Network Sniffer with Python

Muhammad Shoaib Ishaq Khan

LinkedIn Profile

Table of Contents

Network Sniffer in Python using Scapy: Project Report

1. Introduction

1.1 Overview of Network Sniffing

Network sniffing refers to the process of intercepting and logging traffic passing over a computer network. This is useful for network administrators and security professionals who need to monitor and analyze network traffic for issues such as unauthorized access or bandwidth misuse. The goal is to capture packets in transit and examine them for patterns, errors, or malicious activity.

1.2 Purpose of the Project

The primary purpose of this project is to develop a network sniffer using **Python** and the **Scapy** library. This sniffer will capture and analyze network traffic, enabling the identification of different protocols and packet structures. By analyzing captured data, users can understand how data flows through their network, troubleshoot issues, and detect security concerns.

1.3 Scope and Objectives

This project will focus on:

- Capturing network packets using Scapy.
- Filtering specific types of traffic.
- Saving captured packets to a PCAP file.
- Analyzing the packets using tools like **Wireshark**.

1.4 Screenshot(s): Network Sniffer Script

---

2. Tools and Technologies Used

2.1 Python and Scapy

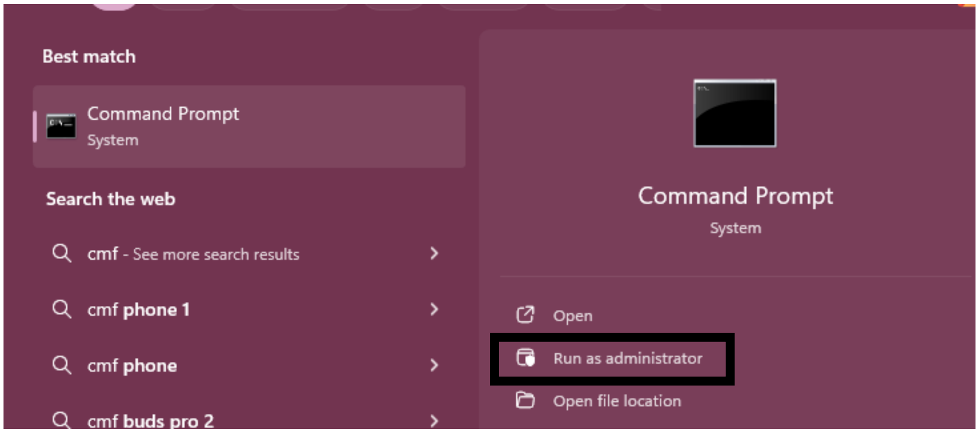
- **Python**: Python is a high-level programming language that provides an easy-to-understand syntax and rich libraries. For this project, Python was used to develop the sniffer due to its versatility and extensive support for networking tasks.

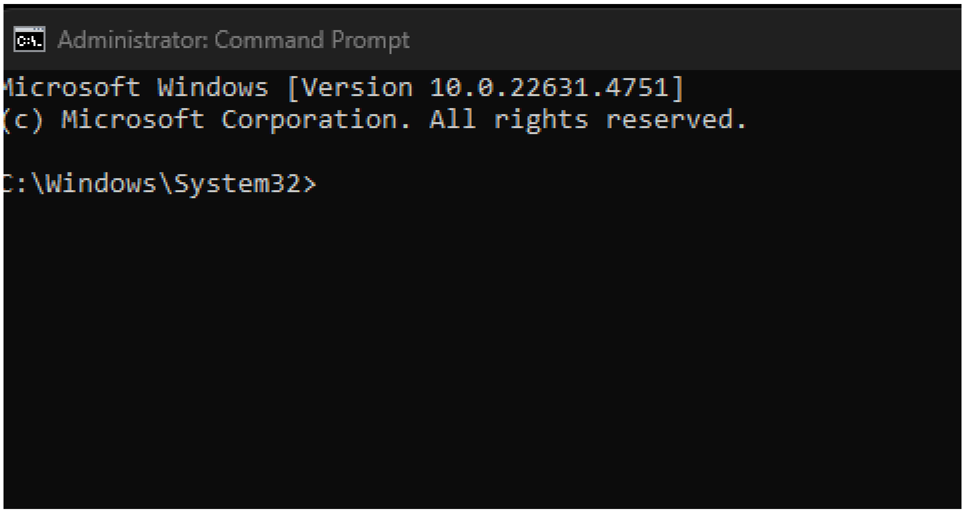
- **Scapy**: Scapy is a powerful Python library used for packet crafting and network analysis. It allows users to capture, manipulate, and send network packets easily. Scapy provides a convenient interface for sniffing network traffic at different layers of the OSI model.

2.2 Wireshark for Packet Analysis

Wireshark is a network protocol analyzer used for packet analysis. It can interpret PCAP files generated by network sniffers like Scapy and present the data in a readable form, showing detailed packet contents such as headers and payloads.

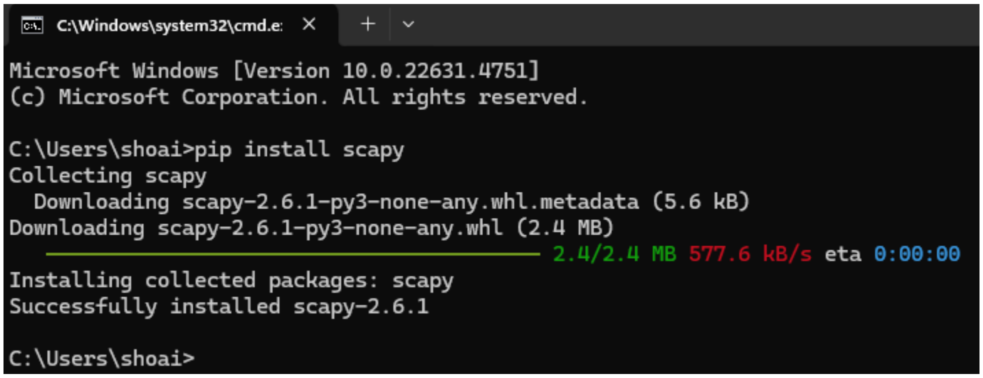
2.3 Screenshot(s): Scapy Installation in Terminal





```
licrosoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

:\Windows\System32>
```



```
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\shoai>pip install scapy
Collecting scapy
  Downloading scapy-2.6.1-py3-none-any.whl.metadata (5.6 kB)
Downloading scapy-2.6.1-py3-none-any.whl (2.4 MB)
   ---------------------------------------- 2.4/2.4 MB 577.6 kB/s eta 0:00:00
Installing collected packages: scapy
Successfully installed scapy-2.6.1

C:\Users\shoai>
```

