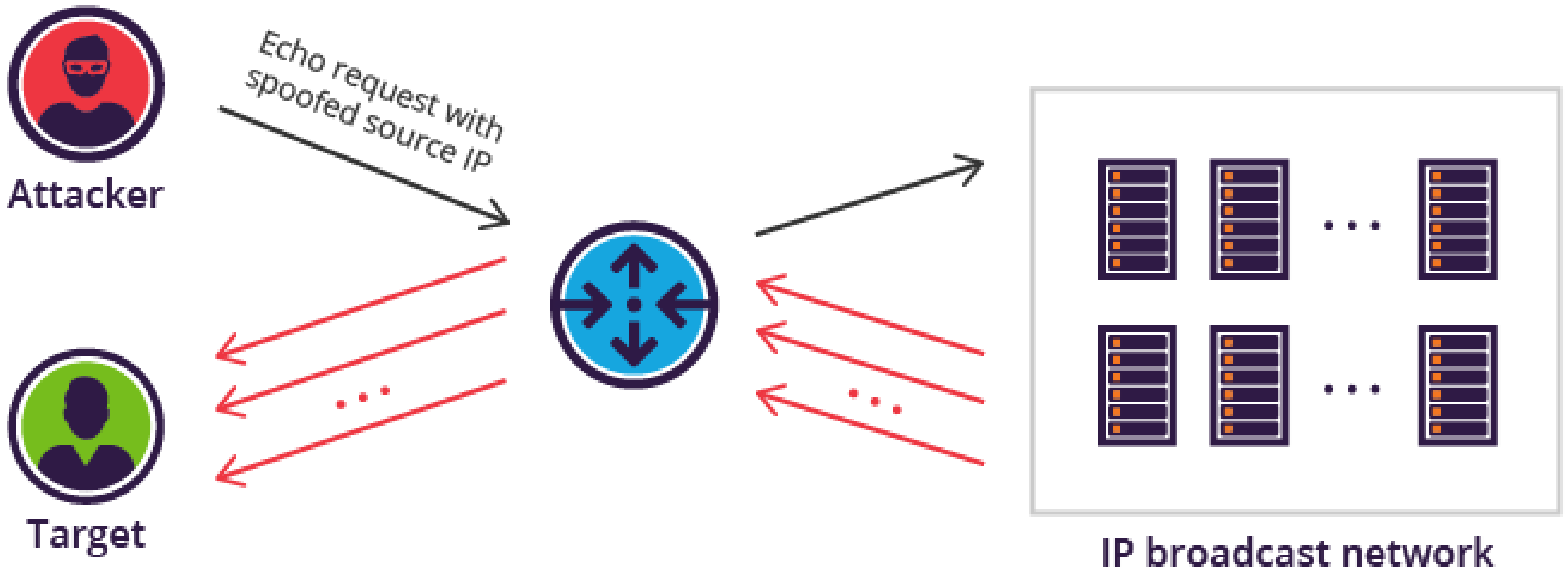
- Must have access to the network.
- Scanning the network to determine the IP addresses of connected device network.
- Attacker uses spoofing tool (i.e. Arpspoof) to forged ARP responses.
- The forged responses advertise that the correct MAC address for both IP addresses, belonging to the router and 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.



