**DESCRIPTION**:

A network sniffer is a software tool used by network administrators and hackers and also for a person who is learning Hacking. It is also known is packet sniffer sniffers. This tool can monitor network traffic for devices, protocols and various information captured in different packets.
The main reason for this project being undertaken is to help network administrator capture and analyze network traffic and also it is used for educational purpose for a person gaining knowledge in Hacking.
Netwok sniffing or Packet sniffing is a way one can see what's going on in a network environment you can see what packets are being shared and monitor it in real time. One can do it only if he/she has access to the network environment in which they are going to sniff the packets.
Packet sniffer is a great tool to identify the network traffic and analyze the real-time traffic in a network of interconnected environment only if you have access to that network.

## METHODOLOGY

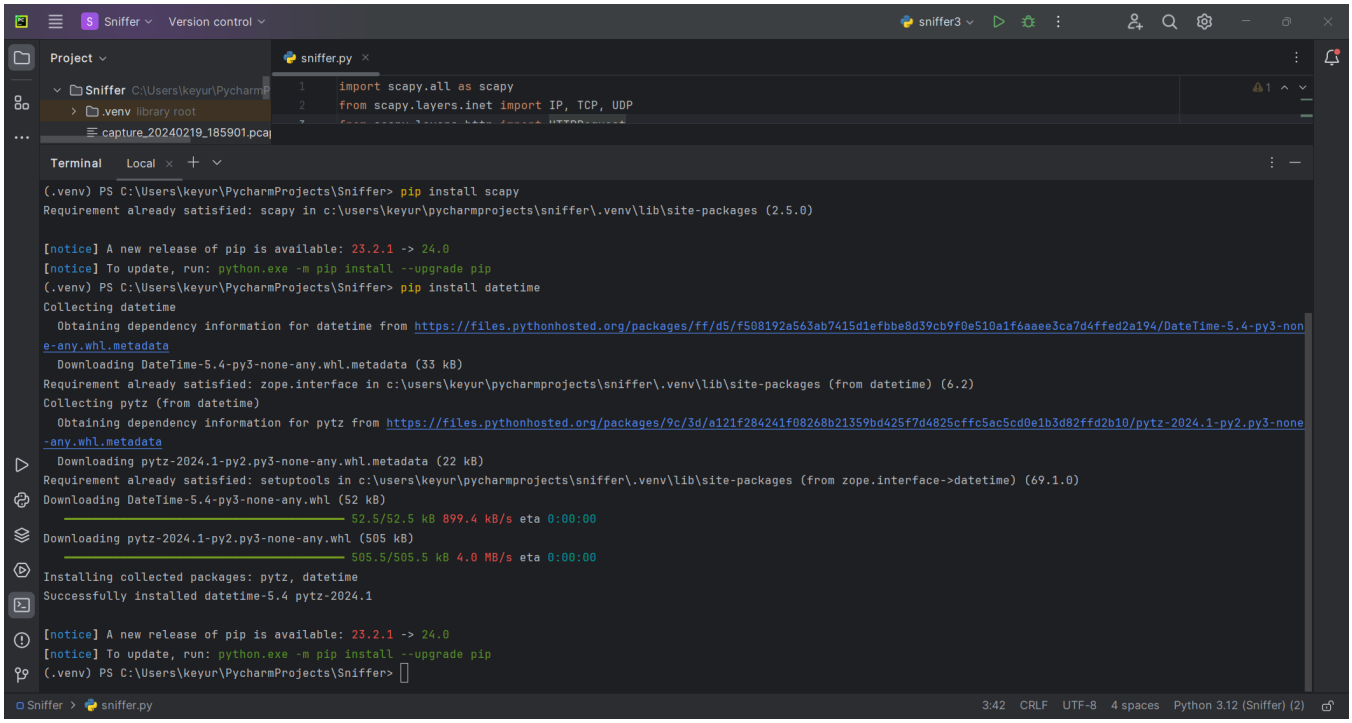
1. **TOOL/S USED:**
   A) PyCharm – version: 2023.3.3
   B) Wireshark – version: 4.2.2
   C) scapy (Python Library) – version: 2.5.0
   D) datetime (Python Library) - version: 5.4
   E) Nmap – version: 7.94

2. **TECHNIQUES**
   Used Scapy python library to implement this python code. Scapy is a packet manipulation library in Python that can sniff. Connected the device over wi-fi for sniffing the network Classified the total number of packets based on the different protocol (IP, TCP, UDP) and also the HTTP request. Lastly saved the networks sniffed in the ".pcap" format for future use.