3. System Requirements

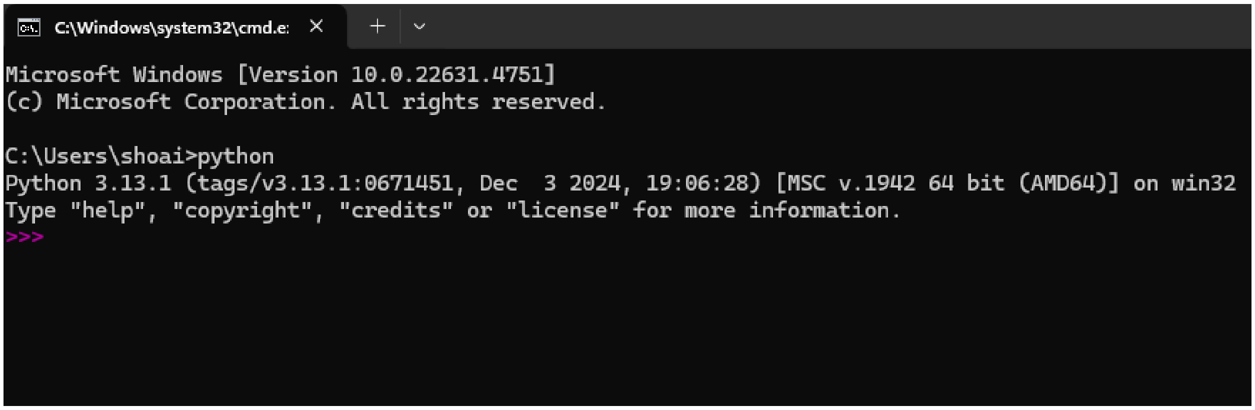
3.1 Software Requirements

- **Python 3.x**: Python programming language installed on the system.
- **Scapy**: Python library for packet manipulation and sniffing.
- **Wireshark**: Software for packet analysis and visualization.
- **Windows/Linux OS**: Compatible operating systems for running the sniffer.

3.2 Hardware Requirements

- A computer with a network interface card (NIC) capable of capturing network traffic.
- Sufficient memory and processing power to handle large network traffic captures.

3.3 Screenshot(s): Python Environment



```
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\shoai>python
Python 3.13.1 (tags/v3.13.1:0671451, Dec  3 2024, 19:06:28) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

4. Network Sniffer Script Development

4.1 Overview of the Script

The network sniffer script captures packets from a specified network interface using the Scapy library. It processes each packet to extract relevant information such as source/destination IPs, protocols, and more. The captured packets are saved in a PCAP file for further analysis.

4.2 Code Walkthrough:

```
import scapy.all as scapy



# Function to process captured packets

def packet_callback(packet):

    print(packet.summary())



# Function to start sniffing on the chosen network interface

def start_sniffing(interface):

    scapy.sniff(iface=interface, store=False, prn=packet_callback)



# Request the user to enter the network interface to sniff on

interface = input("Enter the network interface to sniff on (e.g., WiFi 2): ")

start_sniffing(interface)
```

4.3 Screenshot(s): Python Code for Network Sniffer

---

5. Packet Capture and Analysis

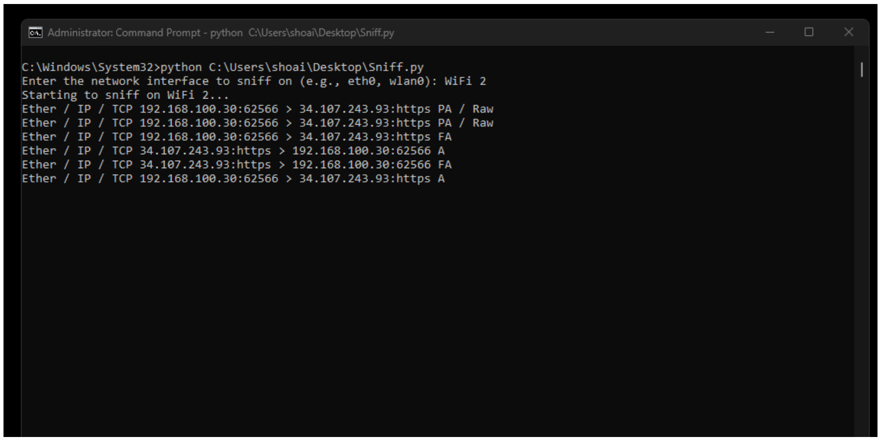
5.1 Running the Sniffer

Once the script is executed, it starts sniffing packets from the network interface specified by the user. The script runs continuously and processes packets as they are captured.

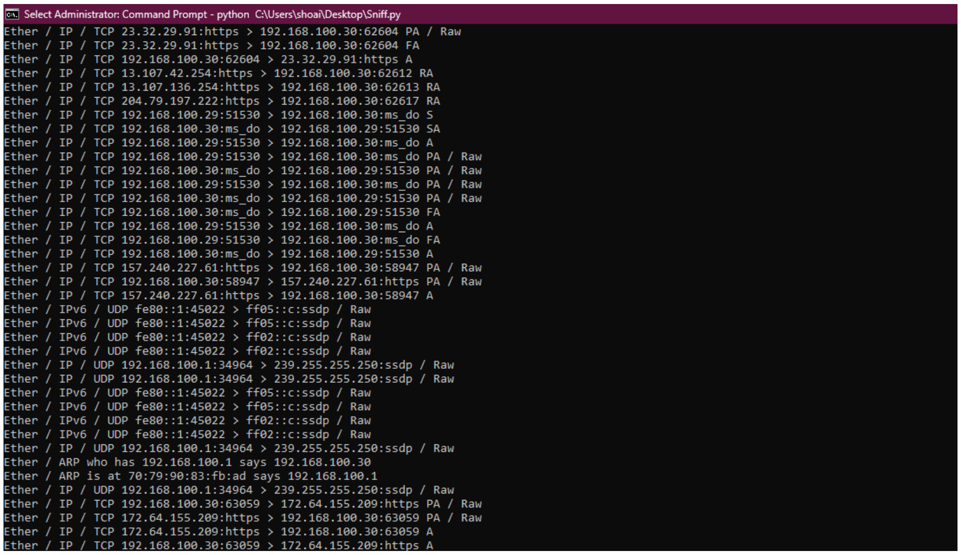
5.2 Capturing Network Packets

Packets are captured from the specified interface and displayed in the terminal using the packet.summary() function. Each packet provides a summary of the network layer data, including source and destination IP addresses, protocol information, etc.

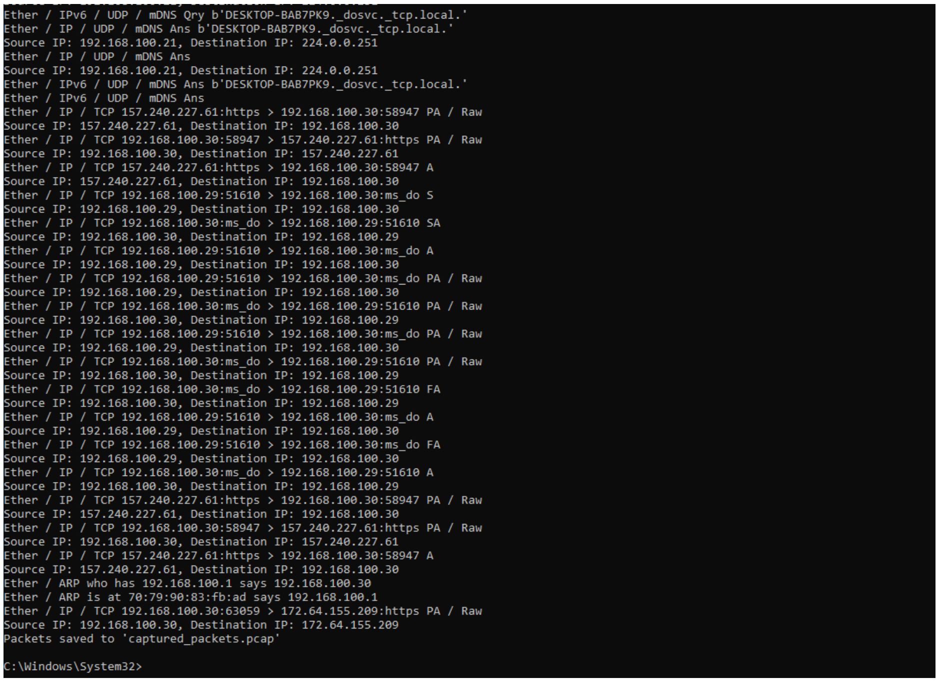
5.3 Screenshot(s): Network Sniffer Output in Terminal


Administrator: Command Prompt - python C:\Users\shoai\Desktop\Sniff.py