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

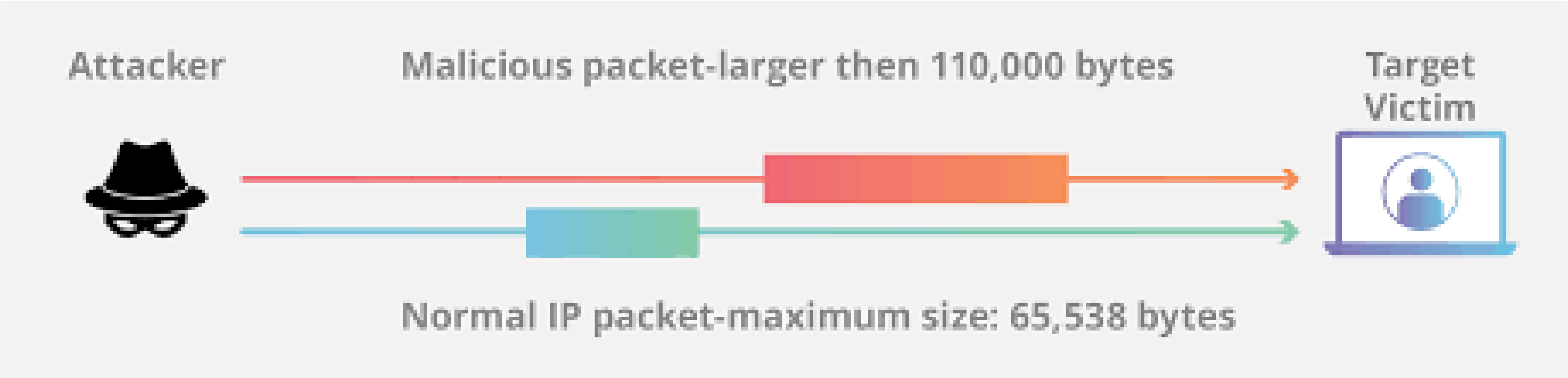
Ping of Death

- 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

References

1. Bai, Lei, Lina Yao, Salil S. Kanhere, Xianzhi Wang, and Zheng Yang. "Automatic device classification from network traffic streams of internet of things." in Proceedings of the 43rd International Conference on Local Computer Networks (LCN’18), 2018, pp. 1-9.
2. Yao, Haipeng, Pengcheng Gao, Jingjing Wang, Peiying Zhang, Chunxiao Jiang, and Zhu Han. "Capsule Network Assisted IoT Traffic Classification Mechanism for Smart Cities." IEEE Internet of Things Journal, vol. 6, pp. 7515-7525, 2019.
3. N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, “Network intrusion detection for iot security based on learning techniques,” IEEE Communications Surveys & Tutorials, vol. 21, pp. 2671– 2701, 2019.
4. Kolias, Constantinos, et al. "DDoS in the IoT: Mirai and other botnets." Computer 50.7 (2017): 80-84.
5. Sivanathan, Arunan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. "Classifying IoT devices in smart environments using network traffic characteristics." IEEE Transactions on Mobile Computing, vol. 18, pp. 1745-1759, 2019.
6. M. Frustaci, P. Pace, G. Aloi, and G. Fortino, “Evaluating critical security issues of the iot world: Present and future challenges,” IEEE Internet of Things Journal, vol. 5, pp. 2483–2495, 2017.
7. Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, “A survey on security and privacy issues in internet-of-things,” IEEE Internet of Things Journal, vol. 4, pp. 1250–1258, 2017.
8. Q. Yan, W. Huang, X. Luo, Q. Gong, and F. R. Yu, “A multi-level ddos mitigation framework for the industrial internet of things,” IEEE Communications Magazine, vol. 56, pp. 30–36, 2018.

References

9. I. Makhdoom, M. Abolhasan, J. Lipman, R. P. Liu, and W. Ni, “Anatomy of threats to the internet of things,” IEEE Communications Surveys & Tutorials, vol. 21, pp. 1636–1675, 2018.
10. J. Granjal, E. Monteiro, and J. S. Silva, “Security for the internet of things: a survey of existing protocols and open research issues,” IEEE Communications Surveys & Tutorials, vol. 17, pp. 1294–1312, 2015.
11. N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, “Demystifying iot security: an exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations,” IEEE Communications Surveys & Tutorials, pp. 2702–2733, 2019.
12. S. Murali and A. Jamalipour, “A lightweight intrusion detection for sybil attack under mobile rpl in the internet of things, ”IEEE Internet of Things Journal (Early Access), vol. 6.
13. https://blackarch.org/spoof.html
14. https://www.networkworld.com/article/2272520/six-worst-internet-routing-attacks.html
15. https://www.infosecurity-magazine.com/news/massive-bruteforce-attack-on/
16. https://canadiandimension.com/articles/view/web-exclusive-author-calls-on-ccla-board-members-to-repudiate-attack-on-dis
17. S. Li, L. Da Xu, and S. Zhao, “The internet of things: a survey,” Information Systems Frontiers, vol. 17, pp. 243–259, 2015.
18. https://posts.specterops.io/cve-2018-8414-a-case-study-in-responsible-disclosure-ff74c39615ba
19. https://www.cvedetails.com/cve/CVE-2019-0735/

References

20. A. Mosenia and N. K. Jha, “A comprehensive study of security of internet-of-things,” IEEE Transactions on Emerging Topics in Computing, vol. 5, pp. 586–602, 2016
21. https://www.kaspersky.com/blog/five-most-notorious-cyberattacks/24506/
22. https://medium.com/ledger-on-security-and-blockchain/introducing-rainbow-donjons-side-channel-analysis-simulation-tool-2f23fa1f11b3
23. Pinheiro, Antônio J., Jeandro de M. Bezerra, Caio AP Burgardt, and Divanilson R. Campelo, "Identifying IoT devices and events based on packet length from encrypted traffic" Computer Communications, vol. 144, pp. 8-17, 2019.
24. A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, “Managing iotcyber-security using programmable telemetry and machine learning,”IEEE Transactions on Network and Service Management, vol. 17, pp.60–74, 2020.
25. J. Ortiz, C. Crawford, and F. Le, “Devicemien: network device behaviour modelling for identifying unknown iot devices,” in Proceedings of the 2nd International Conference on Internet of Things Design and Implementation(IOTDI’19, 2019, pp. 106–117.

References

26. Lopez-Martin, Manuel, Belen Carro, and Antonio Sanchez-Esguevillas. "Neural network architecture based on gradient boosting for IoT traffic prediction.“ Future Generation Computer Systems , vol. 100, pp. 656-673, 2019.

