NAME: Adeyemi Akande

REG NUM: 2024/INT/3800

OSI Model Layers

1. **Physical Layer (Layer 1):**
    - **Function:** Responsible for transmitting raw binary data (bits) over a physical medium. Deals with hardware components such as cables, switches, and connectors.
    - **Examples:** Ethernet cables, fiber optics, hubs, repeaters.
    - **TCP/IP Match:** Link Layer.
2. **Data Link Layer (Layer 2):**
    - **Function:** Ensures reliable transmission of data frames between two nodes. Handles error detection, flow control, and MAC addressing.
    - **Examples:** Ethernet (802.3), Wi-Fi (802.11), PPP.
    - **TCP/IP Match:** Link Layer.
3. **Network Layer (Layer 3):**
    - **Function:** Manages logical addressing (IP addresses) and routes data packets between devices across different networks.
    - **Examples:** IP (IPv4, IPv6), ICMP, ARP.
    - **TCP/IP Match:** Internet Layer.
4. **Transport Layer (Layer 4):**
    - **Function:** Ensures reliable data transfer (error correction, flow control) between hosts. Offers end-to-end communication services.
    - **Examples:** TCP (reliable), UDP (unreliable).
    - **TCP/IP Match:** Transport Layer.
5. **Session Layer (Layer 5):**
    - **Function:** Establishes, manages, and terminates communication sessions between applications.
    - **Examples:** NetBIOS, RPC.
    - **TCP/IP Match:** Application Layer (combined in TCP/IP).
6. **Presentation Layer (Layer 6):**
    - **Function:** Handles data translation, encryption, decryption, and compression to make data understandable between different systems.
    - **Examples:** SSL/TLS, JPEG, PNG.
    - **TCP/IP Match:** Application Layer (combined in TCP/IP).
7. **Application Layer (Layer 7):**
    - **Function:** Provides network services to end users and applications, such as email, file transfer, and web browsing.
    - **Examples:** HTTP, FTP, SMTP, DNS.
    - **TCP/IP Match:** Application Layer.

OSI to TCP/IP Layer Mapping

OSI Layer	TCP/IP Layer	Function Match
Application (7)	Application	Provides end-user services like HTTP, DNS.
Presentation (6)	Application	Handles data formatting, encryption, compression.
Session (5)	Application	Manages sessions between communicating devices.
Transport (4)	Transport	Ensures reliable communication (TCP) or fast (UDP).
Network (3)	Internet	Handles routing and addressing of data packets.
Data Link (2)	Link	Deals with reliable node-to-node data transfer.
Physical (1)	Link	Transmits raw bits over physical hardware.

This structure simplifies networking concepts and highlights how the OSI model's theoretical framework maps to the practical implementation of the TCP/IP model.

Exercise 2

Exercise 3:

What Happens When You Ping the OWASP Application?

The ping command is used to test connectivity between your device and a remote application or server (in this case, the OWASP application) by sending ICMP (Internet Control Message Protocol) packets.

Process Breakdown:

1. **Initiating the Ping Command:**
    - You enter the command ping <OWASP application IP/hostname> in the terminal or command prompt.
    - For example, ping owasp.org.
2. **DNS Resolution (if hostname is used):**
    - If you use a domain name (e.g., owasp.org), the operating system performs a DNS query to resolve the domain name to its corresponding IP address.

    - **Layer Involved:** Application Layer (DNS protocol).
3. **ICMP Echo Request:**
    - The system generates an ICMP Echo Request packet containing the source IP address (your device) and the destination IP address (the OWASP application).
    - The packet is encapsulated and sent down the stack.
4. **Packet Transmission:**
    - **Network Layer (OSI Layer 3):** The ICMP Echo Request is routed to the destination IP using the Internet Protocol (IP). Routers forward the packet based on the destination address.
    - **Data Link Layer (OSI Layer 2):** The packet is encapsulated into frames and transmitted across the network hardware (e.g., Ethernet, Wi-Fi).
    - **Physical Layer (OSI Layer 1):** Bits are sent as electrical signals or radio waves over the physical medium.
5. **Reception by the OWASP Server:**
    - The OWASP server receives the ICMP Echo Request packet, processes it, and generates an ICMP Echo Reply packet.
6. **Echo Reply Transmission:**
    - The Echo Reply follows the reverse path back to your device, traversing the same OSI layers.
7. **Result Display:**
    - The ping command calculates the round-trip time (RTT) and displays statistics, such as the number of packets sent, received, and lost.

Which OSI Layer Does the Ping Command Operate In?

- **Primary Layer: Network Layer (Layer 3)**
    - The ping command uses ICMP, which operates at the Network Layer to send and receive packets.
- **Supporting Layers:**
    - **Application Layer (Layer 7):** If a domain name is used, DNS resolves it to an IP address.
    - **Data Link Layer (Layer 2):** Ensures the packet is transmitted over the local network.
    - **Physical Layer (Layer 1):** Handles the actual transmission of bits over the medium.

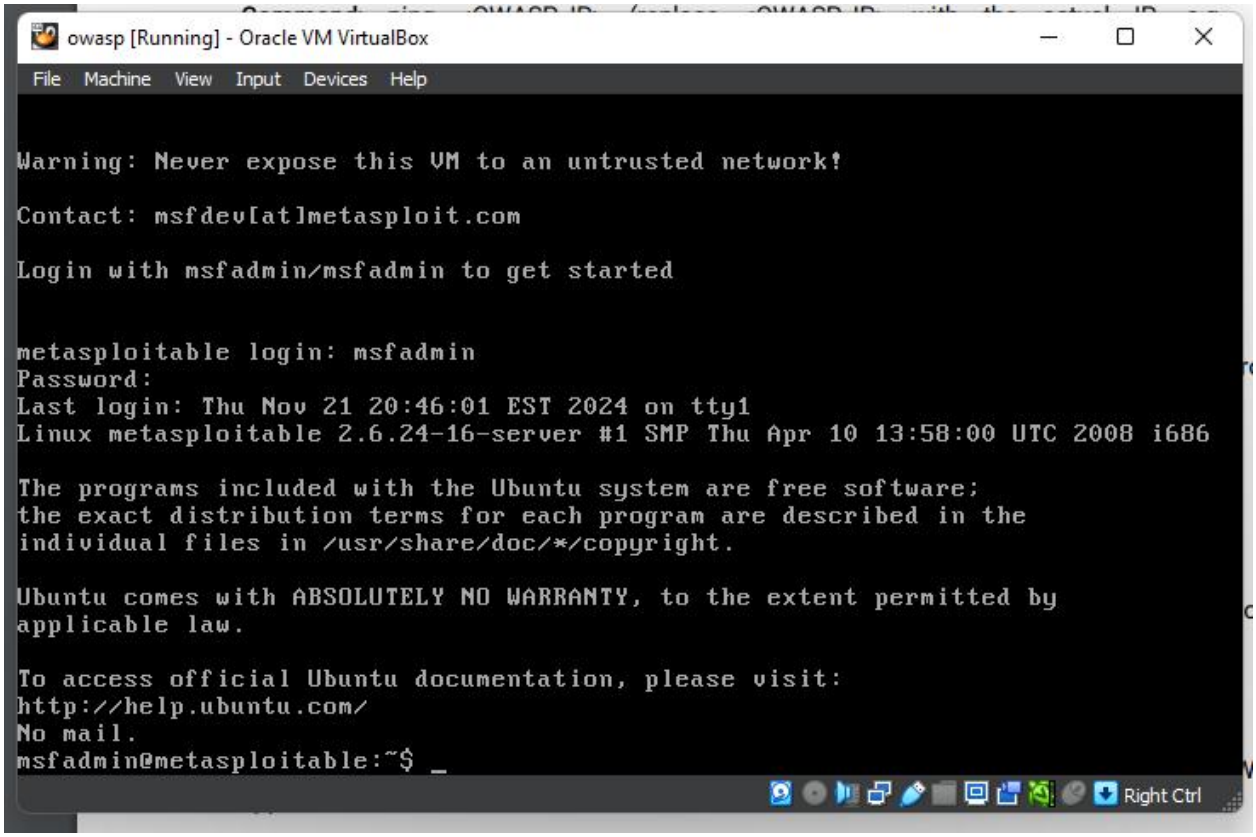
Explanation: Why Network Layer?

ICMP is specifically designed for network diagnostics and error reporting, which are functions of the Network Layer. It doesn't involve higher-layer protocols like TCP or UDP, as it's focused on delivering diagnostic messages, such as "Echo Request" and "Echo Reply."

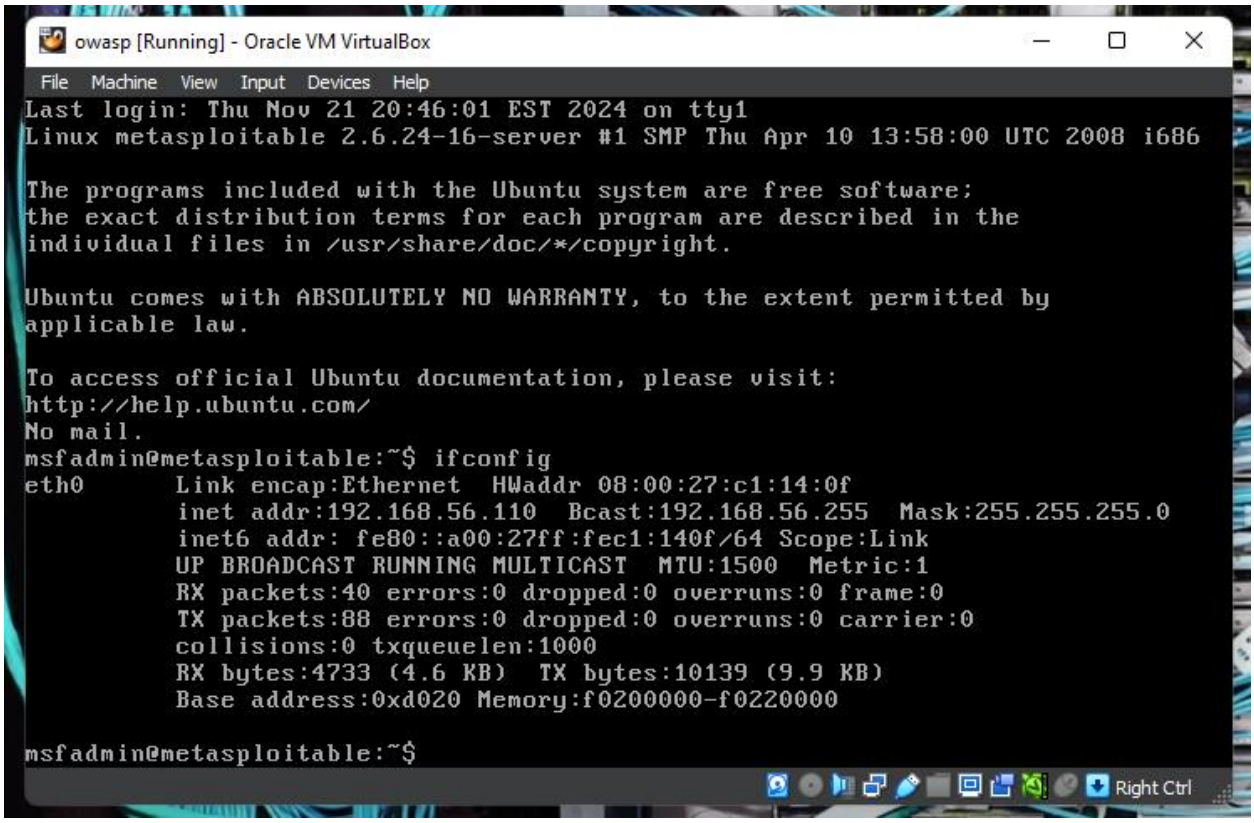
This makes the ping command a vital tool for checking basic connectivity and troubleshooting network issues.