```
C:\Windows\System32>python C:\Users\shoai\Desktop\Sniff.py
Enter the network interface to sniff on (e.g., eth0, wlan0): WiFi 2
Starting to sniff on WiFi 2...
Ether / IP / TCP 192.168.100.30:62566 > 34.107.243.93:https PA / Raw
Ether / IP / TCP 192.168.100.30:62566 > 34.107.243.93:https PA / Raw
Ether / IP / TCP 192.168.100.30:62566 > 34.107.243.93:https FA
Ether / IP / TCP 34.107.243.93:https > 192.168.100.30:62566 A
Ether / IP / TCP 34.107.243.93:https > 192.168.100.30:62566 FA
Ether / IP / TCP 192.168.100.30:62566 > 34.107.243.93:https A
```


Select Administrator: Command Prompt - python C:\Users\shoai\Desktop\Sniff.py

```
Ether / IP / TCP 23.32.29.91:https > 192.168.100.30:62604 PA / Raw
Ether / IP / TCP 23.32.29.91:https > 192.168.100.30:62604 FA
Ether / IP / TCP 192.168.100.30:62604 > 23.32.29.91:https A
Ether / IP / TCP 13.107.42.254:https > 192.168.100.30:62612 RA
Ether / IP / TCP 13.107.136.254:https > 192.168.100.30:62613 RA
Ether / IP / TCP 204.79.197.222:https > 192.168.100.30:62617 RA
Ether / IP / TCP 192.168.100.29:51530 > 192.168.100.30:ms_do S
Ether / IP / TCP 192.168.100.30:ms_do > 192.168.100.29:51530 SA
Ether / IP / TCP 192.168.100.29:51530 > 192.168.100.30:ms_do A
Ether / IP / TCP 192.168.100.29:51530 > 192.168.100.30:ms_do PA / Raw
Ether / IP / TCP 192.168.100.30:ms_do > 192.168.100.29:51530 PA / Raw
Ether / IP / TCP 192.168.100.29:51530 > 192.168.100.30:ms_do PA / Raw
Ether / IP / TCP 192.168.100.30:ms_do > 192.168.100.29:51530 PA / Raw
Ether / IP / TCP 192.168.100.30:ms_do > 192.168.100.29:51530 FA
Ether / IP / TCP 192.168.100.29:51530 > 192.168.100.30:ms_do A
Ether / IP / TCP 192.168.100.29:51530 > 192.168.100.30:ms_do FA
Ether / IP / TCP 192.168.100.30:ms_do > 192.168.100.29:51530 A
Ether / IP / TCP 157.240.227.61:https > 192.168.100.30:58947 PA / Raw
Ether / IP / TCP 192.168.100.30:58947 > 157.240.227.61:https PA / Raw
Ether / IP / TCP 157.240.227.61:https > 192.168.100.30:58947 A
Ether / IPv6 / UDP fe80::1:45022 > ff05::c:ssdp / Raw
Ether / IPv6 / UDP fe80::1:45022 > ff05::c:ssdp / Raw
Ether / IPv6 / UDP fe80::1:45022 > ff02::c:ssdp / Raw
Ether / IPv6 / UDP fe80::1:45022 > ff02::c:ssdp / Raw
Ether / IP / UDP 192.168.100.1:34964 > 239.255.255.250:ssdp / Raw
Ether / IP / UDP 192.168.100.1:34964 > 239.255.255.250:ssdp / Raw
Ether / IPv6 / UDP fe80::1:45022 > ff05::c:ssdp / Raw
Ether / IPv6 / UDP fe80::1:45022 > ff05::c:ssdp / Raw
Ether / IPv6 / UDP fe80::1:45022 > ff02::c:ssdp / Raw
Ether / IPv6 / UDP fe80::1:45022 > ff02::c:ssdp / Raw
Ether / IP / UDP 192.168.100.1:34964 > 239.255.255.250:ssdp / Raw
Ether / ARP who has 192.168.100.1 says 192.168.100.30
Ether / ARP is at 70:79:90:83:fb:ad says 192.168.100.1
Ether / IP / UDP 192.168.100.1:34964 > 239.255.255.250:ssdp / Raw
Ether / IP / TCP 192.168.100.30:63059 > 172.64.155.209:https PA / Raw
Ether / IP / TCP 172.64.155.209:https > 192.168.100.30:63059 PA / Raw
Ether / IP / TCP 172.64.155.209:https > 192.168.100.30:63059 A
Ether / IP / TCP 192.168.100.30:63059 > 172.64.155.209:https A
```