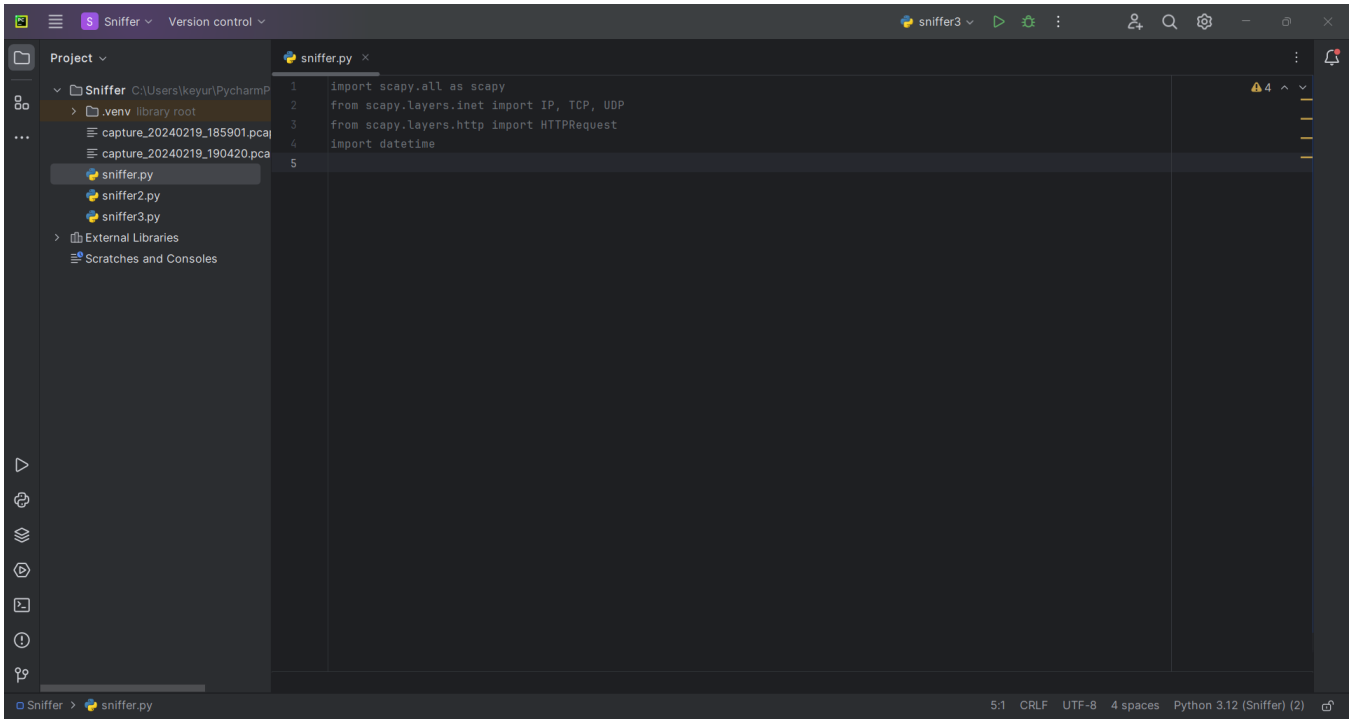
3. **TARGET SYSTEM**
   The target system will be the data packets travelling over a network. Sniffers capture these packets to analyze their contents.

## 4. PROCEDURE WITH STEPWISE SCREENSHOT

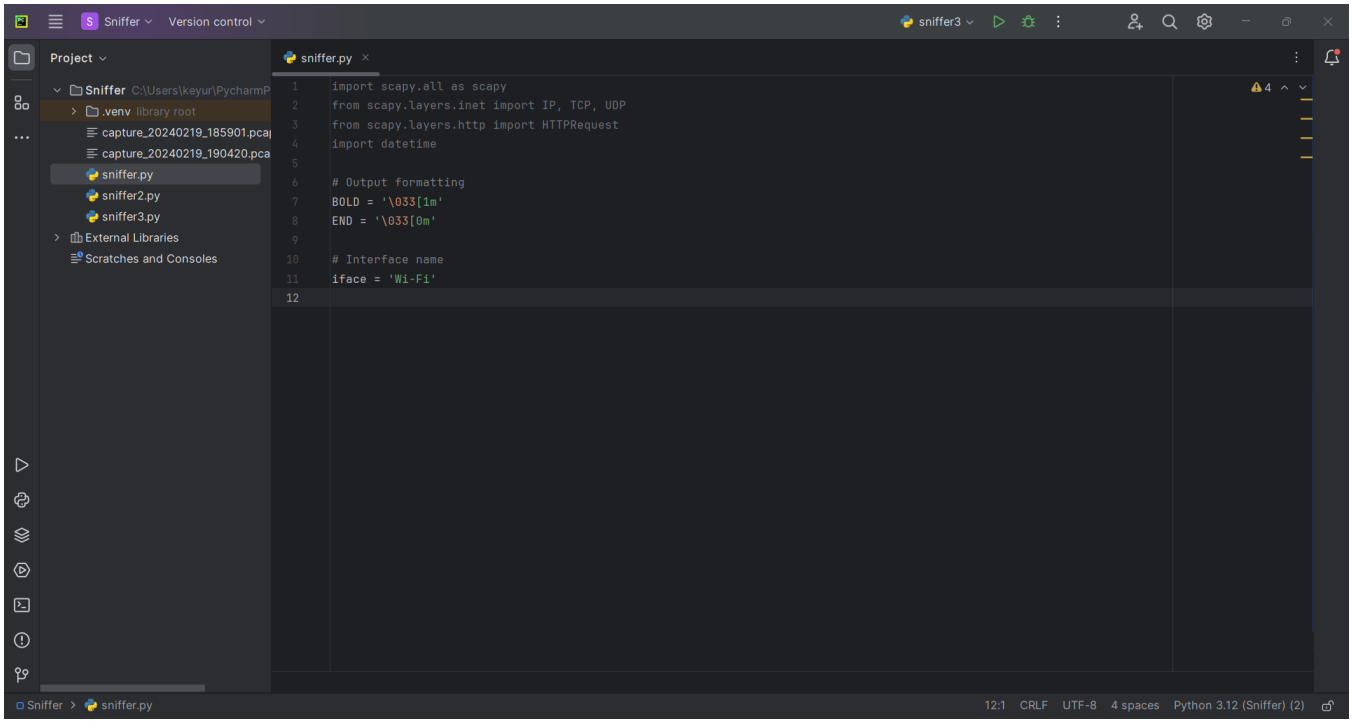
Step 1: Opened Pycharm and installed the necessary library