Exercise 3: Tracing the Path to the OWASP Application

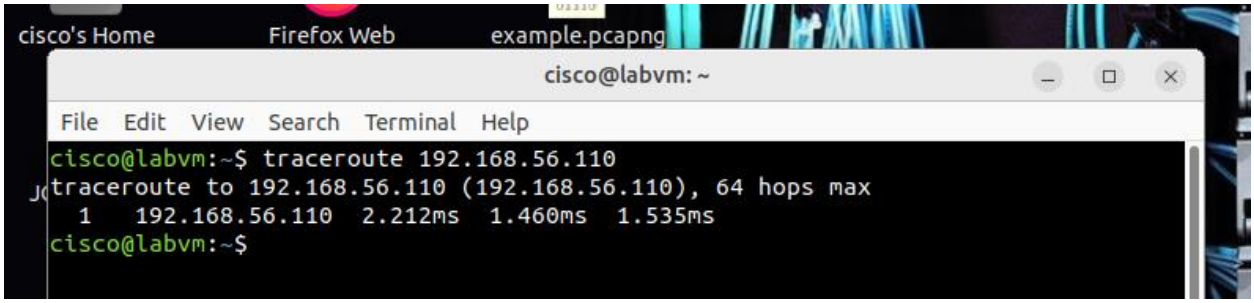
My broken webapp powered on



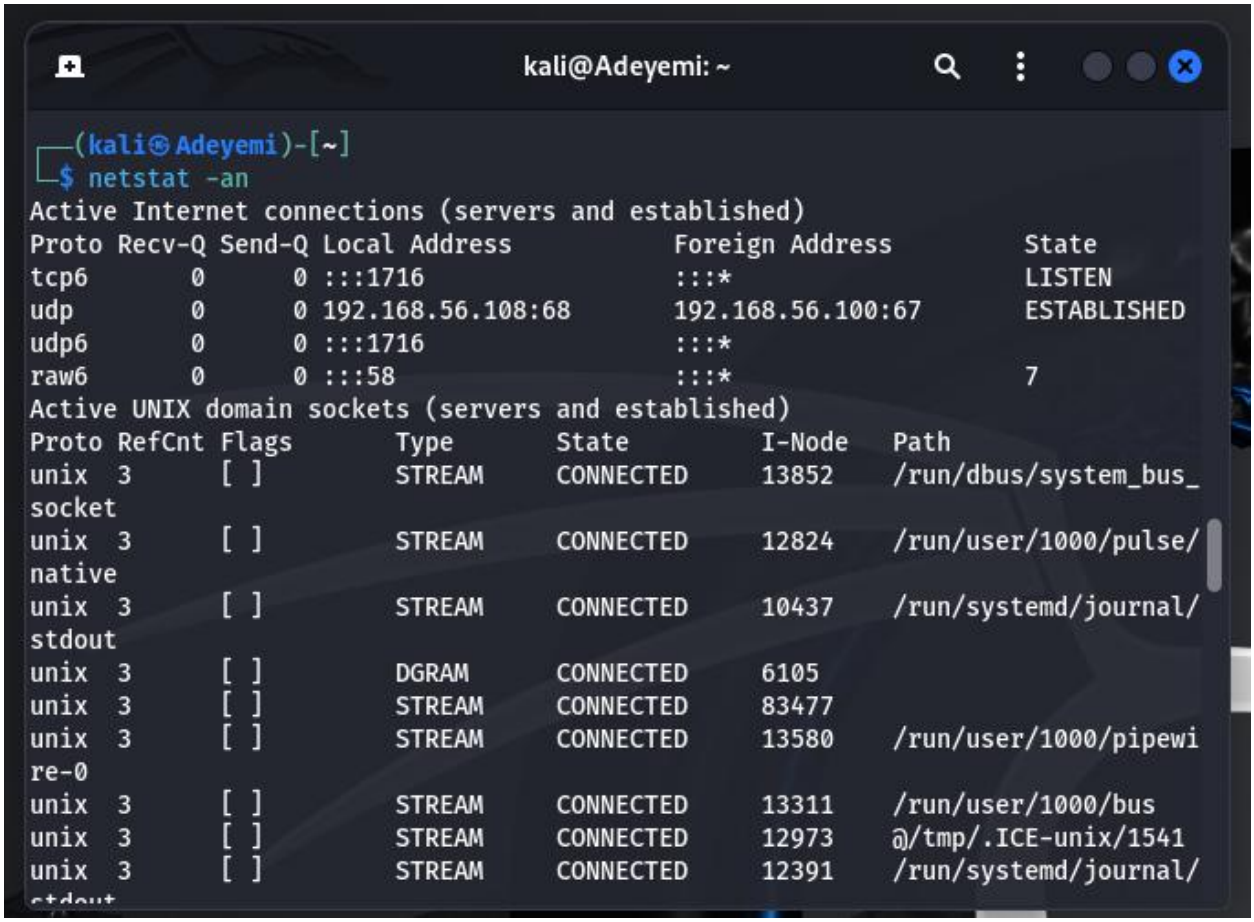
Accessing the ip address with ifconfig

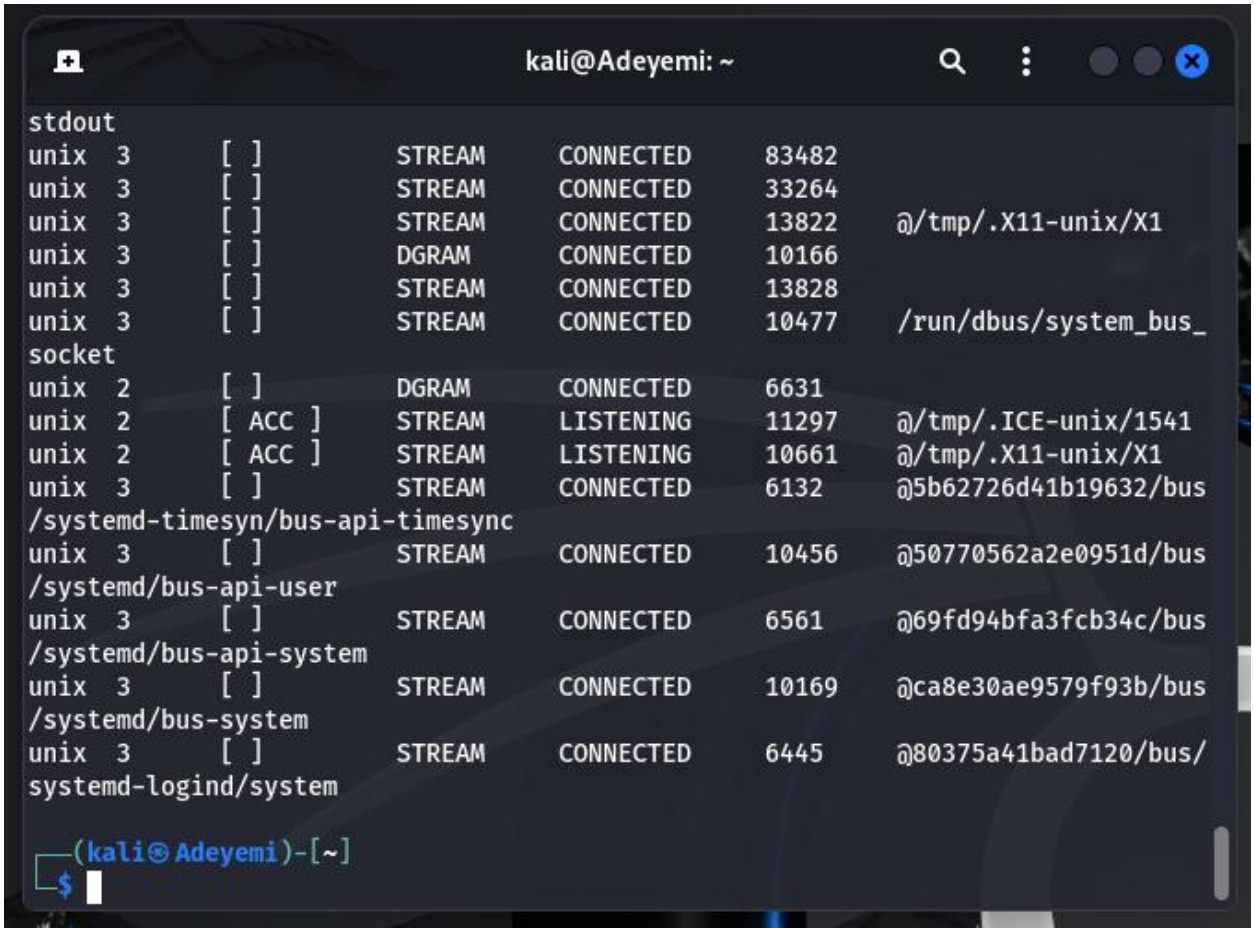


Traceroute from my Cisco vm lab

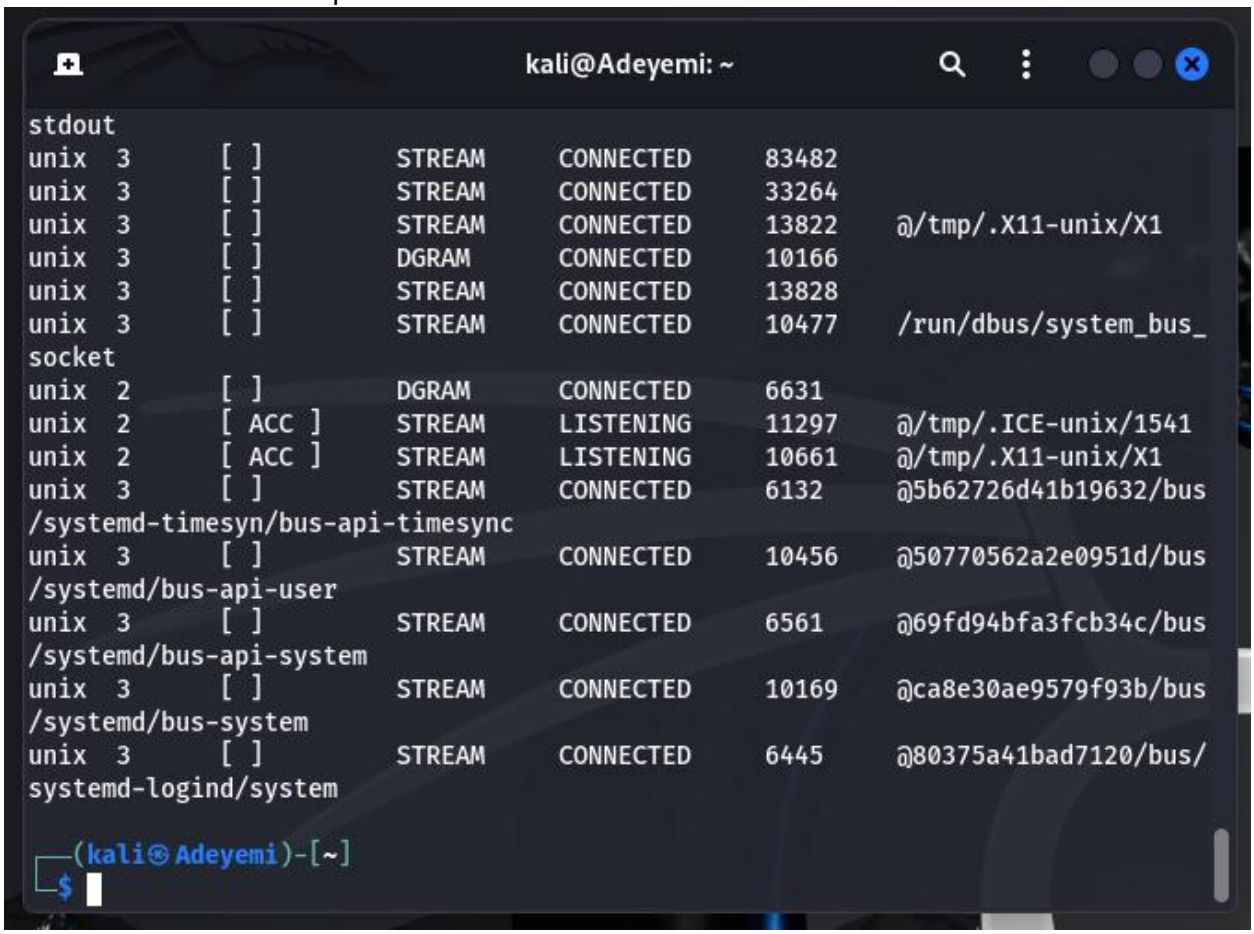


Exercise 4


kali@Adeyemi: ~
┌──(kali㉿Adeyemi)-[~]
└─$ netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp6 0 0 :::1716 :::* LISTEN
udp 0 0 192.168.56.108:68 192.168.56.100:67 ESTABLISHED
udp6 0 0 :::1716 :::*
raw6 0 0 :::58 :::* 7
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags Type State I-Node Path
unix 3 [ ] STREAM CONNECTED 13852 /run/dbus/system_bus_
socket
unix 3 [ ] STREAM CONNECTED 12824 /run/user/1000/pulse/
native
unix 3 [ ] STREAM CONNECTED 10437 /run/systemd/journal/
stdout
unix 3 [ ] DGRAM CONNECTED 6105
unix 3 [ ] STREAM CONNECTED 83477
unix 3 [ ] STREAM CONNECTED 13580 /run/user/1000/pipewi
re-0
unix 3 [ ] STREAM CONNECTED 13311 /run/user/1000/bus
unix 3 [ ] STREAM CONNECTED 12973 @/tmp/.ICE-unix/1541
unix 3 [ ] STREAM CONNECTED 12391 /run/systemd/journal/


kali@Adeyemi: ~
stdout
unix 3 [ ] STREAM CONNECTED 83482
unix 3 [ ] STREAM CONNECTED 33264
unix 3 [ ] STREAM CONNECTED 13822 @/tmp/.X11-unix/X1
unix 3 [ ] DGRAM CONNECTED 10166
unix 3 [ ] STREAM CONNECTED 13828
unix 3 [ ] STREAM CONNECTED 10477 /run/dbus/system_bus_
socket
unix 2 [ ] DGRAM CONNECTED 6631
unix 2 [ACC] STREAM LISTENING 11297 @/tmp/.ICE-unix/1541
unix 2 [ACC] STREAM LISTENING 10661 @/tmp/.X11-unix/X1
unix 3 [ ] STREAM CONNECTED 6132 @5b62726d41b19632/bus
/systemd-timesyn/bus-api-timesync
unix 3 [ ] STREAM CONNECTED 10456 @50770562a2e0951d/bus
/systemd/bus-api-user
unix 3 [ ] STREAM CONNECTED 6561 @69fd94bfa3fcb34c/bus
/systemd/bus-api-system
unix 3 [ ] STREAM CONNECTED 10169 @ca8e30ae9579f93b/bus
/systemd/bus-system
unix 3 [ ] STREAM CONNECTED 6445 @80375a41bad7120/bus/
systemd-logind/system
(kali㉿Adeyemi)-[~]
$

It identified a number of ip addresses it named one local



Exercise 5: TCP vs. UDP Task: Investigate the difference between TCP and UDP by scanning the OWASP BrokenWebApplication. Run a TCP scan using nmap. Command: nmap -sT

cisco@labvm:~$ nmap -sT 192.168.56.110

Starting Nmap 7.80 (https://nmap.org) at 2024-11-24 02:54 UTC

Nmap scan report for 192.168.56.110

Host is up (0.0021s latency).