```
Ether / IPv6 / UDP / mDNS Qry b'DESKTOP-BAB7PK9._dosvc._tcp.local.'
Ether / IP / UDP / mDNS Ans b'DESKTOP-BAB7PK9._dosvc._tcp.local.'
Source IP: 192.168.100.21, Destination IP: 224.0.0.251
Ether / IP / UDP / mDNS Ans
Source IP: 192.168.100.21, Destination IP: 224.0.0.251
Ether / IPv6 / UDP / mDNS Ans b'DESKTOP-BAB7PK9._dosvc._tcp.local.'
Ether / IPv6 / UDP / mDNS Ans
Ether / IP / TCP 157.240.227.61:https > 192.168.100.30:58947 PA / Raw
Source IP: 157.240.227.61, Destination IP: 192.168.100.30
Ether / IP / TCP 192.168.100.30:58947 > 157.240.227.61:https PA / Raw
Source IP: 192.168.100.30, Destination IP: 157.240.227.61
Ether / IP / TCP 157.240.227.61:https > 192.168.100.30:58947 A
Source IP: 157.240.227.61, Destination IP: 192.168.100.30
Ether / IP / TCP 192.168.100.29:51610 > 192.168.100.30:ms_do S
Source IP: 192.168.100.29, Destination IP: 192.168.100.30
Ether / IP / TCP 192.168.100.30:ms_do > 192.168.100.29:51610 SA
Source IP: 192.168.100.30, Destination IP: 192.168.100.29
Ether / IP / TCP 192.168.100.29:51610 > 192.168.100.30:ms_do A
Source IP: 192.168.100.29, Destination IP: 192.168.100.30
Ether / IP / TCP 192.168.100.29:51610 > 192.168.100.30:ms_do PA / Raw
Source IP: 192.168.100.29, Destination IP: 192.168.100.30
Ether / IP / TCP 192.168.100.30:ms_do > 192.168.100.29:51610 PA / Raw
Source IP: 192.168.100.30, Destination IP: 192.168.100.29
Ether / IP / TCP 192.168.100.29:51610 > 192.168.100.30:ms_do PA / Raw
Source IP: 192.168.100.29, Destination IP: 192.168.100.30
Ether / IP / TCP 192.168.100.30:ms_do > 192.168.100.29:51610 PA / Raw
Source IP: 192.168.100.30, Destination IP: 192.168.100.29
Ether / IP / TCP 192.168.100.30:ms_do > 192.168.100.29:51610 FA
Source IP: 192.168.100.30, Destination IP: 192.168.100.29
Ether / IP / TCP 192.168.100.29:51610 > 192.168.100.30:ms_do A
Source IP: 192.168.100.29, Destination IP: 192.168.100.30
Ether / IP / TCP 192.168.100.29:51610 > 192.168.100.30:ms_do FA
Source IP: 192.168.100.29, Destination IP: 192.168.100.30
Ether / IP / TCP 192.168.100.30:ms_do > 192.168.100.29:51610 A
Source IP: 192.168.100.30, Destination IP: 192.168.100.29
Ether / IP / TCP 157.240.227.61:https > 192.168.100.30:58947 PA / Raw
Source IP: 157.240.227.61, Destination IP: 192.168.100.30
Ether / IP / TCP 192.168.100.30:58947 > 157.240.227.61:https PA / Raw
Source IP: 192.168.100.30, Destination IP: 157.240.227.61
Ether / IP / TCP 157.240.227.61:https > 192.168.100.30:58947 A
Source IP: 157.240.227.61, Destination IP: 192.168.100.30
Ether / ARP who has 192.168.100.1 says 192.168.100.30
Ether / ARP is at 70:79:90:83:fb:ad says 192.168.100.1
Ether / IP / TCP 192.168.100.30:63059 > 172.64.155.209:https PA / Raw
Source IP: 192.168.100.30, Destination IP: 172.64.155.209
Packets saved to 'captured_packets.pcap'

C:\Windows\System32>
```

5.4 Packet Details

Captured packets include various types such as ARP requests, IP packets, and TCP/UDP communication. Some examples include:

- **ARP Requests/Replies**: Used to map IP addresses to MAC addresses on the network.
- **TCP/UDP Packets**: Data sent between devices on the network, including HTTP, HTTPS, etc.

---

6. Saving Packets to a PCAP File

6.1 PCAP File Format

PCAP (Packet Capture) files are used to store captured network data. They can be opened in tools like Wireshark to perform deeper analysis. Scapy allows users to write captured packets to a PCAP file using the wrpcap() function.

6.2 Storing Packets Using Scapy

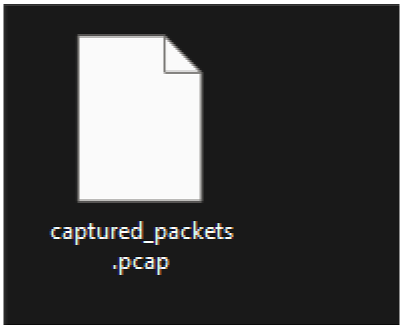
Captured packets can be stored in a .pcap file on the local system for future analysis:

from scapy.utils import wrpcap

# Save packets to a PCAP file

wrpcap('captured_packets.pcap', packets)

6.3 Screenshot(s): Saved PCAP File Confirmation


captured_packets
.pcap

---

7. Packet Analysis with Wireshark

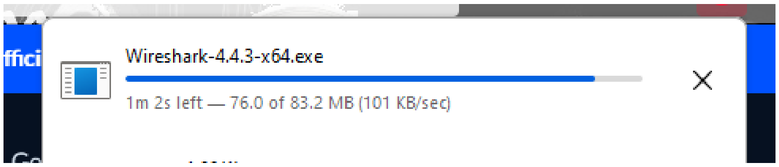
7.1 Opening PCAP File in Wireshark

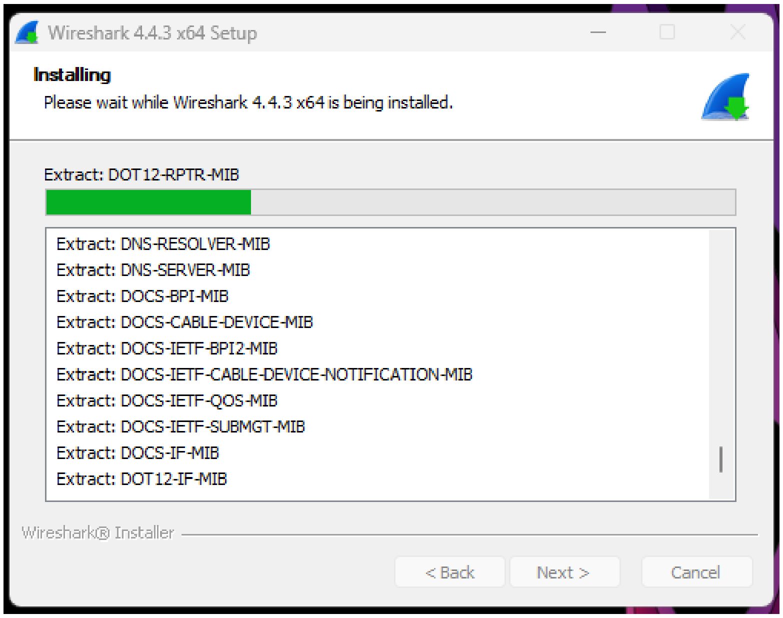
Once packets are saved to a PCAP file, the file can be opened in Wireshark for detailed packet analysis. Wireshark allows users to filter and view packet details such as protocol headers, payloads, and timestamps.

7.2 Analyzing Captured Packets

Wireshark provides powerful filtering options, making it easy to analyze specific types of packets or protocols, such as HTTP, DNS, or TCP traffic.

7.3 Screenshot(s): Wireshark Packet Analysis


Wireshark-4.4.3-x64.exe
1m 2s left — 76.0 of 83.2 MB (101 KB/sec)



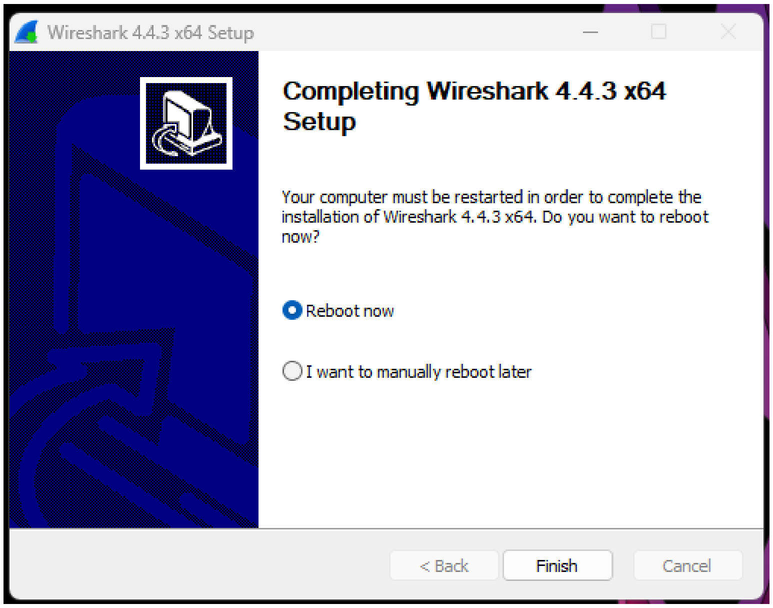
Installing

Please wait while Wireshark 4.4.3 x64 is being installed.

Extract: DOT12-RPTR-MIB