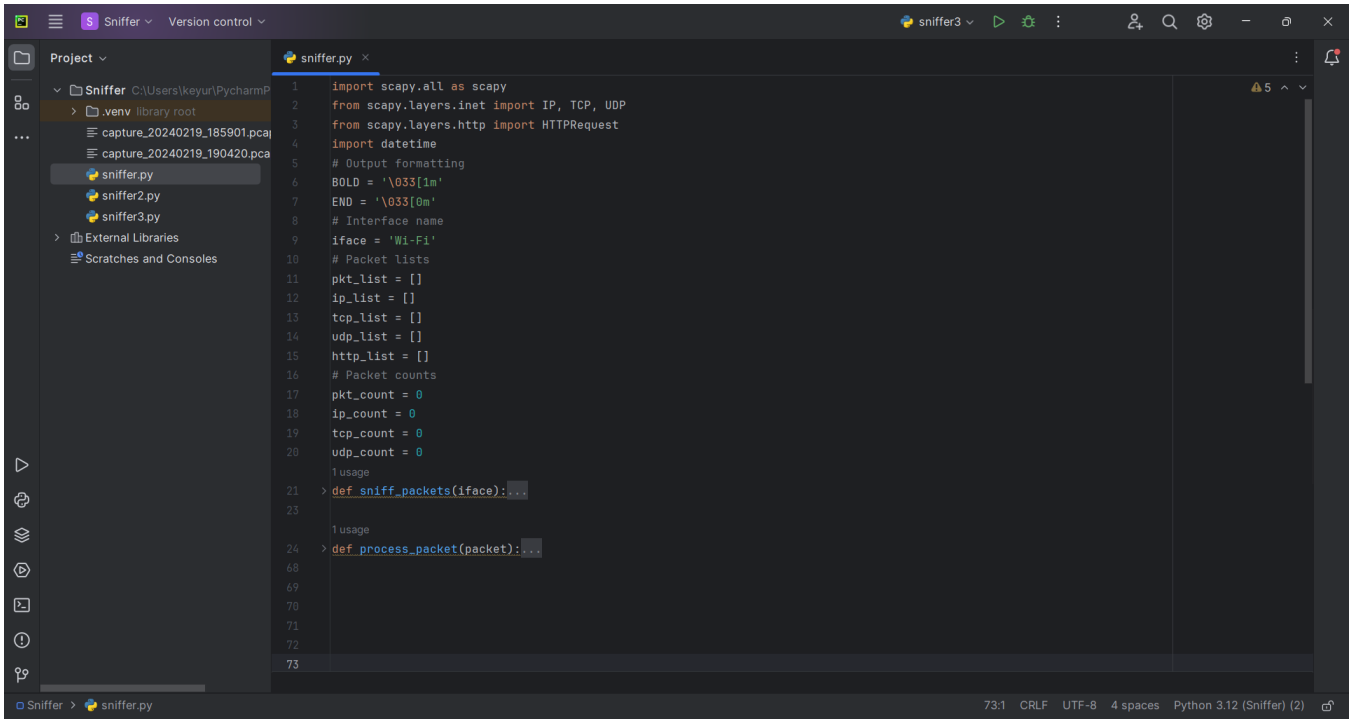

Step 2: import the necessary libraries “Scapy” and “datetime”



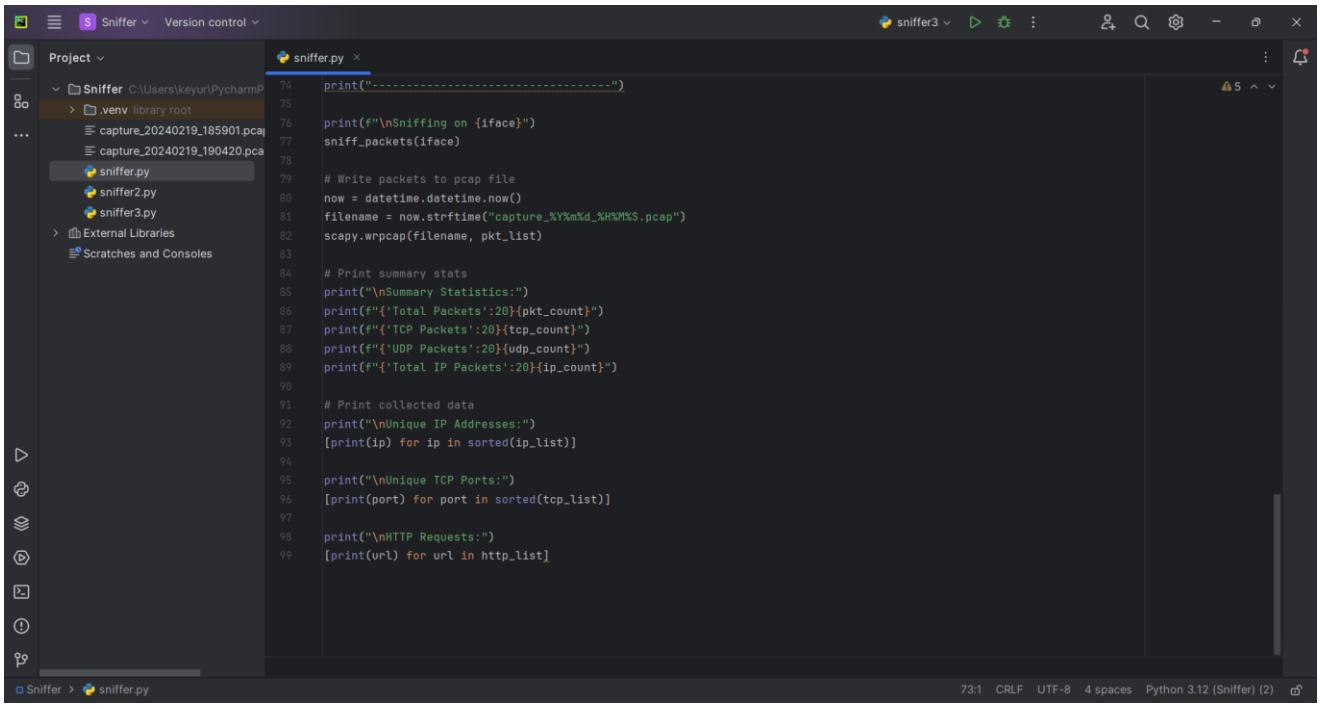
Step 3: interfaces the “wi-fi” because the packets on the wi-fi network will be sniffed.