27. M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, “Iot type-of-traffic forecasting method based on gradient boosting neural networks.” Future Generation Computer Systems, vol. 105, pp. 331–345, 2020.

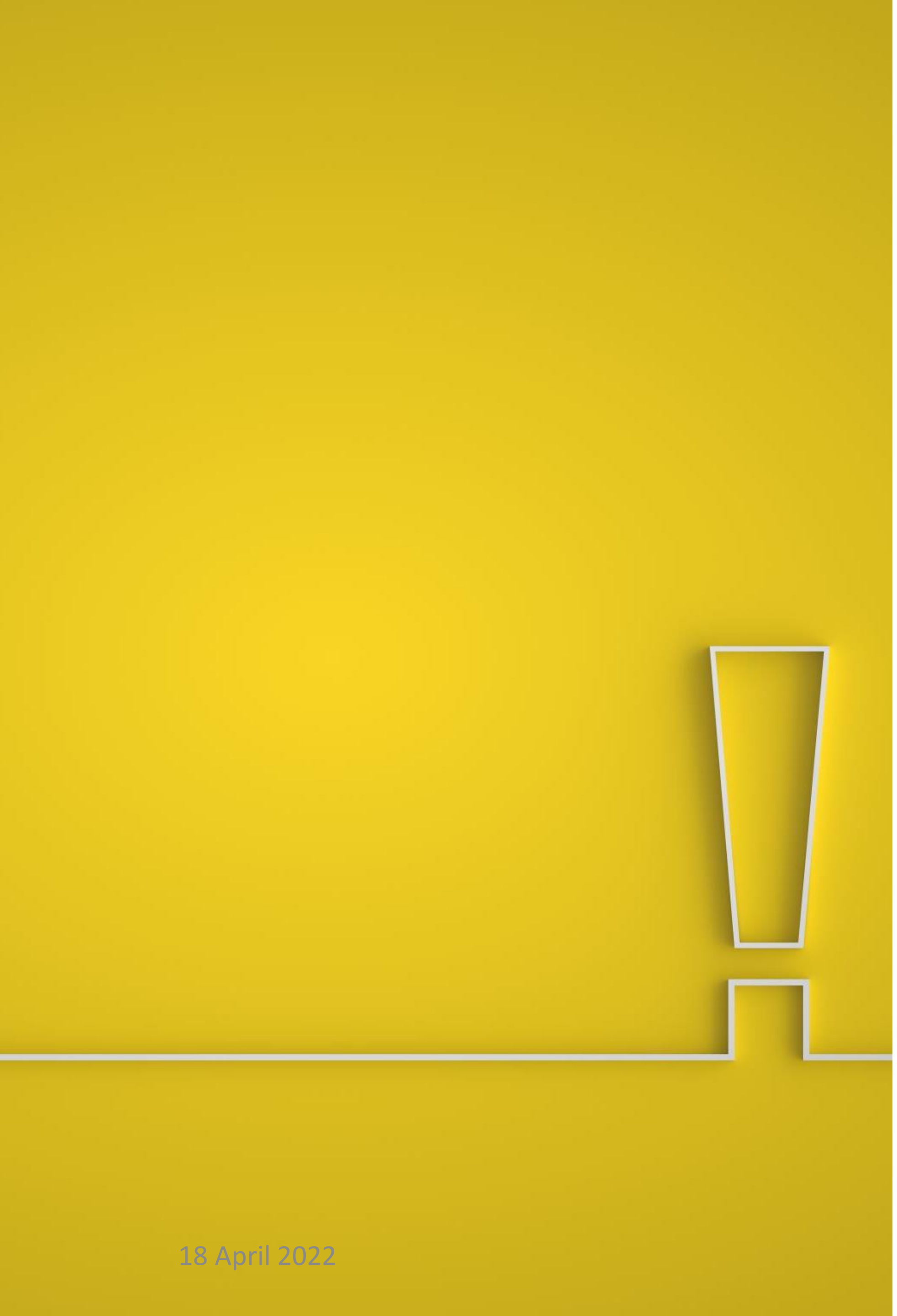
28. https://www.3pillarglobal.com/insights/approaches-tools-techniques-for-security-testing

29. https://resources.infosecinstitute.com/popular-tools-for-brute-force-attacks/#gref

30. https://www.itpro.co.uk/security/innovation-at-work/29577/the-10-best-or-should-that-be-worst-malware

31. Kolias, Constantinos, et al. "DDoS in the IoT: Mirai and other botnets." Computer 50.7, pp. 80-84, 2017

32. Nguyen, Thanh Thi, and Vijay Janapa Reddi. "Deep reinforcement learning for cyber security." *arXiv preprint arXiv:1906.05799* (2019).



Thank You

For more information, please visit the following links:

gauravsingal789@gmail.com

gaurav.singal@nsut.ac.in

https://www.linkedin.com/in/gauravsingal789/

http://www.gauravsingal.in