```
Extract: DNS-RESOLVER-MIB
Extract: DNS-SERVER-MIB
Extract: DOCS-BPI-MIB
Extract: DOCS-CABLE-DEVICE-MIB
Extract: DOCS-IETF-BPI2-MIB
Extract: DOCS-IETF-CABLE-DEVICE-NOTIFICATION-MIB
Extract: DOCS-IETF-QOS-MIB
Extract: DOCS-IETF-SUBMGT-MIB
Extract: DOCS-IF-MIB
Extract: DOT12-IF-MIB
```

Wireshark® Installer

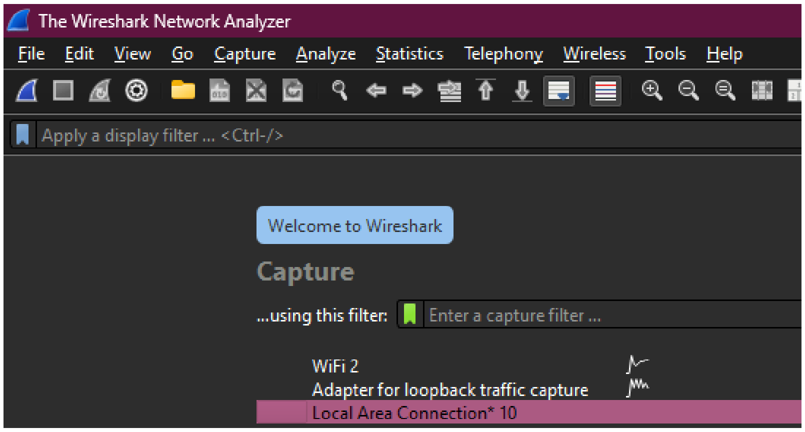


Completing Wireshark 4.4.3 x64 Setup

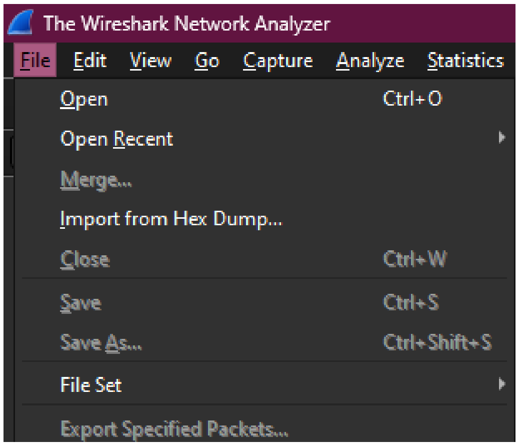
Your computer must be restarted in order to complete the installation of Wireshark 4.4.3 x64. Do you want to reboot now?

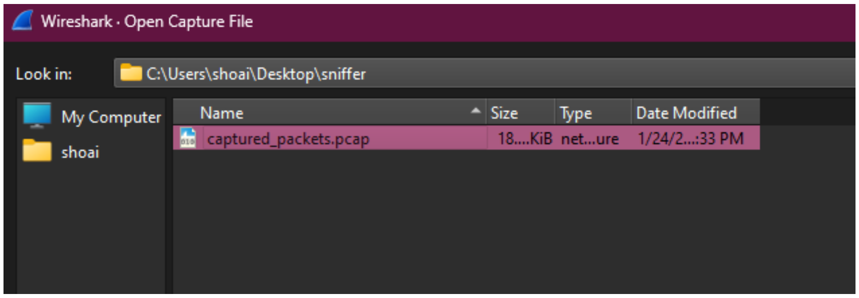
● Reboot now

○ I want to manually reboot later



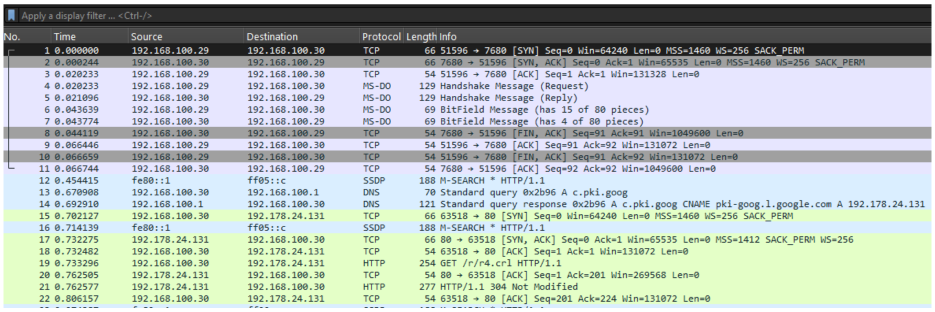
WiFi 2
Adapter for loopback traffic capture
Local Area Connection* 10



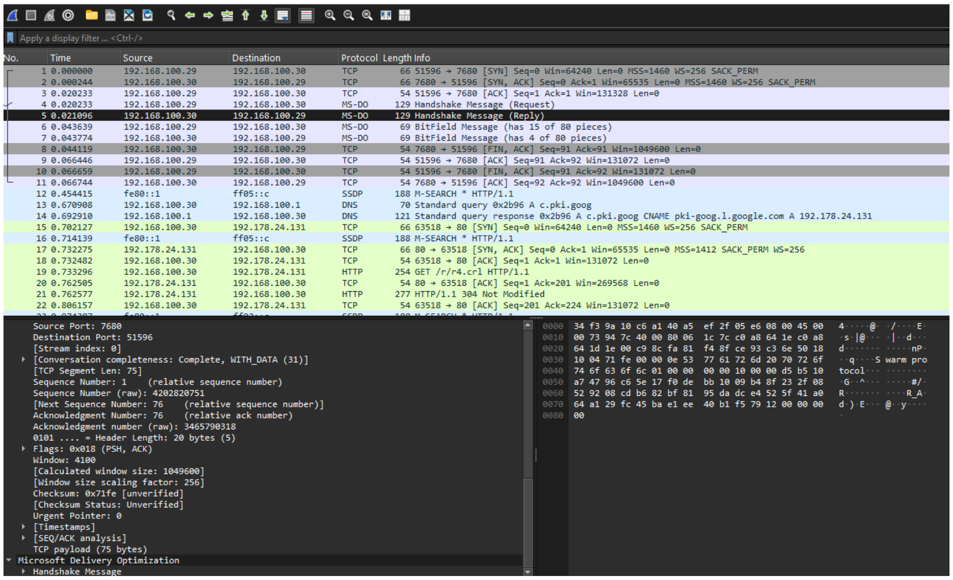


Look in: C:\Users\shoai\Desktop\sniffer

Name	Size	Type	Date Modified
captured_packets.pcap	18....KiB	net...ure	1/24/2...:33 PM



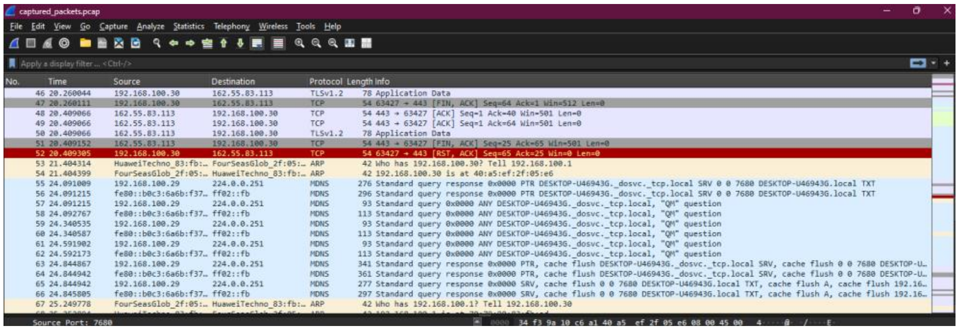
```
No.   Time       Source           Destination      Protocol Length Info
    1 0.000000   192.168.100.29   192.168.100.30   TCP        66 51596 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
    2 0.000244   192.168.100.30   192.168.100.29   TCP        66 7680 → 51596 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
    3 0.020233   192.168.100.29   192.168.100.30   TCP        54 51596 → 7680 [ACK] Seq=1 Ack=1 Win=131328 Len=0
    4 0.020233   192.168.100.29   192.168.100.30   MS-DO     129 Handshake Message (Request)
    5 0.021096   192.168.100.30   192.168.100.29   MS-DO     129 Handshake Message (Reply)
    6 0.043639   192.168.100.29   192.168.100.30   MS-DO      69 BitField Message (has 15 of 80 pieces)
    7 0.043774   192.168.100.30   192.168.100.29   MS-DO      69 BitField Message (has 4 of 80 pieces)
    8 0.044119   192.168.100.30   192.168.100.29   TCP        54 7680 → 51596 [FIN, ACK] Seq=91 Ack=91 Win=1049600 Len=0
    9 0.066446   192.168.100.29   192.168.100.30   TCP        54 51596 → 7680 [ACK] Seq=91 Ack=92 Win=131072 Len=0
   10 0.066659   192.168.100.29   192.168.100.30   TCP        54 51596 → 7680 [FIN, ACK] Seq=91 Ack=92 Win=131072 Len=0
   11 0.066744   192.168.100.30   192.168.100.29   TCP        54 7680 → 51596 [ACK] Seq=92 Ack=92 Win=1049600 Len=0
   12 0.454415   fe80::1          ff05::c          SSDP      188 M-SEARCH * HTTP/1.1
   13 0.670908   192.168.100.30   192.168.100.1    DNS        70 Standard query 0x2b96 A c.pki.goog
   14 0.692910   192.168.100.1    192.168.100.30   DNS       121 Standard query response 0x2b96 A c.pki.goog CNAME pki-goog.l.google.com A 192.178.24.131
   15 0.702127   192.168.100.30   192.178.24.131   TCP        66 63518 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
   16 0.714139   fe80::1          ff05::c          SSDP      188 M-SEARCH * HTTP/1.1
   17 0.732275   192.178.24.131   192.168.100.30   TCP        66 80 → 63518 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
   18 0.732482   192.168.100.30   192.178.24.131   TCP        54 63518 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