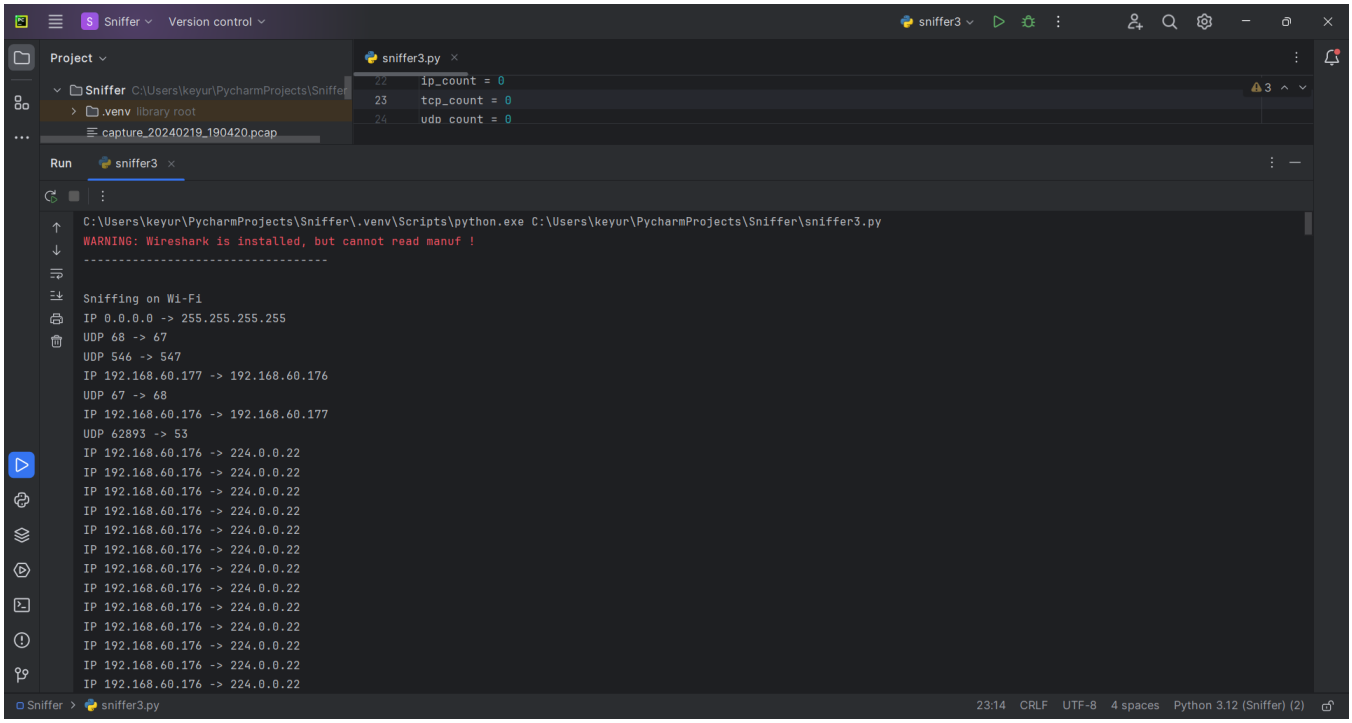

Step 4 : made packet lists and packet counts different functions required in the code