Not shown: 978 closed ports

PORT STATE SERVICE

21/tcp open ftp

22/tcp open ssh

23/tcp open telnet

25/tcp open smtp

53/tcp open domain

80/tcp open http

111/tcp open rpcbind

139/tcp open netbios-ssn

445/tcp open microsoft-ds

512/tcp open exec

513/tcp open login

514/tcp open shell

1099/tcp open rmiregistry

1524/tcp open ingreslock

2121/tcp open ccproxy-ftp

3306/tcp open mysql

5432/tcp open postgresql

5900/tcp open vnc

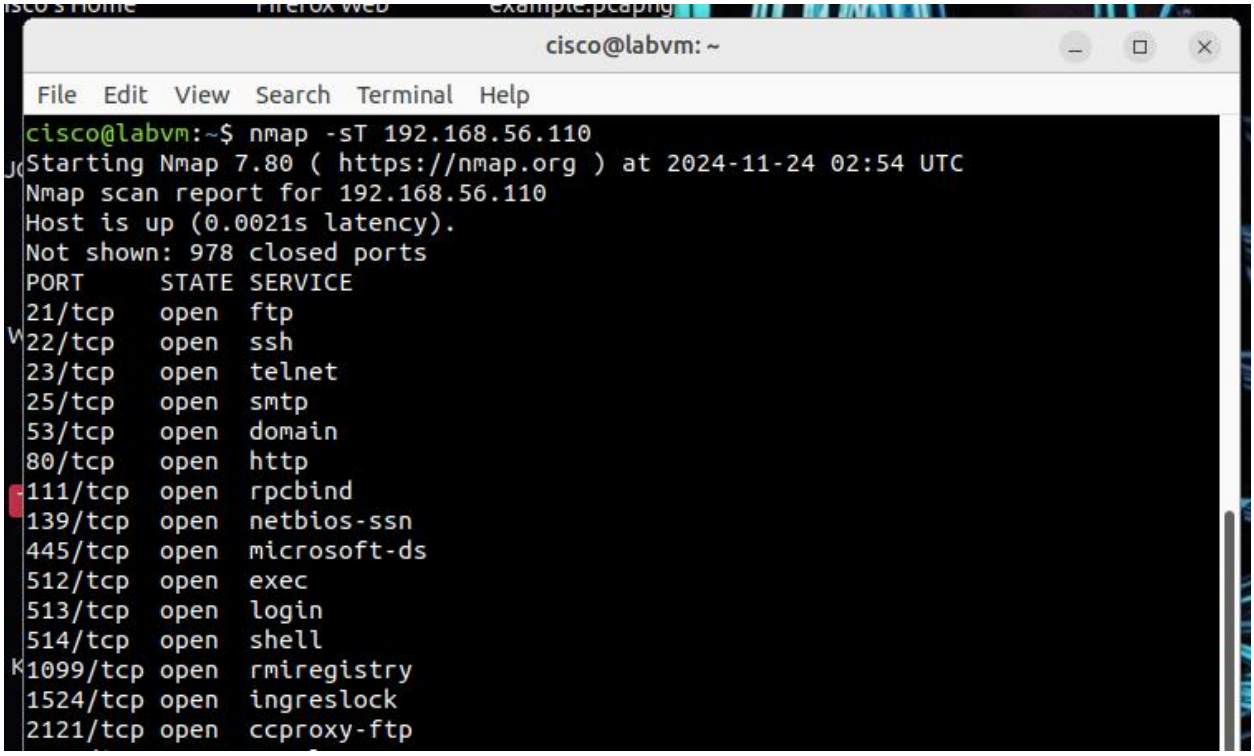
6000/tcp open X11

6667/tcp open irc

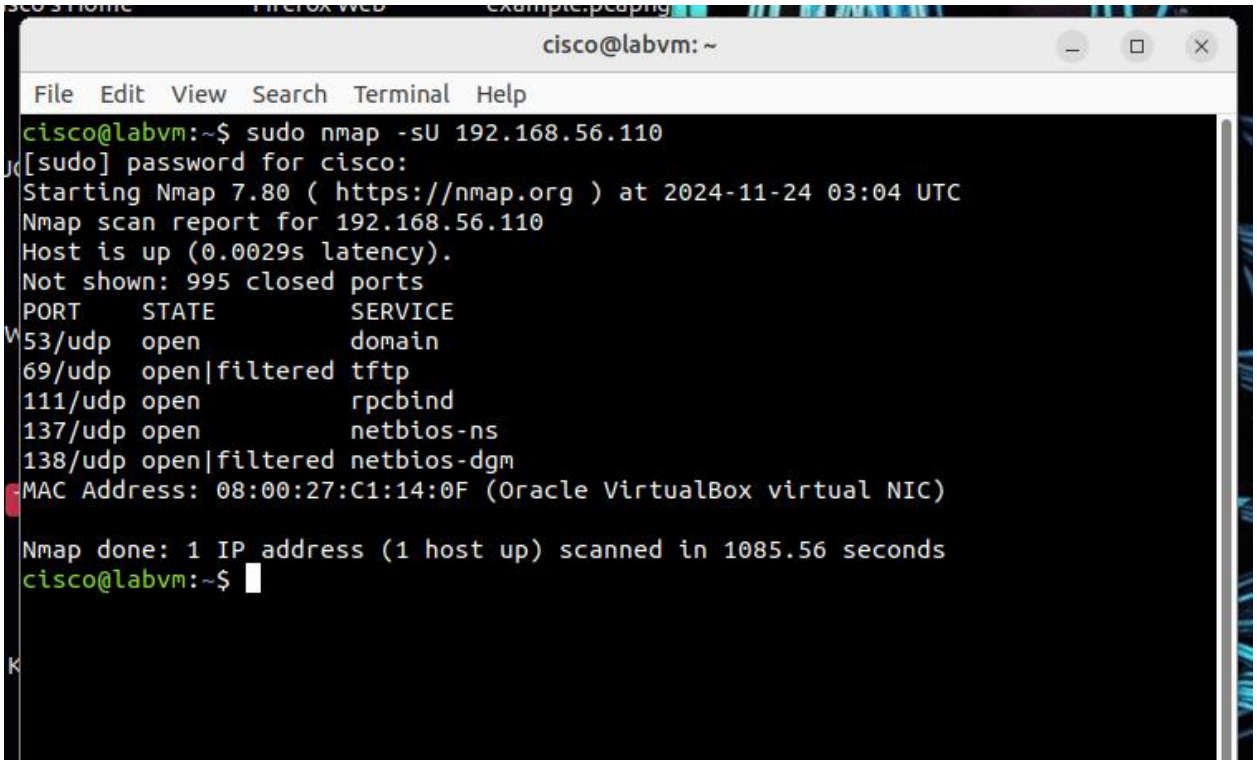
8009/tcp open ajp13

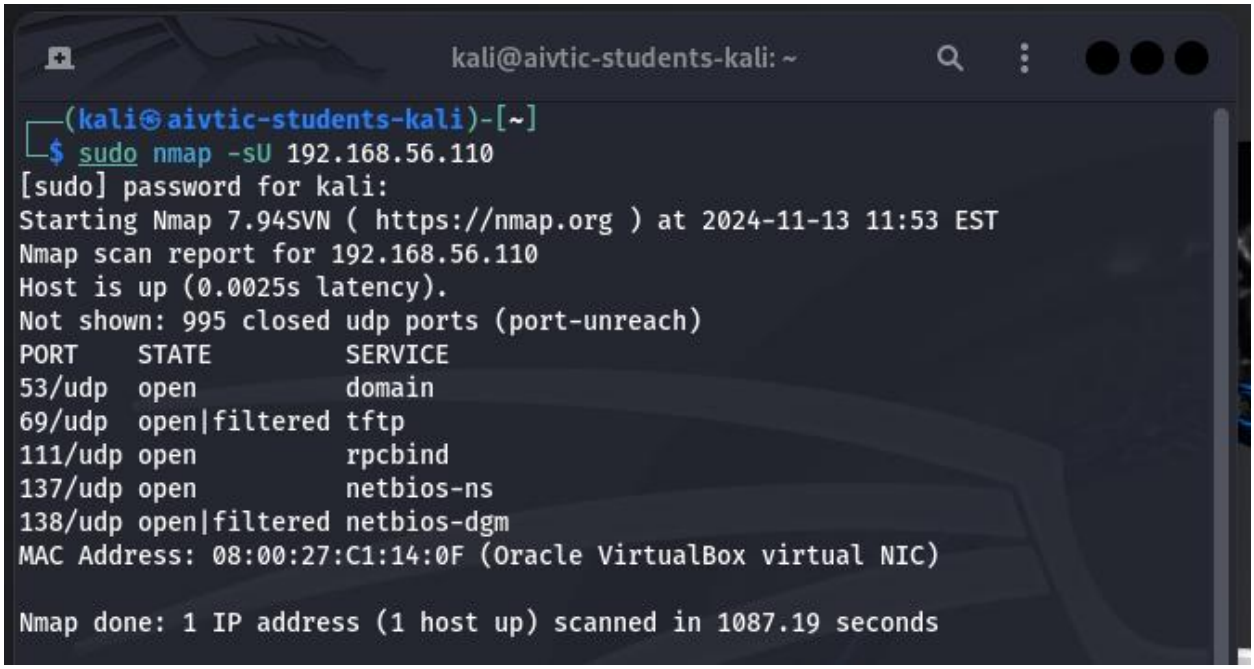
8180/tcp open unknown

Nmap done: 1 IP address (1 host up) scanned in 13.37 seconds



Run a UDP scan. Command: nmap -sU

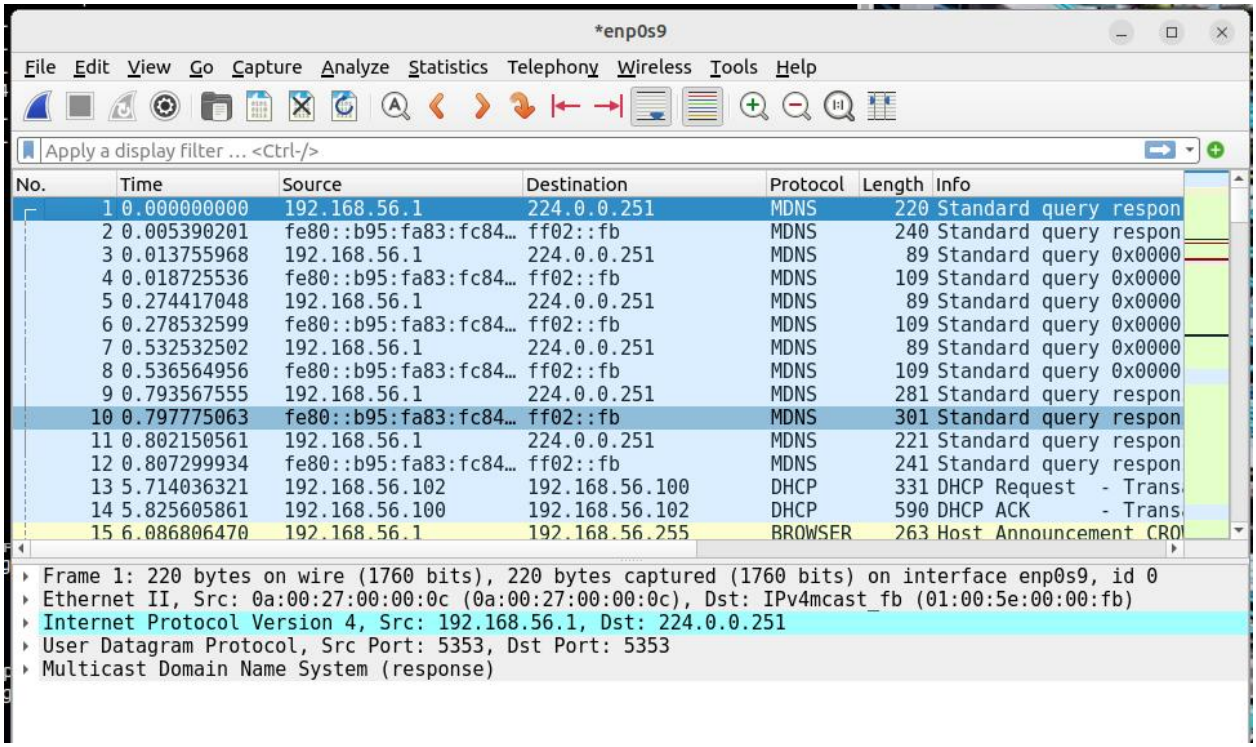




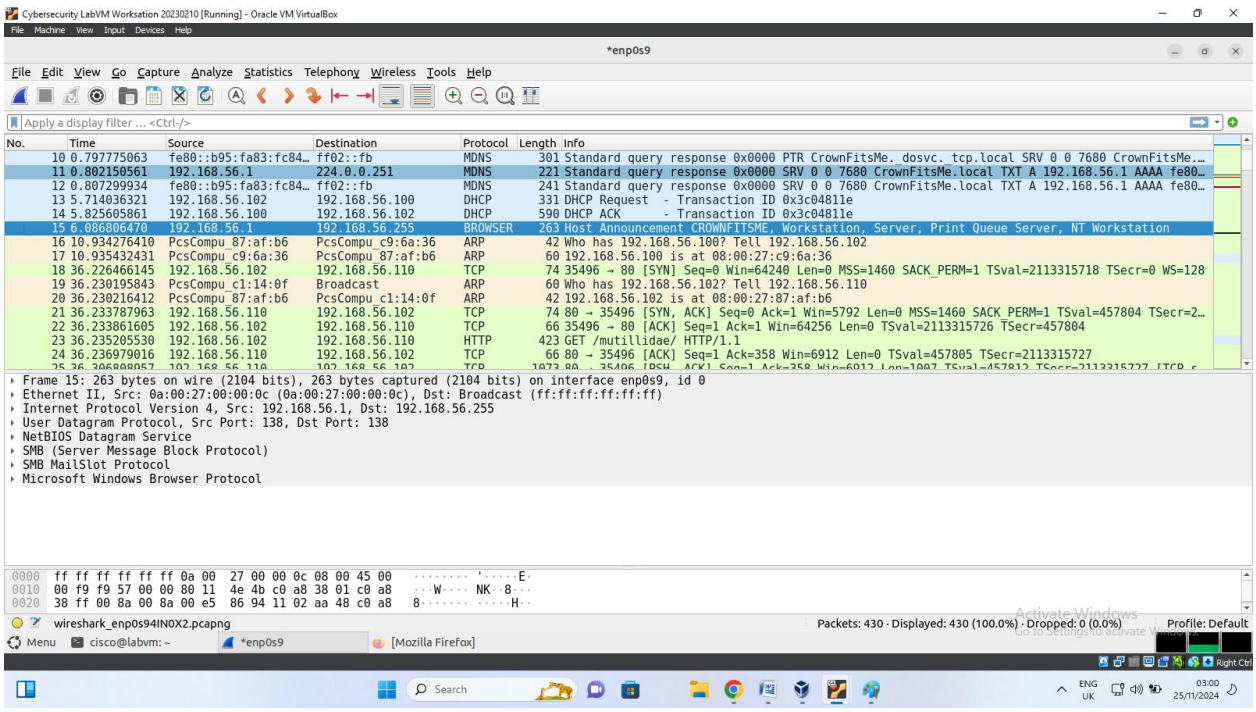
Exercise 6: Discovering MAC Addresses with ARP

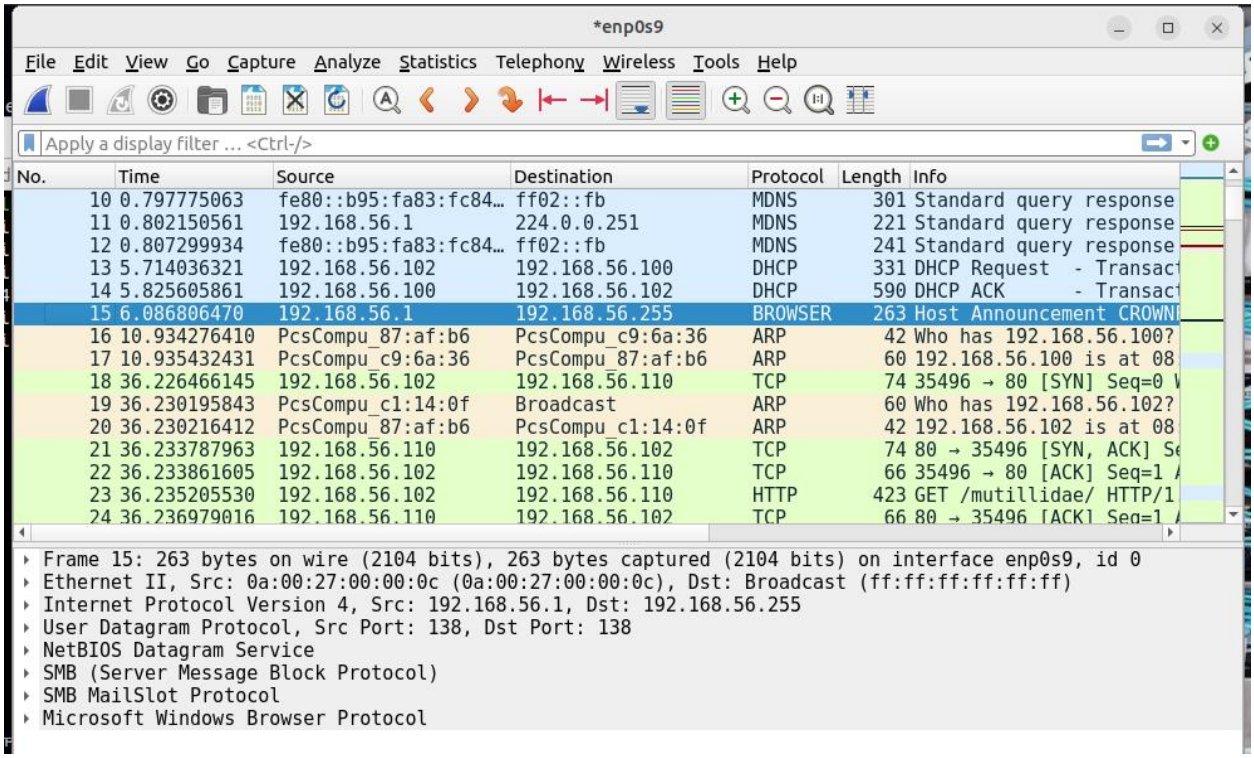
Exercise 7: Capturing Network Traffic with Wireshark