   19 0.733296   192.168.100.30   192.178.24.131   HTTP      254 GET /r/r4.crl HTTP/1.1
   20 0.762505   192.178.24.131   192.168.100.30   TCP        54 80 → 63518 [ACK] Seq=1 Ack=201 Win=269568 Len=0
   21 0.762577   192.178.24.131   192.168.100.30   HTTP      277 HTTP/1.1 304 Not Modified
   22 0.806157   192.168.100.30   192.178.24.131   TCP        54 63518 → 80 [ACK] Seq=201 Ack=224 Win=131072 Len=0
```



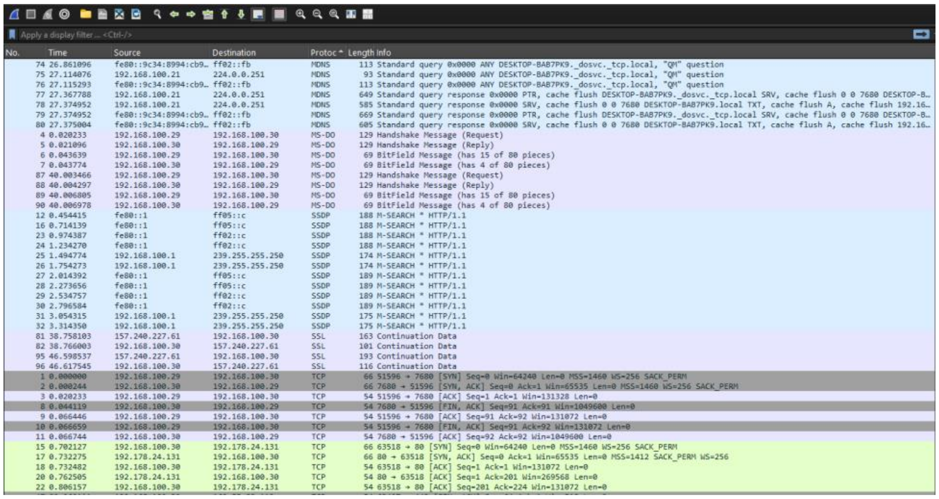
```
No.   Time       Source           Destination      Protocol Length Info
    1 0.000000   192.168.100.29   192.168.100.30   TCP        66 51596 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
    2 0.000244   192.168.100.30   192.168.100.29   TCP        66 7680 → 51596 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
    3 0.020233   192.168.100.29   192.168.100.30   TCP        54 51596 → 7680 [ACK] Seq=1 Ack=1 Win=131328 Len=0
    4 0.020233   192.168.100.29   192.168.100.30   MS-DO     129 Handshake Message (Request)
    5 0.021096   192.168.100.30   192.168.100.29   MS-DO     129 Handshake Message (Reply)
    6 0.043639   192.168.100.29   192.168.100.30   MS-DO      69 BitField Message (has 15 of 80 pieces)
    7 0.043774   192.168.100.30   192.168.100.29   MS-DO      69 BitField Message (has 4 of 80 pieces)
    8 0.044119   192.168.100.30   192.168.100.29   TCP        54 7680 → 51596 [FIN, ACK] Seq=91 Ack=91 Win=1049600 Len=0
    9 0.066446   192.168.100.29   192.168.100.30   TCP        54 51596 → 7680 [ACK] Seq=91 Ack=92 Win=131072 Len=0
   10 0.066659   192.168.100.29   192.168.100.30   TCP        54 51596 → 7680 [FIN, ACK] Seq=91 Ack=92 Win=131072 Len=0
   11 0.066744   192.168.100.30   192.168.100.29   TCP        54 7680 → 51596 [ACK] Seq=92 Ack=92 Win=1049600 Len=0
   12 0.454415   fe80::1          ff05::c          SSDP      188 M-SEARCH * HTTP/1.1
   13 0.670908   192.168.100.30   192.168.100.1    DNS        70 Standard query 0x2b96 A c.pki.goog
   14 0.692910   192.168.100.1    192.168.100.30   DNS       121 Standard query response 0x2b96 A c.pki.goog CNAME pki-goog.l.google.com A 192.178.24.131
   15 0.702127   192.168.100.30   192.178.24.131   TCP        66 63518 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
   16 0.714139   fe80::1          ff05::c          SSDP      188 M-SEARCH * HTTP/1.1
   17 0.732275   192.178.24.131   192.168.100.30   TCP        66 80 → 63518 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
   18 0.732482   192.168.100.30   192.178.24.131   TCP        54 63518 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
   19 0.733296   192.168.100.30   192.178.24.131   HTTP      254 GET /r/r4.crl HTTP/1.1
   20 0.762505   192.178.24.131   192.168.100.30   TCP        54 80 → 63518 [ACK] Seq=1 Ack=201 Win=269568 Len=0
   21 0.762577   192.178.24.131   192.168.100.30   HTTP      277 HTTP/1.1 304 Not Modified
   22 0.806157   192.168.100.30   192.178.24.131   TCP        54 63518 → 80 [ACK] Seq=201 Ack=224 Win=131072 Len=0
```