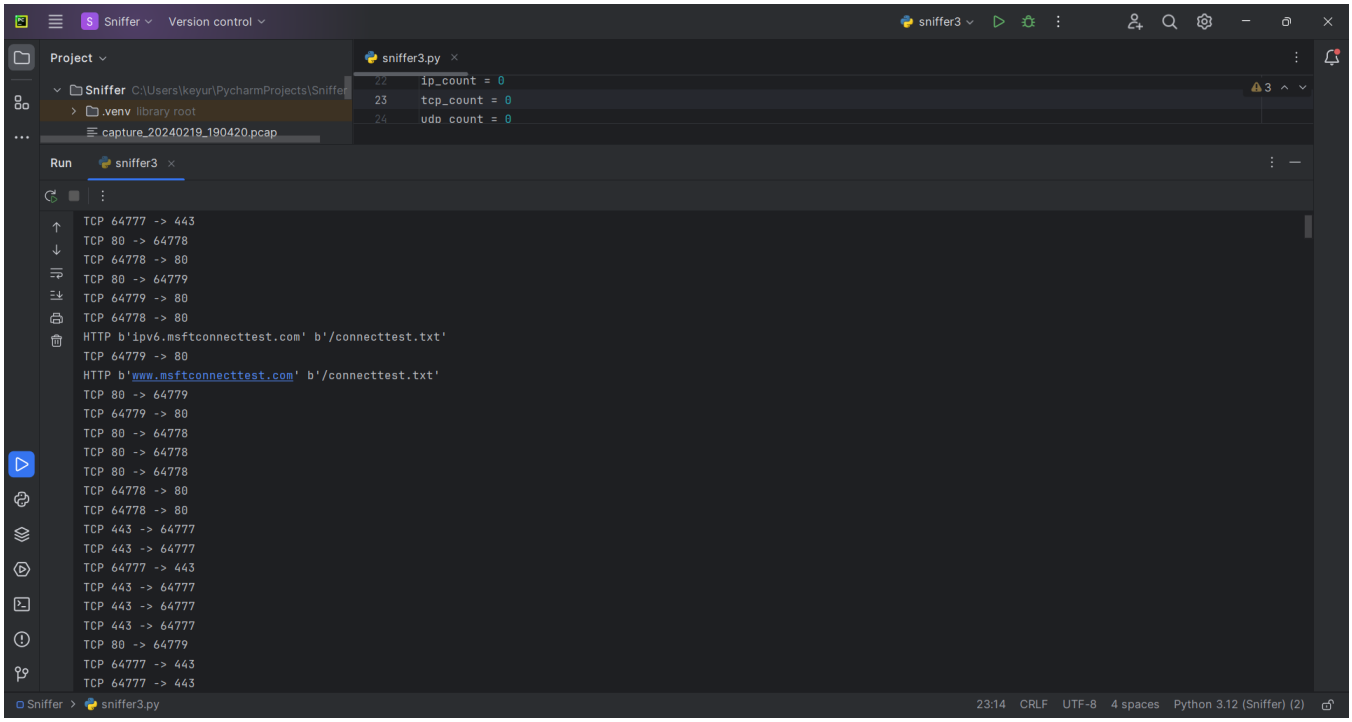


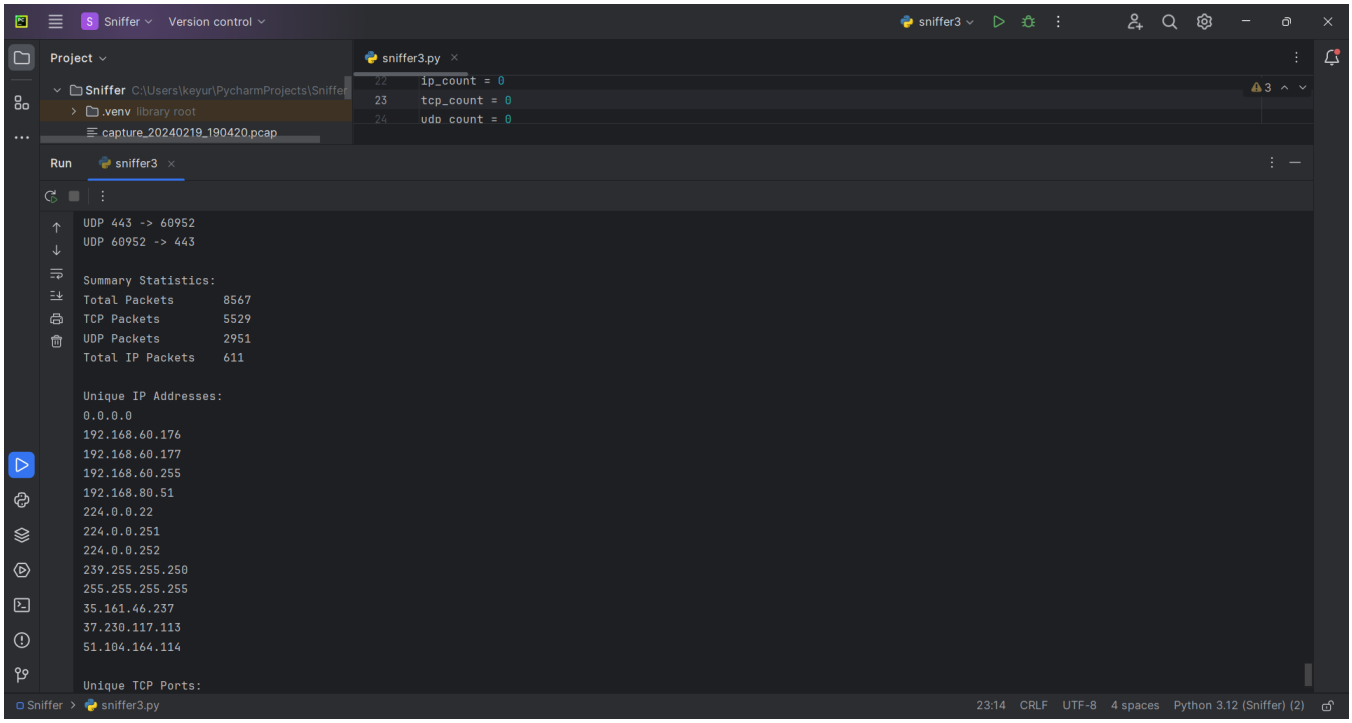
Step 5: Printed the output in a particular format and from the total number of packets filetered the TCP, UDP and Total IP Packets then printed the unique ip address then the UDP ports and printed the HTTP requests made