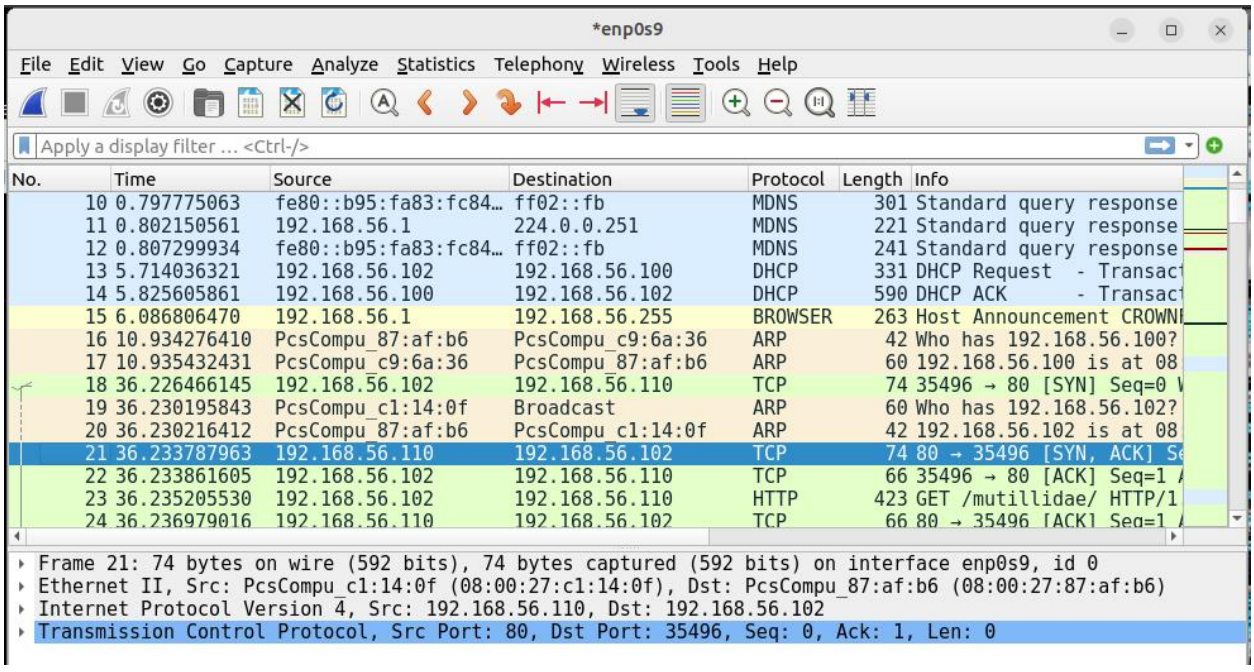
When I started the wireshark

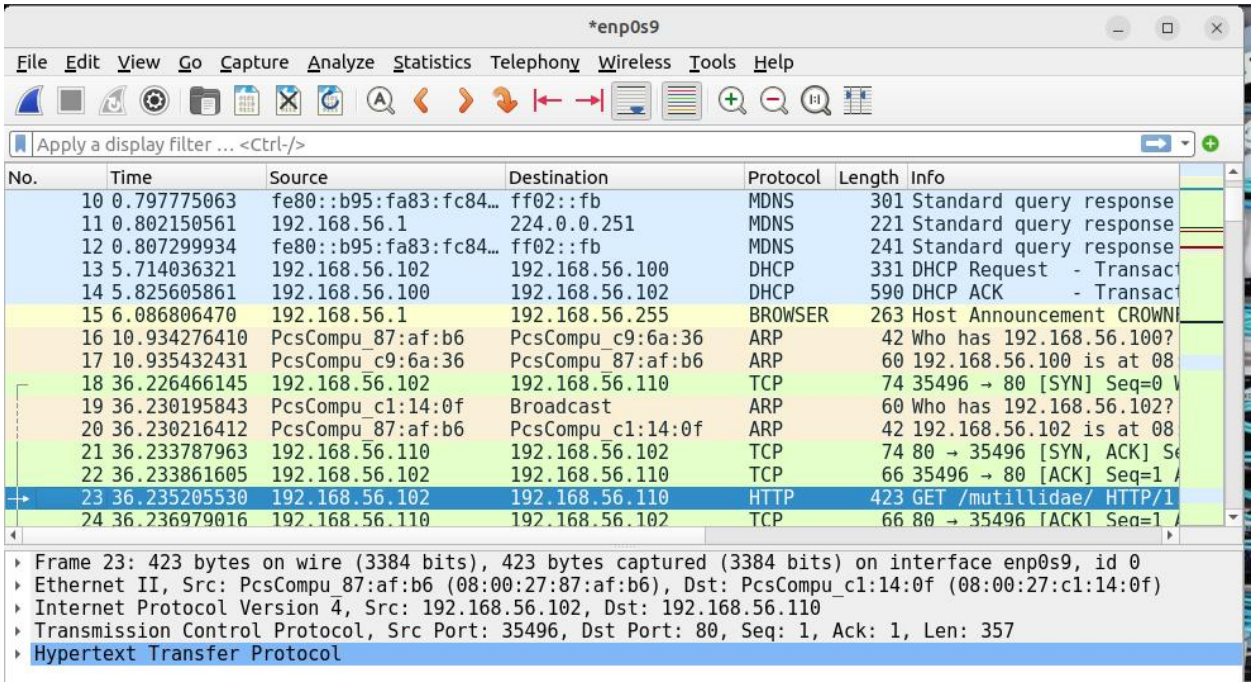


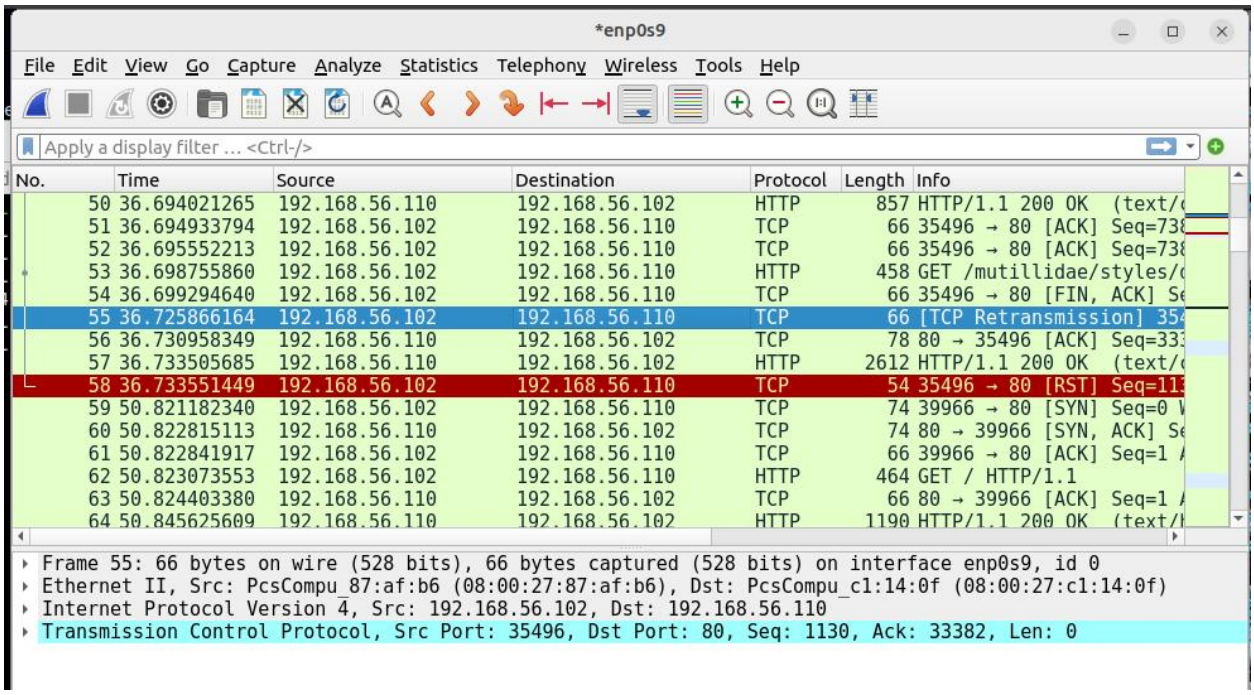
When I started the browser

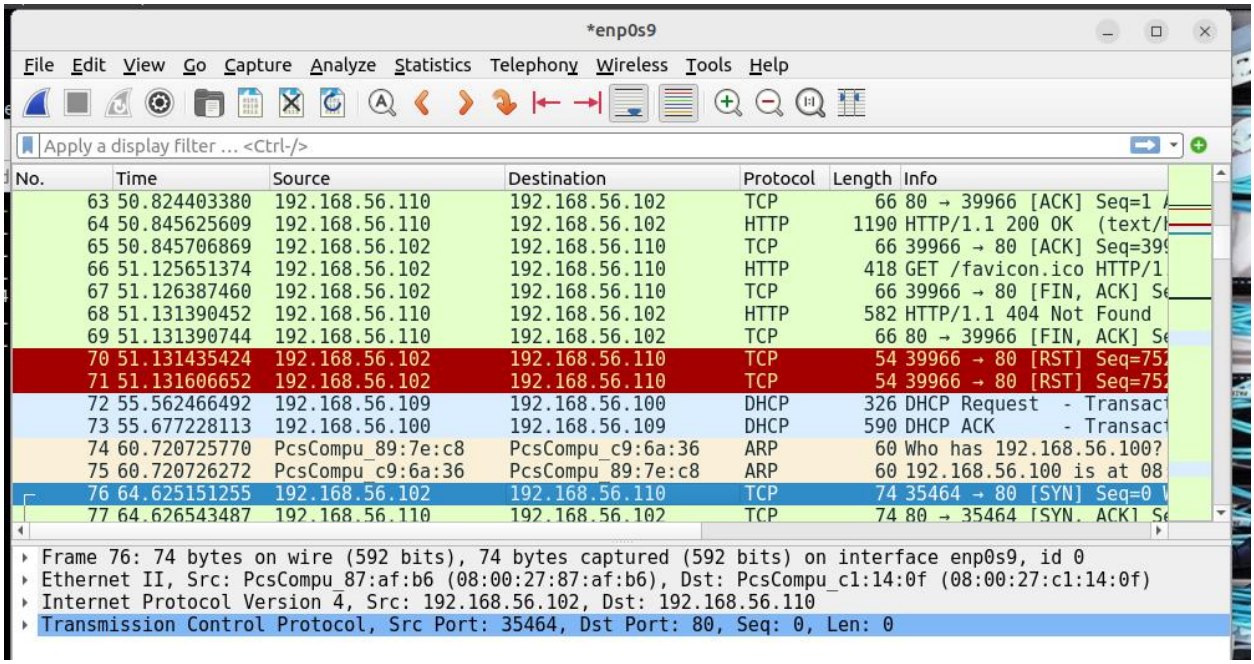


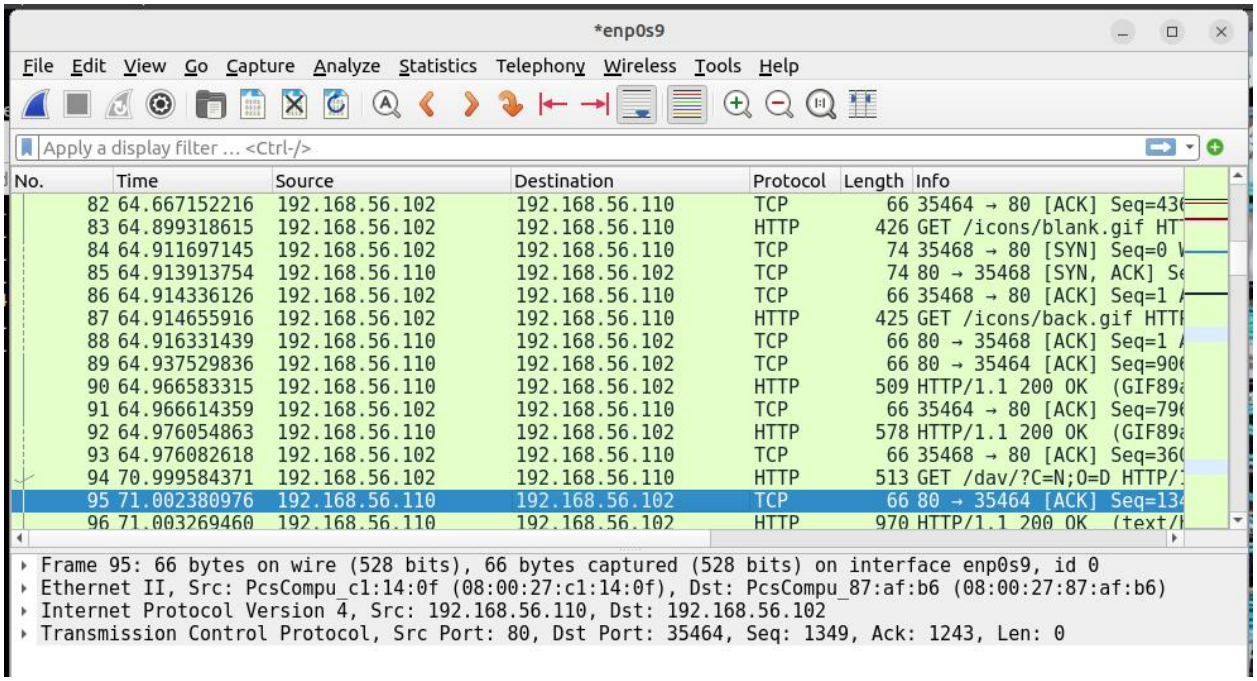

*enp0s9
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
No. Time Source Destination Protocol Length Info
10 0.797775063 fe80::b95:fa83:fc84… ff02::fb MDNS 301 Standard query response
11 0.802150561 192.168.56.1 224.0.0.251 MDNS 221 Standard query response
12 0.807299934 fe80::b95:fa83:fc84… ff02::fb MDNS 241 Standard query response
13 5.714036321 192.168.56.102 192.168.56.100 DHCP 331 DHCP Request - Transact
14 5.825605861 192.168.56.100 192.168.56.102 DHCP 590 DHCP ACK - Transact
15 6.086806470 192.168.56.1 192.168.56.255 BROWSER 263 Host Announcement CROWN
16 10.934276410 PcsCompu_87:af:b6 PcsCompu_c9:6a:36 ARP 42 Who has 192.168.56.100?
17 10.935432431 PcsCompu_c9:6a:36 PcsCompu_87:af:b6 ARP 60 192.168.56.100 is at 08
18 36.226466145 192.168.56.102 192.168.56.110 TCP 74 35496 → 80 [SYN] Seq=0
19 36.230195843 PcsCompu_c1:14:0f Broadcast ARP 60 Who has 192.168.56.102?
20 36.230216412 PcsCompu_87:af:b6 PcsCompu_c1:14:0f ARP 42 192.168.56.102 is at 08
21 36.233787963 192.168.56.110 192.168.56.102 TCP 74 80 → 35496 [SYN, ACK] Se
22 36.233861605 192.168.56.102 192.168.56.110 TCP 66 35496 → 80 [ACK] Seq=1
23 36.235205530 192.168.56.102 192.168.56.110 HTTP 423 GET /mutillidae/ HTTP/1
24 36.236979016 192.168.56.110 192.168.56.102 TCP 66 80 → 35496 [ACK] Seq=1
Frame 15: 263 bytes on wire (2104 bits), 263 bytes captured (2104 bits) on interface enp0s9, id 0
Ethernet II, Src: 0a:00:27:00:00:0c (0a:00:27:00:00:0c), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 192.168.56.1, Dst: 192.168.56.255
User Datagram Protocol, Src Port: 138, Dst Port: 138
NetBIOS Datagram Service
SMB (Server Message Block Protocol)
SMB MailSlot Protocol
Microsoft Windows Browser Protocol


*enp0s9
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
No. Time Source Destination Protocol Length Info
10 0.797775063 fe80::b95:fa83:fc84… ff02::fb MDNS 301 Standard query response
11 0.802150561 192.168.56.1 224.0.0.251 MDNS 221 Standard query response
12 0.807299934 fe80::b95:fa83:fc84… ff02::fb MDNS 241 Standard query response
13 5.714036321 192.168.56.102 192.168.56.100 DHCP 331 DHCP Request - Transact
14 5.825605861 192.168.56.100 192.168.56.102 DHCP 590 DHCP ACK - Transact
15 6.086806470 192.168.56.1 192.168.56.255 BROWSER 263 Host Announcement CROWN
16 10.934276410 PcsCompu_87:af:b6 PcsCompu_c9:6a:36 ARP 42 Who has 192.168.56.100?