```
    Source Port: 7680
    Destination Port: 51596
    [Stream index: 0]
  ▸ [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 75]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 4202820751
    [Next Sequence Number: 76    (relative sequence number)]
    Acknowledgment Number: 76    (relative ack number)
    Acknowledgment number (raw): 3465790318
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x018 (PSH, ACK)
    Window: 4100
    [Calculated window size: 1049600]
    [Window size scaling factor: 256]
    Checksum: 0x71fe [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▸ [Timestamps]
  ▸ [SEQ/ACK analysis]
    TCP payload (75 bytes)
▾ Microsoft Delivery Optimization
  ▸ Handshake Message
```

```
0000  34 f3 9a 10 c6 a1 40 a5  ef 2f 05 e6 08 00 45 00   4·····@· ·/····E·
0010  00 73 94 7c 40 00 80 06  1c 7c c0 a8 64 1e c0 a8   ·s·|@··· ·|··d···
0020  64 1d 1e 00 c9 8c fa 81  f4 8f ce 93 c3 6e 50 18   d······· ·····nP·
0030  10 04 71 fe 00 00 0e 53  77 61 72 6d 20 70 72 6f   ··q····S warm pro
0040  74 6f 63 6f 6c 01 00 00  00 00 10 00 00 d5 b5 10   tocol··· ········
0050  a7 47 96 c6 5e 17 f0 de  bb 10 09 b4 8f 23 2f 08   ·G··^··· ·····#/·
0060  52 92 08 cd b6 82 bf 81  95 da dc e4 52 5f 41 a0   R······· ····R_A·
0070  64 a1 29 fc 45 ba e1 ee  40 b1 f5 79 12 00 00 00   d·)·E··· @··y····
0080  00                                                 ·
```


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

```
No.   Time      Source                Destination           Protocol Length Info
   46 20.260044 192.168.100.30        162.55.83.113         TLSv1.2    78 Application Data
   47 20.260111 192.168.100.30        162.55.83.113         TCP        54 63427 → 443 [FIN, ACK] Seq=64 Ack=1 Win=512 Len=0
   48 20.409066 162.55.83.113         192.168.100.30        TCP        54 443 → 63427 [ACK] Seq=1 Ack=40 Win=501 Len=0
   49 20.409066 162.55.83.113         192.168.100.30        TCP        54 443 → 63427 [ACK] Seq=1 Ack=64 Win=501 Len=0
   50 20.409066 162.55.83.113         192.168.100.30        TLSv1.2    78 Application Data
   51 20.409152 162.55.83.113         192.168.100.30        TCP        54 443 → 63427 [FIN, ACK] Seq=25 Ack=65 Win=501 Len=0
   52 20.409305 192.168.100.30        162.55.83.113         TCP        54 63427 → 443 [RST, ACK] Seq=65 Ack=25 Win=0 Len=0
   53 21.404314 HuaweiTechno_83:fb:…  FourSeasGlob_2f:05:…  ARP        42 Who has 192.168.100.30? Tell 192.168.100.1
   54 21.404399 FourSeasGlob_2f:05:…  HuaweiTechno_83:fb:…  ARP        42 192.168.100.30 is at 40:a5:ef:2f:05:e6
   55 24.091009 192.168.100.29        224.0.0.251           MDNS      276 Standard query response 0x0000 PTR DESKTOP-U46943G._dosvc._tcp.local SRV 0 0 7680 DESKTOP-U46943G.local TXT
   56 24.091215 fe80::b0c3:6a6b:f37…  ff02::fb              MDNS      296 Standard query response 0x0000 PTR DESKTOP-U46943G._dosvc._tcp.local SRV 0 0 7680 DESKTOP-U46943G.local TXT
   57 24.091215 192.168.100.29        224.0.0.251           MDNS       93 Standard query 0x0000 ANY DESKTOP-U46943G._dosvc._tcp.local, "QM" question
   58 24.092767 fe80::b0c3:6a6b:f37…  ff02::fb              MDNS      113 Standard query 0x0000 ANY DESKTOP-U46943G._dosvc._tcp.local, "QM" question
   59 24.340535 192.168.100.29        224.0.0.251           MDNS       93 Standard query 0x0000 ANY DESKTOP-U46943G._dosvc._tcp.local, "QM" question
   60 24.340587 fe80::b0c3:6a6b:f37…  ff02::fb              MDNS      113 Standard query 0x0000 ANY DESKTOP-U46943G._dosvc._tcp.local, "QM" question
   61 24.591902 192.168.100.29        224.0.0.251           MDNS       93 Standard query 0x0000 ANY DESKTOP-U46943G._dosvc._tcp.local, "QM" question
   62 24.592173 fe80::b0c3:6a6b:f37…  ff02::fb              MDNS      113 Standard query 0x0000 ANY DESKTOP-U46943G._dosvc._tcp.local, "QM" question
   63 24.844867 192.168.100.29        224.0.0.251           MDNS      341 Standard query response 0x0000 PTR, cache flush DESKTOP-U46943G._dosvc._tcp.local SRV, cache flush 0 0 7680 DESKTOP-U…
   64 24.844942 fe80::b0c3:6a6b:f37…  ff02::fb              MDNS      361 Standard query response 0x0000 PTR, cache flush DESKTOP-U46943G._dosvc._tcp.local SRV, cache flush 0 0 7680 DESKTOP-U…
   65 24.844942 192.168.100.29        224.0.0.251           MDNS      277 Standard query response 0x0000 SRV, cache flush 0 0 7680 DESKTOP-U46943G.local TXT, cache flush A, cache flush 192.16…
   66 24.845805 fe80::b0c3:6a6b:f37…  ff02::fb              MDNS      297 Standard query response 0x0000 SRV, cache flush 0 0 7680 DESKTOP-U46943G.local TXT, cache flush A, cache flush 192.16…
   67 25.249778 FourSeasGlob_2f:05:…  HuaweiTechno_83:fb:…  ARP        42 Who has 192.168.100.1? Tell 192.168.100.30
      Source Port: 7680                                   0000  34 f3 9a 10 c6 a1 40 a5  ef 2f 05 e6 08 00 45 00   4·····@· ·/····E·
```