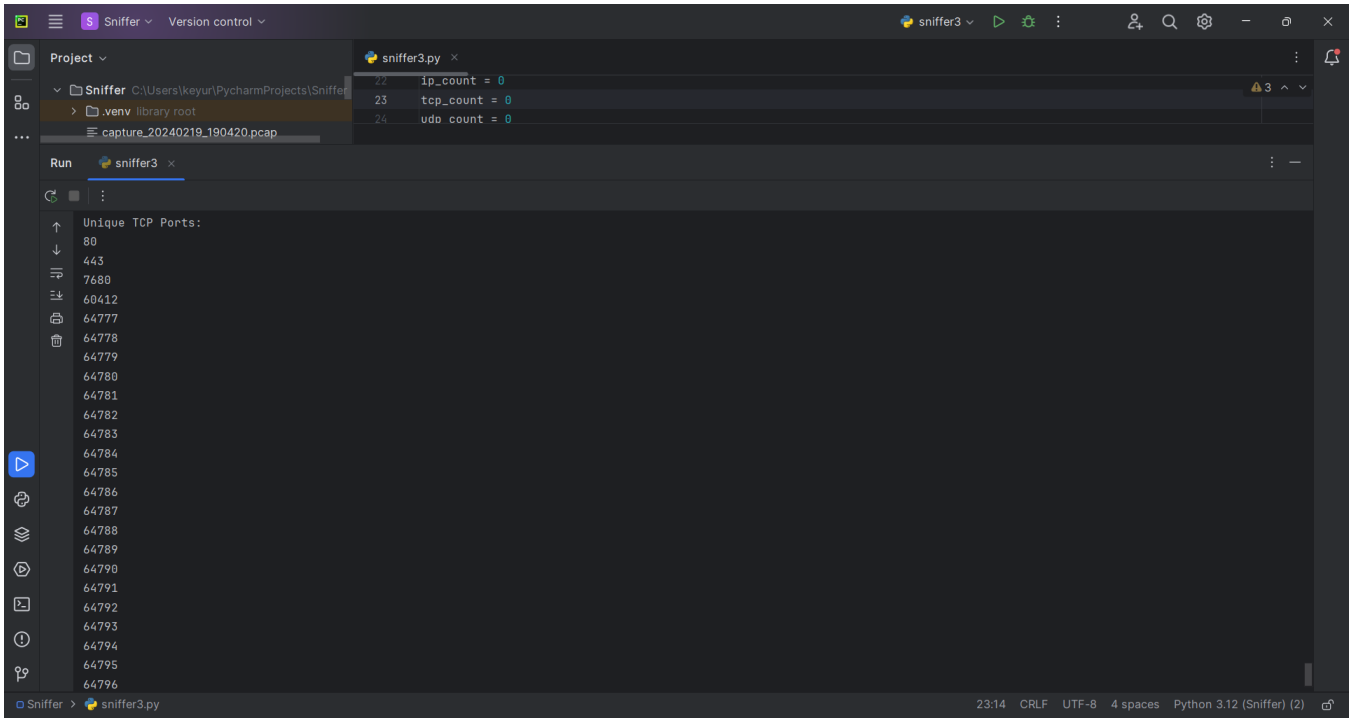


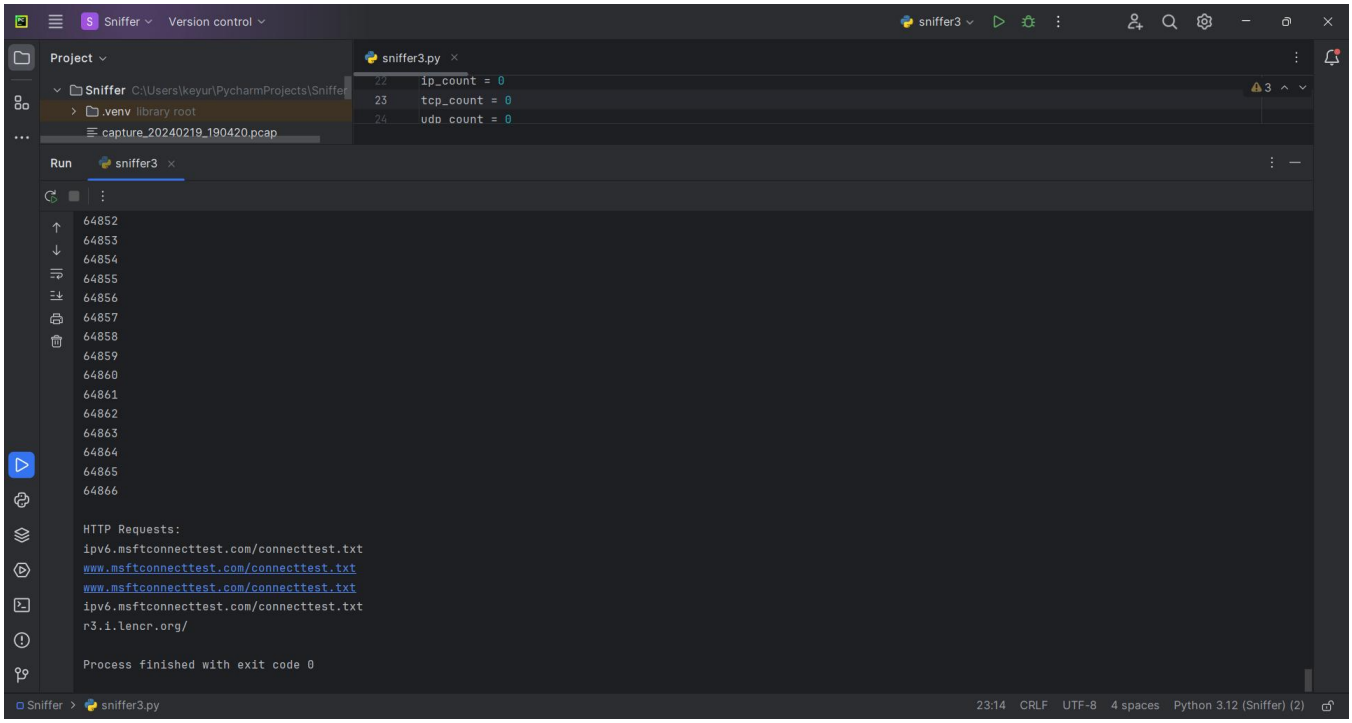

Step 6: Output in the terminal



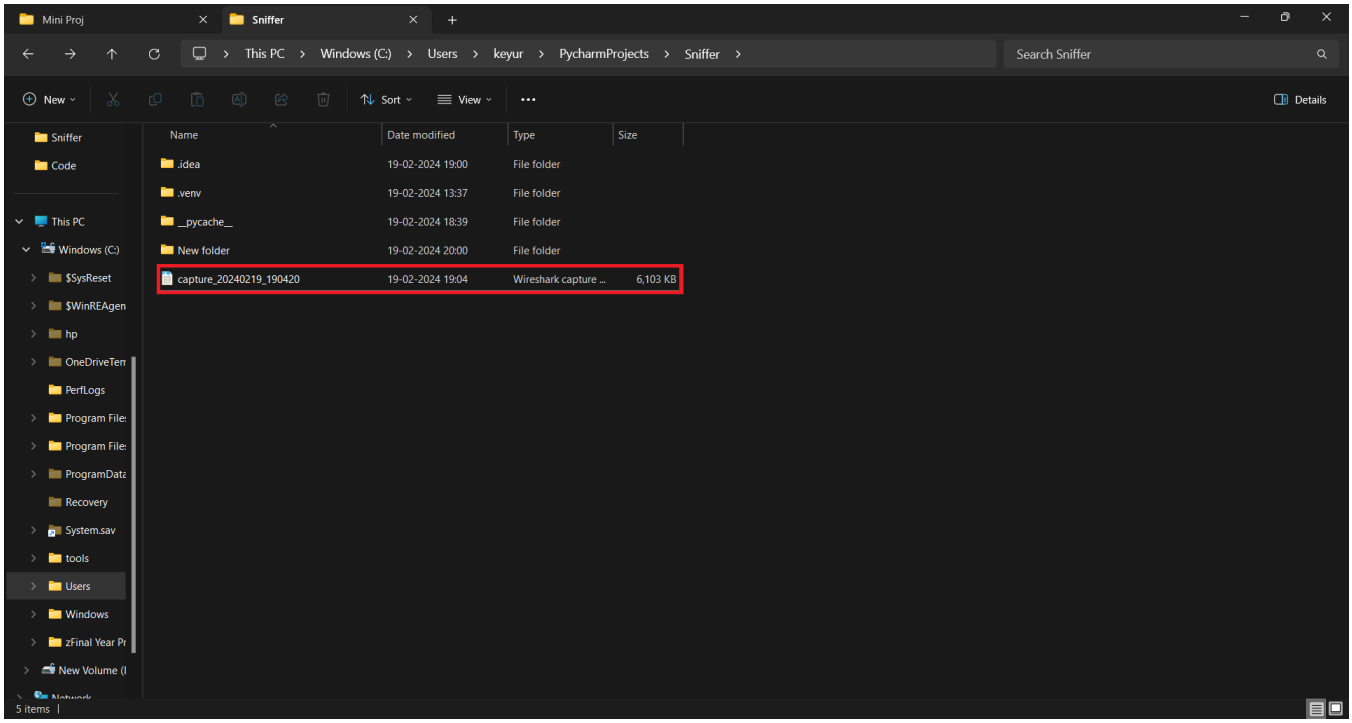

Sniffer
Version control
sniffer3
Project
sniffer3.py
Sniffer C:\Users\keyur\PycharmProjects\Sniffer
.venv library root
capture_20240219_190420.pcap
ip_count = 0
tcp_count = 0
udp_count = 0
Run
sniffer3
TCP 64777 -> 443
TCP 80 -> 64778
TCP 64778 -> 80
TCP 80 -> 64779
TCP 64779 -> 80
TCP 64778 -> 80
HTTP b'ipv6.msftconnecttest.com' b'/connecttest.txt'
TCP 64779 -> 80
HTTP b'www.msftconnecttest.com' b'/connecttest.txt'
TCP 80 -> 64779
TCP 64779 -> 80
TCP 80 -> 64778
TCP 80 -> 64778
TCP 80 -> 64778
TCP 64778 -> 80
TCP 64778 -> 80
TCP 443 -> 64777
TCP 443 -> 64777
TCP 64777 -> 443
TCP 443 -> 64777
TCP 443 -> 64777
TCP 443 -> 64777
TCP 80 -> 64779
TCP 64777 -> 443
TCP 64777 -> 443
Sniffer
sniffer3.py
23:14
CRLF
UTF-8
4 spaces
Python 3.12 (Sniffer) (2)


Sniffer
Version control
sniffer3
Project
sniffer3.py
Sniffer C:\Users\keyur\PycharmProjects\Sniffer
.venv library root
capture_20240219_190420.pcap
ip_count = 0
tcp_count = 0
udp_count = 0
Run
sniffer3
UDP 443 -> 60952
UDP 60952 -> 443
Summary Statistics:
Total Packets 8567
TCP Packets 5529
UDP Packets 2951
Total IP Packets 611
Unique IP Addresses:
0.0.0.0
192.168.60.176
192.168.60.177
192.168.60.255
192.168.80.51
224.0.0.22
224.0.0.251
224.0.0.252
239.255.255.250
255.255.255.255
35.161.46.237
37.230.117.113
51.104.164.114
Unique TCP Ports:
Sniffer
sniffer3.py
23:14
CRLF
UTF-8
4 spaces
Python 3.12 (Sniffer) (2)