17 10.935432431 PcsCompu_c9:6a:36 PcsCompu_87:af:b6 ARP 60 192.168.56.100 is at 08
18 36.226466145 192.168.56.102 192.168.56.110 TCP 74 35496 → 80 [SYN] Seq=0
19 36.230195843 PcsCompu_c1:14:0f Broadcast ARP 60 Who has 192.168.56.102?
20 36.230216412 PcsCompu_87:af:b6 PcsCompu_c1:14:0f ARP 42 192.168.56.102 is at 08
21 36.233787963 192.168.56.110 192.168.56.102 TCP 74 80 → 35496 [SYN, ACK] Se
22 36.233861605 192.168.56.102 192.168.56.110 TCP 66 35496 → 80 [ACK] Seq=1
23 36.235205530 192.168.56.102 192.168.56.110 HTTP 423 GET /mutillidae/ HTTP/1
24 36.236979016 192.168.56.110 192.168.56.102 TCP 66 80 → 35496 [ACK] Seq=1
Frame 21: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface enp0s9, id 0
Ethernet II, Src: PcsCompu_c1:14:0f (08:00:27:c1:14:0f), Dst: PcsCompu_87:af:b6 (08:00:27:87:af:b6)
Internet Protocol Version 4, Src: 192.168.56.110, Dst: 192.168.56.102
Transmission Control Protocol, Src Port: 80, Dst Port: 35496, Seq: 0, Ack: 1, Len: 0


*enp0s9
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
10 0.797775063 fe80::b95:fa83:fc84… ff02::fb MDNS 301 Standard query response
11 0.802150561 192.168.56.1 224.0.0.251 MDNS 221 Standard query response
12 0.807299934 fe80::b95:fa83:fc84… ff02::fb MDNS 241 Standard query response
13 5.714036321 192.168.56.102 192.168.56.100 DHCP 331 DHCP Request - Transact
14 5.825605861 192.168.56.100 192.168.56.102 DHCP 590 DHCP ACK - Transact
15 6.086806470 192.168.56.1 192.168.56.255 BROWSER 263 Host Announcement CROWNI
16 10.934276410 PcsCompu_87:af:b6 PcsCompu_c9:6a:36 ARP 42 Who has 192.168.56.100?
17 10.935432431 PcsCompu_c9:6a:36 PcsCompu_87:af:b6 ARP 60 192.168.56.100 is at 08
18 36.226466145 192.168.56.102 192.168.56.110 TCP 74 35496 → 80 [SYN] Seq=0
19 36.230195843 PcsCompu_c1:14:0f Broadcast ARP 60 Who has 192.168.56.102?
20 36.230216412 PcsCompu_87:af:b6 PcsCompu_c1:14:0f ARP 42 192.168.56.102 is at 08
21 36.233787963 192.168.56.110 192.168.56.102 TCP 74 80 → 35496 [SYN, ACK] Se
22 36.233861605 192.168.56.102 192.168.56.110 TCP 66 35496 → 80 [ACK] Seq=1
23 36.235205530 192.168.56.102 192.168.56.110 HTTP 423 GET /mutillidae/ HTTP/1
24 36.236979016 192.168.56.110 192.168.56.102 TCP 66 80 → 35496 [ACK] Seq=1
Frame 23: 423 bytes on wire (3384 bits), 423 bytes captured (3384 bits) on interface enp0s9, id 0
Ethernet II, Src: PcsCompu_87:af:b6 (08:00:27:87:af:b6), Dst: PcsCompu_c1:14:0f (08:00:27:c1:14:0f)
Internet Protocol Version 4, Src: 192.168.56.102, Dst: 192.168.56.110
Transmission Control Protocol, Src Port: 35496, Dst Port: 80, Seq: 1, Ack: 1, Len: 357
Hypertext Transfer Protocol