```
No.   Time      Source                Destination      Protoc ^ Length Info
   74 26.861096 fe80::9c34:8994:cb9…  ff02::fb         MDNS       113 Standard query 0x0000 ANY DESKTOP-BAB7PK9._dosvc._tcp.local, "QM" question
   75 27.114076 192.168.100.21        224.0.0.251      MDNS        93 Standard query 0x0000 ANY DESKTOP-BAB7PK9._dosvc._tcp.local, "QM" question
   76 27.115293 fe80::9c34:8994:cb9…  ff02::fb         MDNS       113 Standard query 0x0000 ANY DESKTOP-BAB7PK9._dosvc._tcp.local, "QM" question
   77 27.367788 192.168.100.21        224.0.0.251      MDNS       649 Standard query response 0x0000 PTR, cache flush DESKTOP-BAB7PK9._dosvc._tcp.local SRV, cache flush 0 0 7680 DESKTOP-B…
   78 27.374952 192.168.100.21        224.0.0.251      MDNS       585 Standard query response 0x0000 SRV, cache flush 0 0 7680 DESKTOP-BAB7PK9.local TXT, cache flush A, cache flush 192.16…
   79 27.374952 fe80::9c34:8994:cb9…  ff02::fb         MDNS       669 Standard query response 0x0000 PTR, cache flush DESKTOP-BAB7PK9._dosvc._tcp.local SRV, cache flush 0 0 7680 DESKTOP-B…
   80 27.375004 fe80::9c34:8994:cb9…  ff02::fb         MDNS       605 Standard query response 0x0000 SRV, cache flush 0 0 7680 DESKTOP-BAB7PK9.local TXT, cache flush A, cache flush 192.16…
    4 0.020233  192.168.100.29        192.168.100.30   MS-DO      129 Handshake Message (Request)
    5 0.021096  192.168.100.30        192.168.100.29   MS-DO      129 Handshake Message (Reply)
    6 0.043639  192.168.100.29        192.168.100.30   MS-DO       69 BitField Message (has 15 of 80 pieces)
    7 0.043774  192.168.100.30        192.168.100.29   MS-DO       69 BitField Message (has 4 of 80 pieces)
   87 40.003466 192.168.100.29        192.168.100.30   MS-DO      129 Handshake Message (Request)
   88 40.004297 192.168.100.30        192.168.100.29   MS-DO      129 Handshake Message (Reply)
   89 40.006805 192.168.100.29        192.168.100.30   MS-DO       69 BitField Message (has 15 of 80 pieces)
   90 40.006978 192.168.100.30        192.168.100.29   MS-DO       69 BitField Message (has 4 of 80 pieces)
   12 0.454415  fe80::1               ff05::c          SSDP       188 M-SEARCH * HTTP/1.1
   16 0.714139  fe80::1               ff05::c          SSDP       188 M-SEARCH * HTTP/1.1
   23 0.974387  fe80::1               ff02::c          SSDP       188 M-SEARCH * HTTP/1.1
   24 1.234270  fe80::1               ff02::c          SSDP       188 M-SEARCH * HTTP/1.1
   25 1.494774  192.168.100.1         239.255.255.250  SSDP       174 M-SEARCH * HTTP/1.1
   26 1.754273  192.168.100.1         239.255.255.250  SSDP       174 M-SEARCH * HTTP/1.1
   27 2.014392  fe80::1               ff05::c          SSDP       189 M-SEARCH * HTTP/1.1
   28 2.273656  fe80::1               ff05::c          SSDP       189 M-SEARCH * HTTP/1.1
   29 2.534757  fe80::1               ff02::c          SSDP       189 M-SEARCH * HTTP/1.1
   30 2.796584  fe80::1               ff02::c          SSDP       189 M-SEARCH * HTTP/1.1
   31 3.054315  192.168.100.1         239.255.255.250  SSDP       175 M-SEARCH * HTTP/1.1
   32 3.314350  192.168.100.1         239.255.255.250  SSDP       175 M-SEARCH * HTTP/1.1
   81 38.758103 157.240.227.61        192.168.100.30   SSL        163 Continuation Data
   82 38.766003 192.168.100.30        157.240.227.61   SSL        101 Continuation Data
   95 46.598537 157.240.227.61        192.168.100.30   SSL        193 Continuation Data
   96 46.617545 192.168.100.30        157.240.227.61   SSL        116 Continuation Data
    1 0.000000  192.168.100.29        192.168.100.30   TCP         66 51596 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
    2 0.000244  192.168.100.30        192.168.100.29   TCP         66 7680 → 51596 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
    3 0.020233  192.168.100.29        192.168.100.30   TCP         54 51596 → 7680 [ACK] Seq=1 Ack=1 Win=131328 Len=0
    8 0.044119  192.168.100.30        192.168.100.29   TCP         54 7680 → 51596 [FIN, ACK] Seq=91 Ack=91 Win=1049600 Len=0
    9 0.066446  192.168.100.29        192.168.100.30   TCP         54 51596 → 7680 [ACK] Seq=91 Ack=92 Win=131072 Len=0
   10 0.066659  192.168.100.29        192.168.100.30   TCP         54 51596 → 7680 [FIN, ACK] Seq=91 Ack=92 Win=131072 Len=0
   11 0.066744  192.168.100.30        192.168.100.29   TCP         54 7680 → 51596 [ACK] Seq=92 Ack=92 Win=1049600 Len=0
   15 0.702127  192.168.100.30        192.178.24.131   TCP         66 63518 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
   17 0.732275  192.178.24.131        192.168.100.30   TCP         66 80 → 63518 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
   18 0.732482  192.168.100.30        192.178.24.131   TCP         54 63518 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
   20 0.762505  192.178.24.131        192.168.100.30   TCP         54 80 → 63518 [ACK] Seq=1 Ack=201 Win=269568 Len=0
   22 0.806157  192.168.100.30        192.178.24.131   TCP         54 63518 → 80 [ACK] Seq=201 Ack=224 Win=131072 Len=0
```

8. Challenges and Solutions

8.1 Issues Encountered During Development

During development, several issues arose, including:

- **Accessing Network Interfaces**: Sometimes, the wrong interface was selected, leading to no packets being captured.
- **Permissions Issues**: On some systems, administrator privileges were needed to capture packets.

8.2 Solutions and Workarounds

- Ensured the correct network interface was selected by verifying it through ipconfig or ifconfig.
- Ran the script as an administrator on Windows and used sudo on Linux to resolve permission issues.

9. Conclusion

9.1 Key Findings

The project successfully demonstrated how to create a network sniffer in Python using Scapy. The sniffer captured packets and saved them to a PCAP file for further analysis. This tool is useful for network diagnostics, security assessments, and educational purposes.

9.2 Future Improvements and Enhancements

Future improvements could include:

- Adding more advanced filtering for specific types of packets.
- Implementing packet analysis features directly in the script, reducing reliance on external tools like Wireshark.
- Adding a graphical user interface (GUI) for easier interaction.

---

10. References

- Scapy Documentation
- Wireshark Documentation
- Python 3.x Official Documentation: https://docs.python.org/3/