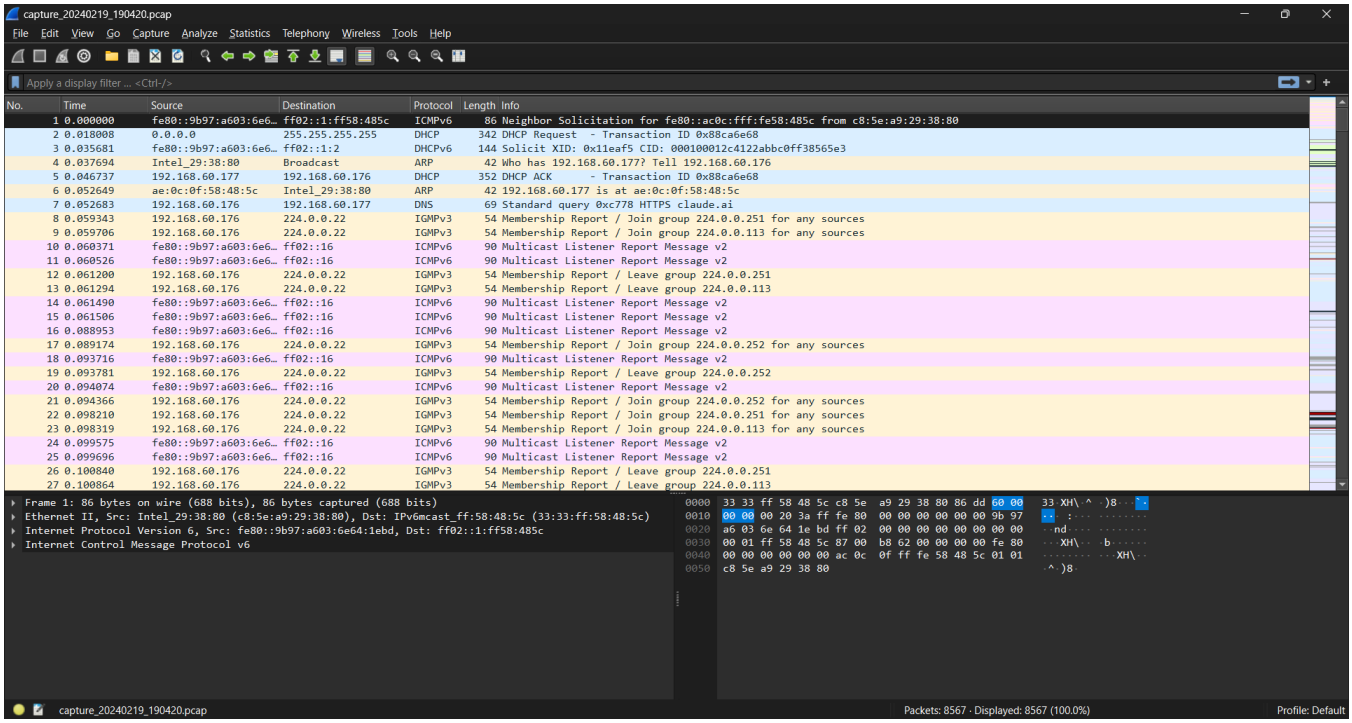

Sniffer
Version control
sniffer3
Project
Sniffer C:\Users\keyur\PycharmProjects\Sniffer
.venv library root
capture_20240219_190420.pcap
sniffer3.py
ip_count = 0
tcp_count = 0
udp_count = 0
Run
sniffer3
Unique TCP Ports:
80
443
7680
60412
64777
64778
64779
64780
64781
64782
64783
64784
64785
64786
64787
64788
64789
64790
64791
64792
64793
64794
64795
64796
Sniffer
sniffer3.py
23:14
CRLF
UTF-8
4 spaces
Python 3.12 (Sniffer) (2)


Sniffer
Version control
sniffer3
Project
Sniffer C:\Users\keyur\PycharmProjects\Sniffer
.venv library root
capture_20240219_190420.pcap
sniffer3.py
ip_count = 0
tcp_count = 0
udp_count = 0
Run
sniffer3
64852
64853
64854
64855
64856
64857
64858
64859
64860
64861
64862
64863
64864
64865
64866
HTTP Requests:
ipv6.msftconnecttest.com/connecttest.txt
www.msftconnecttest.com/connecttest.txt
www.msftconnecttest.com/connecttest.txt
ipv6.msftconnecttest.com/connecttest.txt
r3.i.lencr.org/
Process finished with exit code 0
Sniffer
sniffer3.py
23:14
CRLF
UTF-8
4 spaces
Python 3.12 (Sniffer) (2)

Step 7: After executing it automatically saves the network file in the system.




Step 8: Accessed the saved files using wireshark or such tools.



**CODE:**

```
import scapy.all as scapy
from scapy.layers.inet import IP, TCP, UDP
from scapy.layers.http import HTTPRequest
import datetime

# Output formatting
BOLD = '\033[1m'
END = '\033[0m'

# Interface name
iface = 'Wi-Fi'

# Packet lists
pkt_list = []
ip_list = []
tcp_list = []
udp_list = []
http_list = []

# Packet counts
pkt_count = 0
ip_count = 0
tcp_count = 0
udp_count = 0


def sniff_packets(iface):
    scapy.sniff(iface=iface, prn=process_packet, store=False)


def process_packet(packet):
    global pkt_count, ip_count, tcp_count, udp_count

    # Append packet to list
    pkt_list.append(packet)

    # Increment packet count
    pkt_count += 1

    if IP in packet:
        ip_count += 1

        src_ip = packet[IP].src
```

```
        dst_ip = packet[IP].dst

        # Collect unique IP addresses
        if src_ip not in ip_list:
            ip_list.append(src_ip)
        if dst_ip not in ip_list:
            ip_list.append(dst_ip)

        print(f"IP {packet[IP].src} -> {packet[IP].dst}")

    if TCP in packet:
        tcp_count += 1

        src_port = packet[TCP].sport
        dst_port = packet[TCP].dport

        # Collect unique TCP ports
        if src_port not in tcp_list:
            tcp_list.append(src_port)
        if dst_port not in tcp_list:
            tcp_list.append(dst_port)

        print(f"TCP {packet[TCP].sport} -> {packet[TCP].dport}")

    if UDP in packet:
        udp_count += 1
        print(f"UDP {packet[UDP].sport} -> {packet[UDP].dport}")

    if HTTPRequest in packet:
        http_list.append(packet[HTTPRequest].Host.decode() + packet[HTTPRequest].Path.decode())
        print(f"HTTP {packet[HTTPRequest].Host} {packet[HTTPRequest].Path}")

print("-----------------------------------")

print(f"\nSniffing on {iface}")
sniff_packets(iface)

# Write packets to pcap file
now = datetime.datetime.now()
filename = now.strftime("capture_%Y%m%d_%H%M%S.pcap")
scapy.wrpcap(filename, pkt_list)

# Print summary stats
print("\nSummary Statistics:")
print(f"{'Total Packets':20}{pkt_count}")
```

```
print(f"{'TCP Packets':20}{tcp_count}")
print(f"{'UDP Packets':20}{udp_count}")
print(f"{'Total IP Packets':20}{ip_count}")

# Print collected data
print("\nUnique IP Addresses:")
[print(ip) for ip in sorted(ip_list)]

print("\nUnique TCP Ports:")
[print(port) for port in sorted(tcp_list)]

print("\nHTTP Requests:")
[print(url) for url in http_list]
```

## RESULTS AND FINDINGS

Sniffing or network packet sniffing is the process of monitoring and capturing all the packets passing through a given network using sniffing tools. There is so much possibility that if a set of enterprise switch ports is open, then one of their employees can sniff the whole traffic of the network. Anyone in the same physical location can plug into the network using Ethernet cable or connect wirelessly to that network and sniff the total traffic.In other words, Sniffing allows you to see all sorts of traffic, both protected and unprotected. In the right conditions and with the right protocols in place, an attacking party may be able to gather information that can be used for further attacks or to cause other issues for the network or system owner.

## REFERENCES


https://www.youtube.com/playlist?list=PL6gx4Cwl9DGDdduy0IPDDHYnUx66Vc4ed

https://youtu.be/WMmVheaE0xE?si=3P3-kemej80xOAhD

https://github.com/shrestha-tripathi/offensive-python/blob/master/packet_sniffer.py

https://github.com/EONRaider/Packet-Sniffer

https://betterprogramming.pub/building-a-packet-sniffing-tool-with-python-58ea5d65ace2