*enp0s9
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
50 36.694021265 192.168.56.110 192.168.56.102 HTTP 857 HTTP/1.1 200 OK (text/
51 36.694933794 192.168.56.102 192.168.56.110 TCP 66 35496 → 80 [ACK] Seq=73
52 36.695552213 192.168.56.102 192.168.56.110 TCP 66 35496 → 80 [ACK] Seq=73
53 36.698755860 192.168.56.102 192.168.56.110 HTTP 458 GET /mutillidae/styles/
54 36.699294640 192.168.56.102 192.168.56.110 TCP 66 35496 → 80 [FIN, ACK] Se
55 36.725866164 192.168.56.102 192.168.56.110 TCP 66 [TCP Retransmission] 35
56 36.730958349 192.168.56.110 192.168.56.102 TCP 78 80 → 35496 [ACK] Seq=33
57 36.733505685 192.168.56.110 192.168.56.102 HTTP 2612 HTTP/1.1 200 OK (text/
58 36.733551449 192.168.56.102 192.168.56.110 TCP 54 35496 → 80 [RST] Seq=11
59 50.821182340 192.168.56.102 192.168.56.110 TCP 74 39966 → 80 [SYN] Seq=0
60 50.822815113 192.168.56.110 192.168.56.102 TCP 74 80 → 39966 [SYN, ACK] Se
61 50.822841917 192.168.56.102 192.168.56.110 TCP 66 39966 → 80 [ACK] Seq=1
62 50.823073553 192.168.56.102 192.168.56.110 HTTP 464 GET / HTTP/1.1
63 50.824403380 192.168.56.110 192.168.56.102 TCP 66 80 → 39966 [ACK] Seq=1
64 50.845625609 192.168.56.110 192.168.56.102 HTTP 1190 HTTP/1.1 200 OK (text/
Frame 55: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface enp0s9, id 0
Ethernet II, Src: PcsCompu_87:af:b6 (08:00:27:87:af:b6), Dst: PcsCompu_c1:14:0f (08:00:27:c1:14:0f)
Internet Protocol Version 4, Src: 192.168.56.102, Dst: 192.168.56.110
Transmission Control Protocol, Src Port: 35496, Dst Port: 80, Seq: 1130, Ack: 33382, Len: 0





From the packet captured, protocols identified include:

HTTP – Hypertext Control Protocol

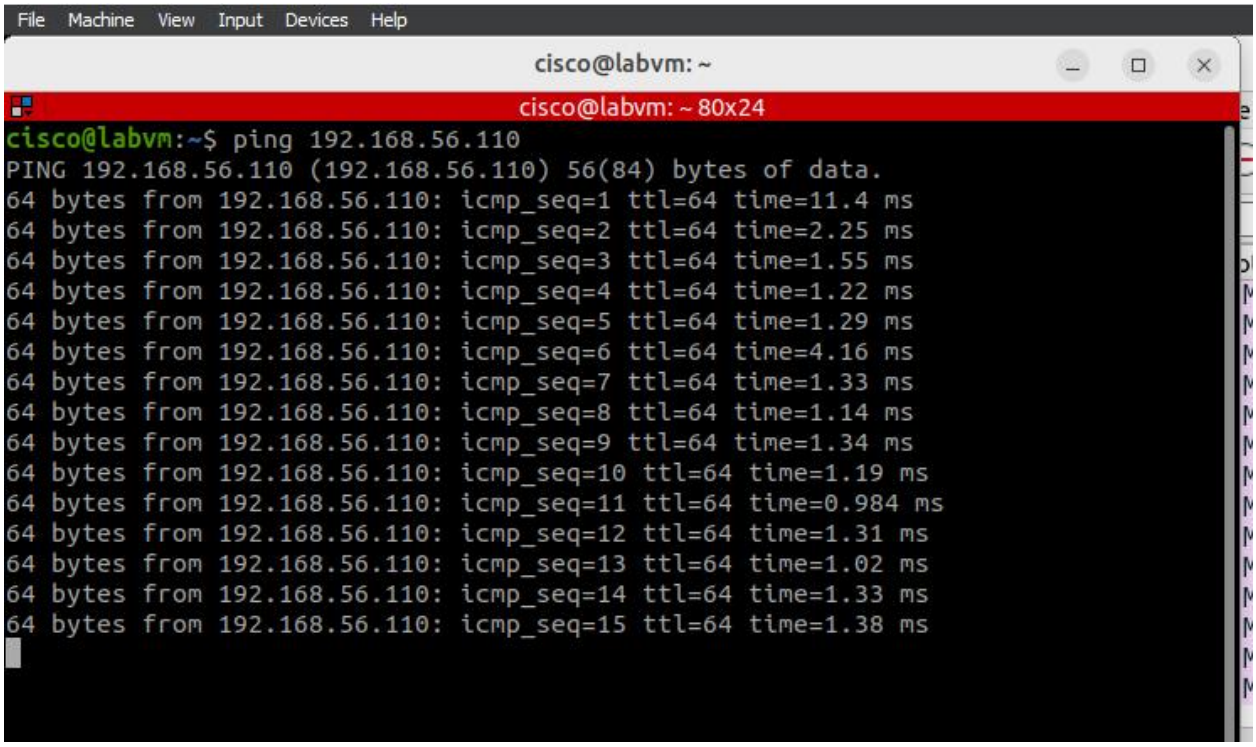
TCP – Transmission control protocol

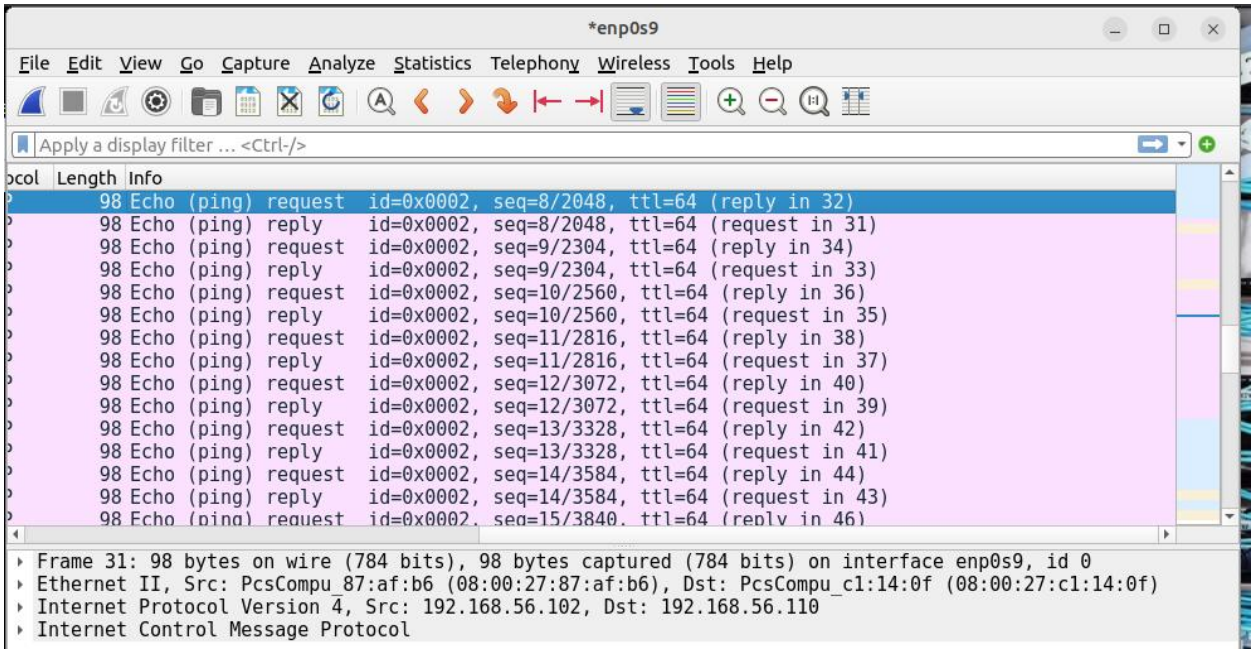
Address Resolution Protocol

There was handshake process as I navigated the internetSYN, ACK was identified. The following port was also in use: 80, 35464,.

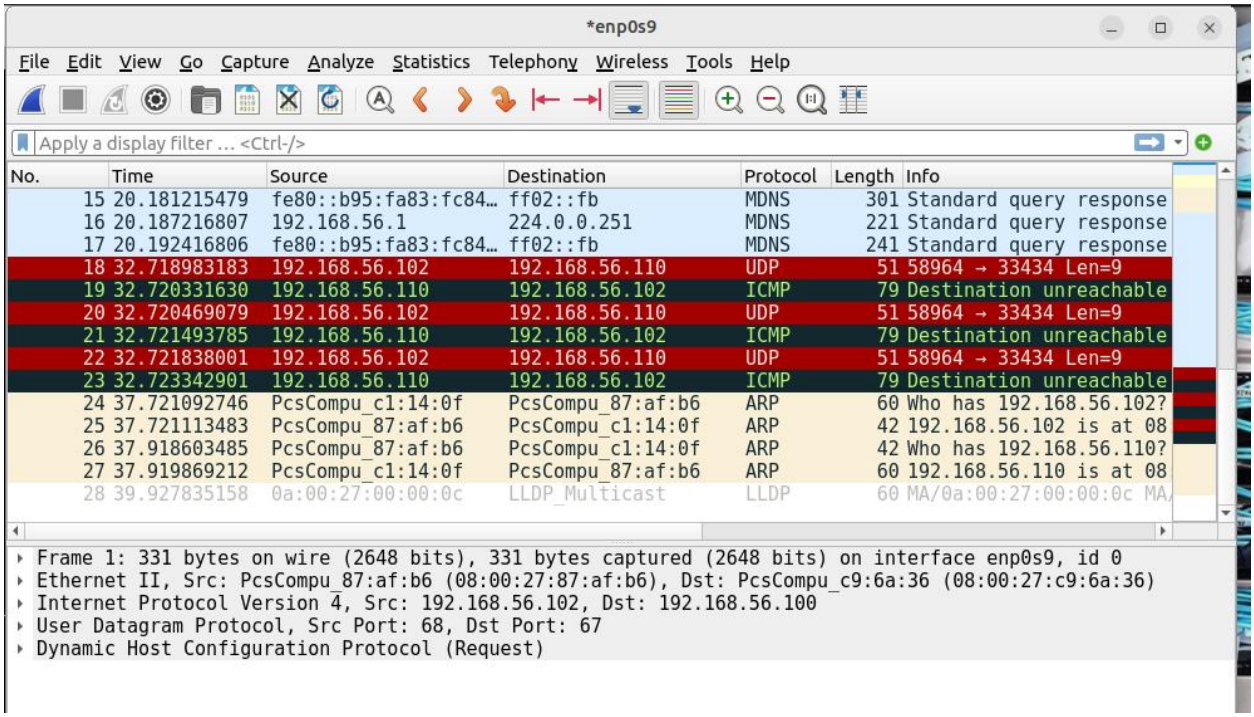
Other significant event identified include failed login in the OWASP top ten application which was used to simulate attack on the web app.

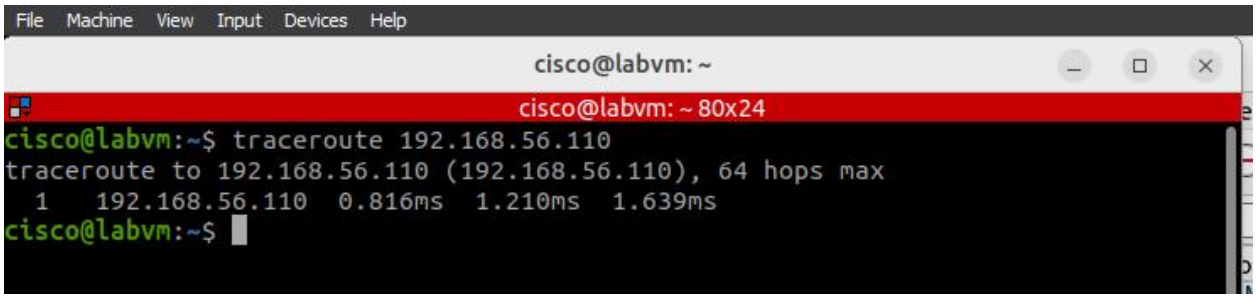
Below is the terminal and wireshark capture as I ping OWASP from my VM

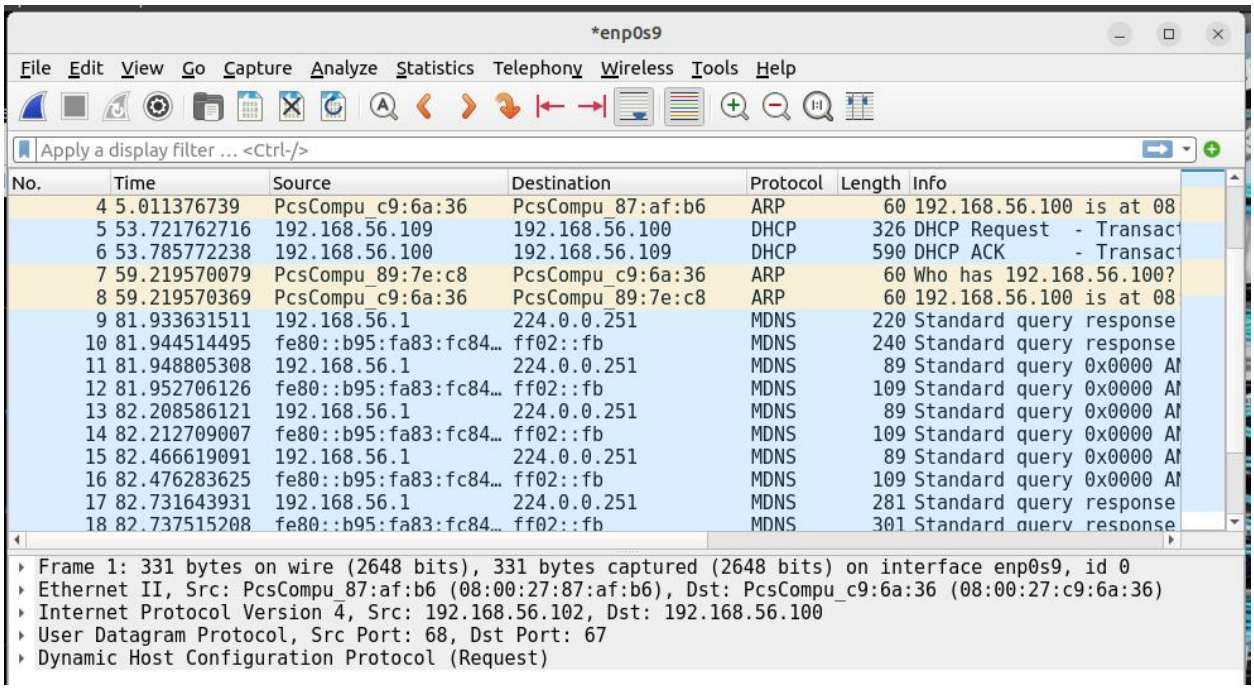


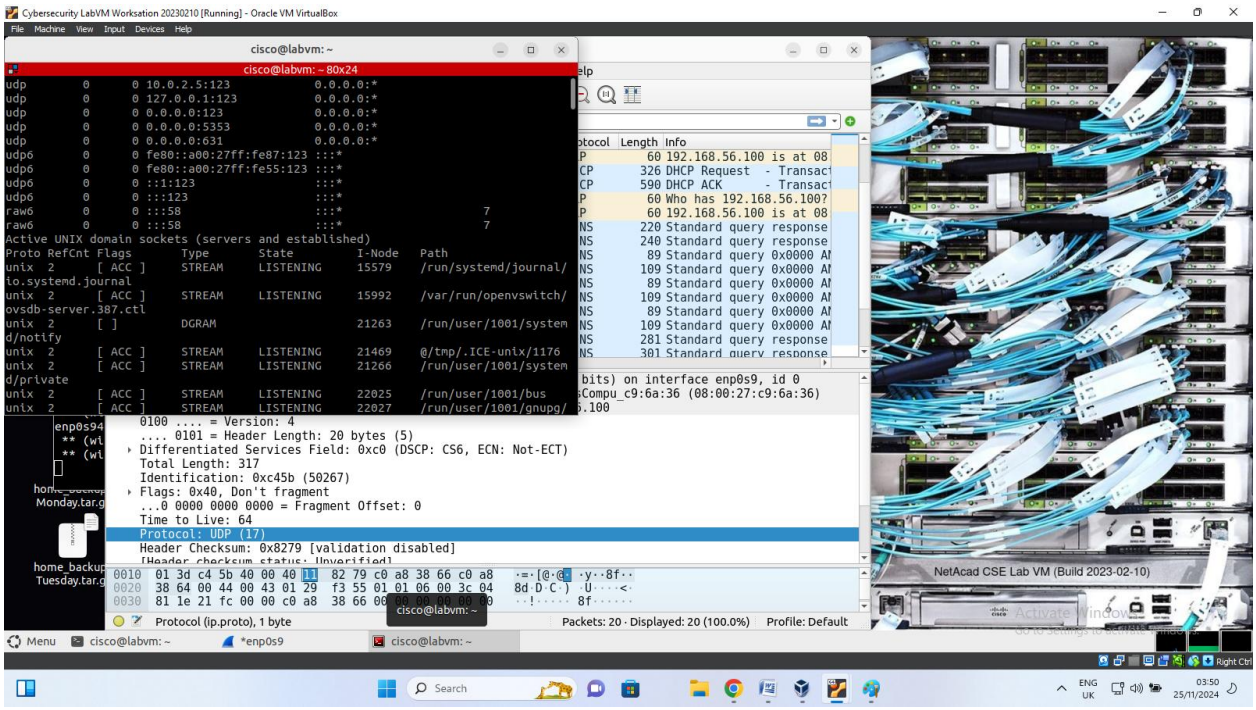


BELOW IS THE TERMINAL AND WIRESHARK FOR TRACEROUTE



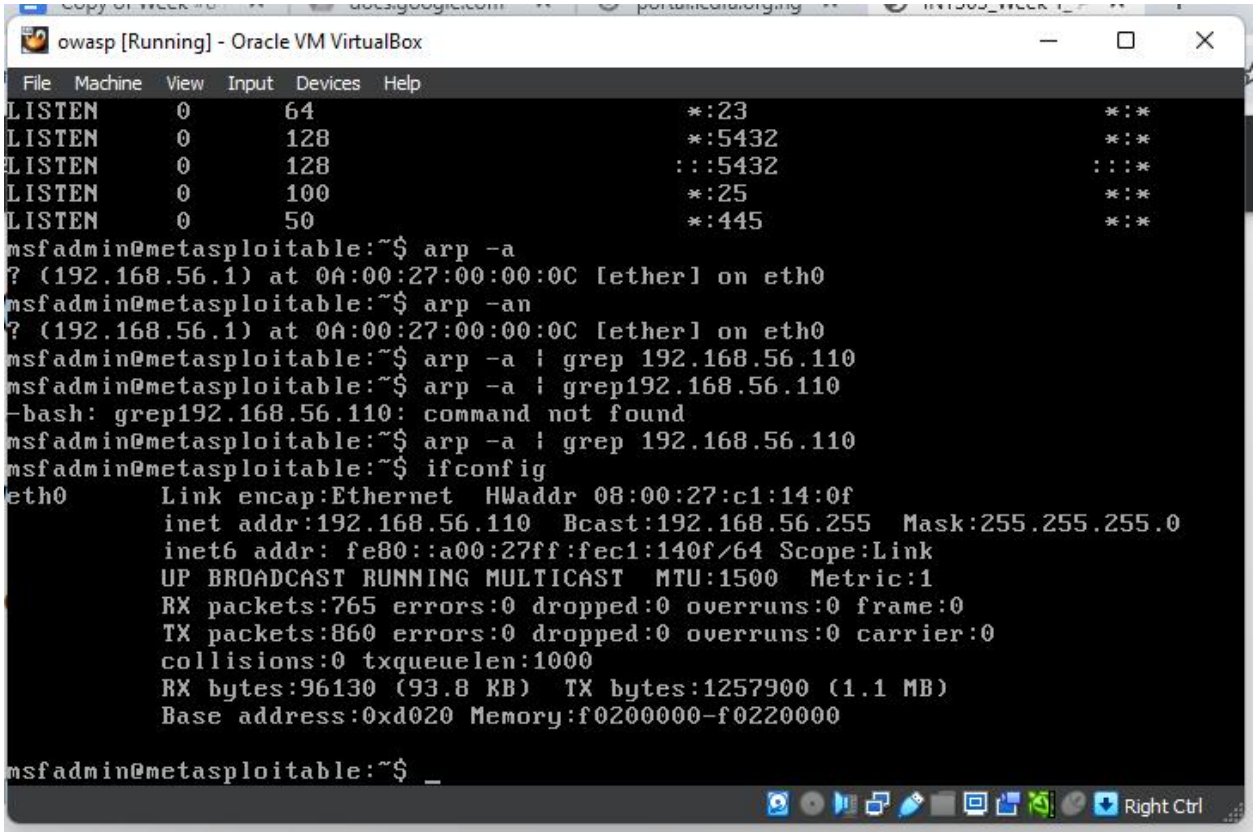


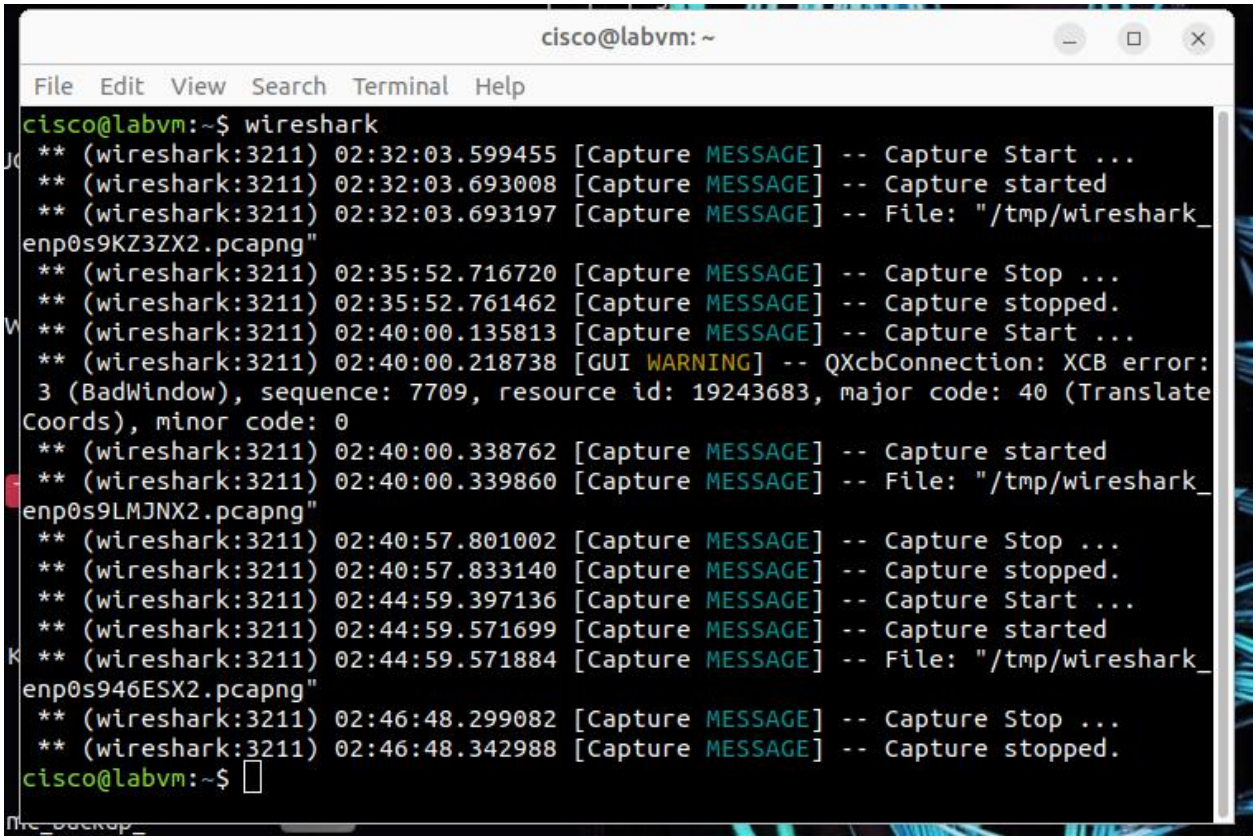




INT303: Networking Fundamentals – Lab 2

Exercise 1: Viewing and Configuring Network Interfaces


owasp [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
LISTEN 0 64 *:23 *:*
LISTEN 0 128 *:5432 *:*
LISTEN 0 128 :::5432 :::*
LISTEN 0 100 *:25 *:*
LISTEN 0 50 *:445 *:*
msfadmin@metasploitable:~$ arp -a
? (192.168.56.1) at 0A:00:27:00:00:0C [ether] on eth0
msfadmin@metasploitable:~$ arp -an
? (192.168.56.1) at 0A:00:27:00:00:0C [ether] on eth0
msfadmin@metasploitable:~$ arp -a | grep 192.168.56.110
msfadmin@metasploitable:~$ arp -a | grep192.168.56.110
-bash: grep192.168.56.110: command not found
msfadmin@metasploitable:~$ arp -a | grep 192.168.56.110
msfadmin@metasploitable:~$ ifconfig
eth0 Link encap:Ethernet HWaddr 08:00:27:c1:14:0f
inet addr:192.168.56.110 Bcast:192.168.56.255 Mask:255.255.255.0
inet6 addr: fe80::a00:27ff:fec1:140f/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:765 errors:0 dropped:0 overruns:0 frame:0
TX packets:860 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:96130 (93.8 KB) TX bytes:1257900 (1.1 MB)
Base address:0xd020 Memory:f0200000-f0220000
msfadmin@metasploitable:~$ _
Right Ctrl


cisco@labvm: ~
File Edit View Search Terminal Help
cisco@labvm:~$ wireshark
** (wireshark:3211) 02:32:03.599455 [Capture MESSAGE] -- Capture Start ...
** (wireshark:3211) 02:32:03.693008 [Capture MESSAGE] -- Capture started
** (wireshark:3211) 02:32:03.693197 [Capture MESSAGE] -- File: "/tmp/wireshark_enp0s9KZ3ZX2.pcapng"
** (wireshark:3211) 02:35:52.716720 [Capture MESSAGE] -- Capture Stop ...
** (wireshark:3211) 02:35:52.761462 [Capture MESSAGE] -- Capture stopped.
** (wireshark:3211) 02:40:00.135813 [Capture MESSAGE] -- Capture Start ...
** (wireshark:3211) 02:40:00.218738 [GUI WARNING] -- QXcbConnection: XCB error: 3 (BadWindow), sequence: 7709, resource id: 19243683, major code: 40 (TranslateCoords), minor code: 0
** (wireshark:3211) 02:40:00.338762 [Capture MESSAGE] -- Capture started
** (wireshark:3211) 02:40:00.339860 [Capture MESSAGE] -- File: "/tmp/wireshark_enp0s9LMJNX2.pcapng"
** (wireshark:3211) 02:40:57.801002 [Capture MESSAGE] -- Capture Stop ...
** (wireshark:3211) 02:40:57.833140 [Capture MESSAGE] -- Capture stopped.
** (wireshark:3211) 02:44:59.397136 [Capture MESSAGE] -- Capture Start ...
** (wireshark:3211) 02:44:59.571699 [Capture MESSAGE] -- Capture started
** (wireshark:3211) 02:44:59.571884 [Capture MESSAGE] -- File: "/tmp/wireshark_enp0s946ESX2.pcapng"
** (wireshark:3211) 02:46:48.299082 [Capture MESSAGE] -- Capture Stop ...
** (wireshark:3211) 02:46:48.342988 [Capture MESSAGE] -- Capture stopped.
cisco@labvm:~$

Describe the difference between a loopback interface and an external network interface

1. Loopback Interface

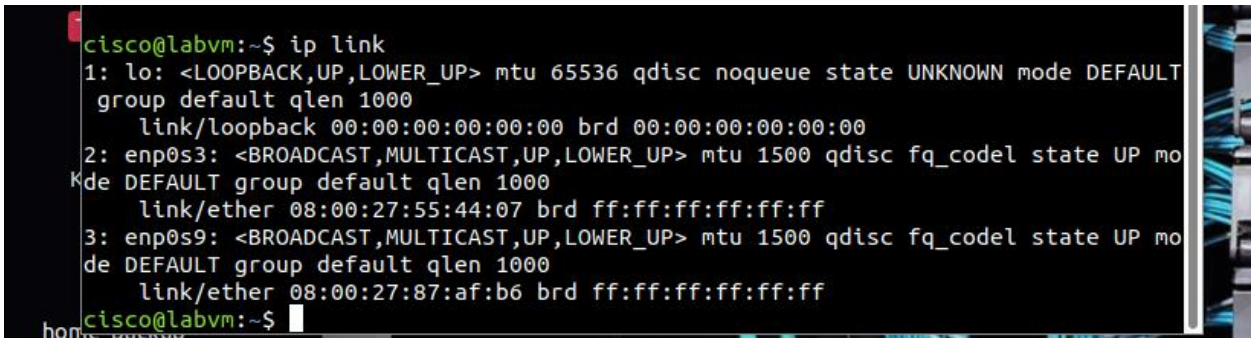
- **Definition**: A virtual interface used internally by the system to communicate with itself.
- **IP Address**: Typically uses the **127.0.0.1** address (IPv4) or **::1** (IPv6).
- **Purpose**:
  - Used for testing and diagnostics (e.g., ensuring network stack functionality).
  - Allows applications on the same system to communicate with each other over network protocols.
  - Does not involve any physical network hardware or external communication.
- **Example Use Cases**:
  - Running a web server locally (e.g., localhost).
  - Debugging or testing networking configurations.

---

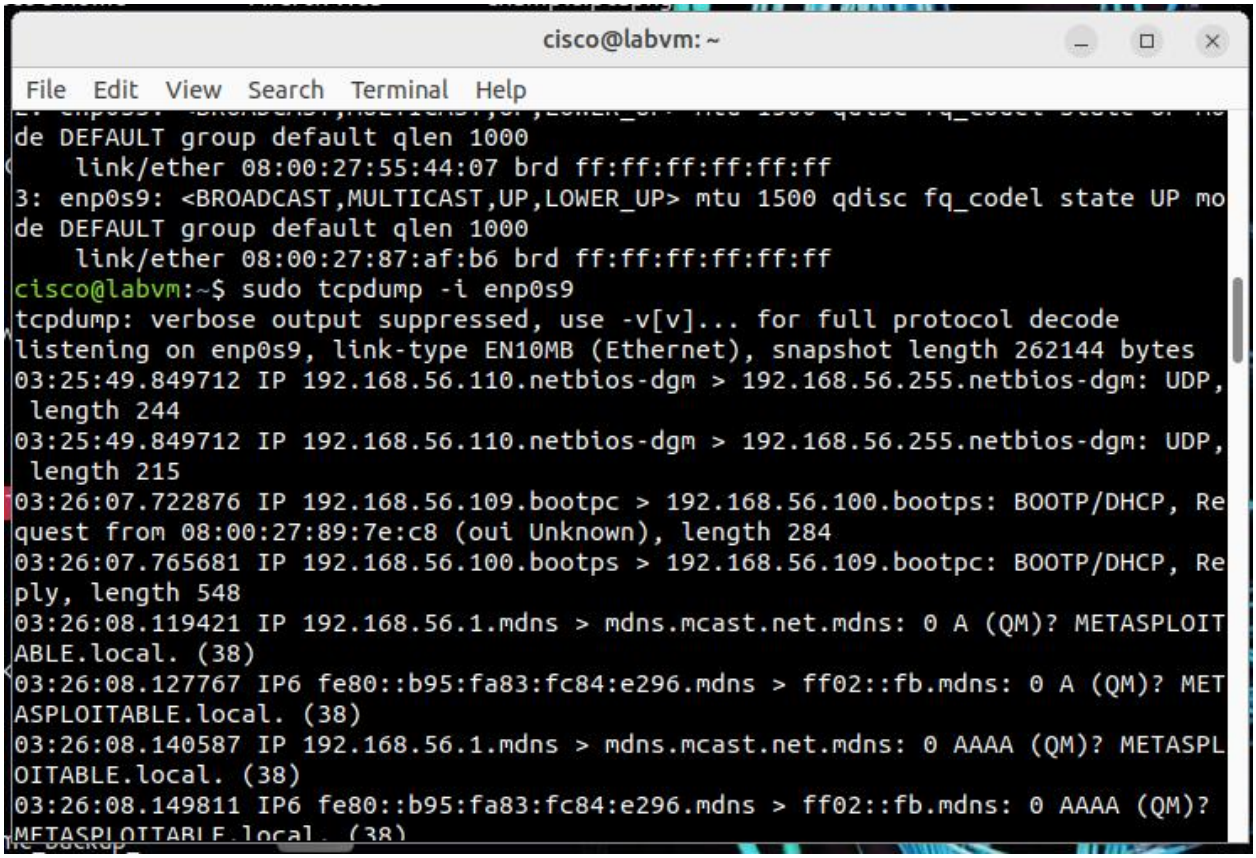
2. External Network Interface

- **Definition**: A physical or virtual network interface that connects a device to an external network.
- **IP Address**: Typically assigned by a DHCP server, manually configured, or set automatically through auto-configuration (e.g., IPv6 SLAAC).
- **Purpose**:
  - Facilitates communication with other devices on the same network or over the internet.
  - Handles traffic going to and coming from external networks.
- **Example Use Cases**:
  - Connecting to a local area network (LAN) or Wi-Fi.
  - Accessing external resources, such as websites or cloud services.

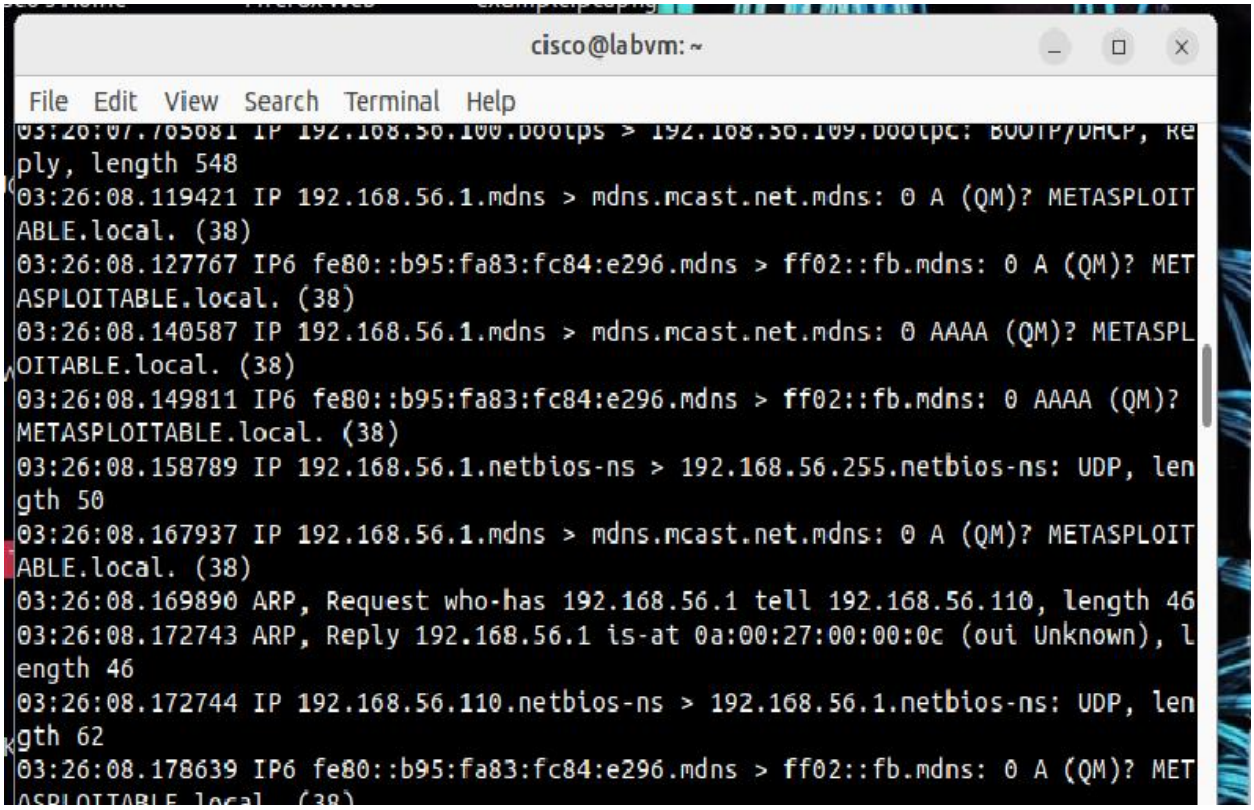
Exercise 2: Capturing Packets on a Specific Interface



Capture packet on specific interface



Captured packets include that of the traffic on OWASP vm



A network interface is a hardware or virtual component that facilitates communication between a computer or device and a network. It plays a crucial role in the transmission and reception of packets, which are the fundamental units of data exchanged over a network. Here's an overview of its roles:

Transmitting Packets:

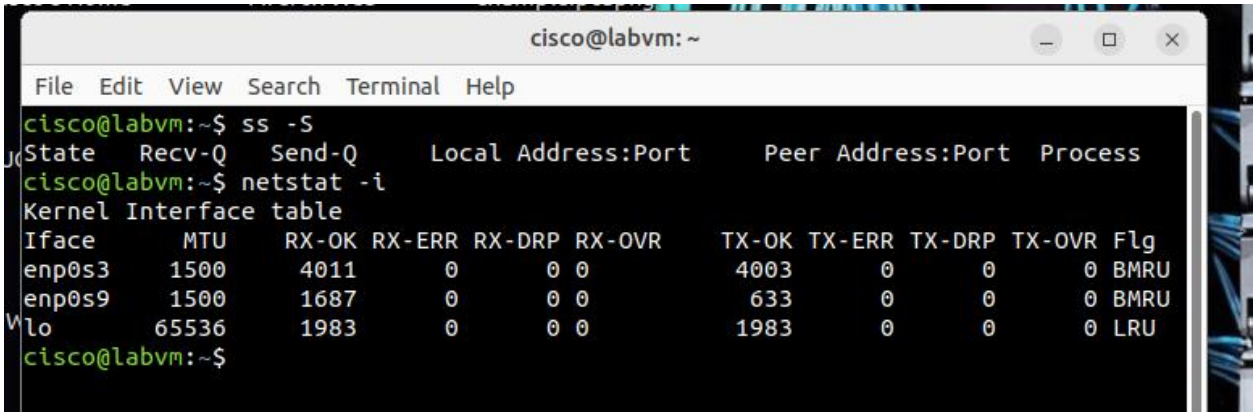
1. **Packet Construction and Encoding:**
   When an application or system sends data, the network interface receives this data from the operating system in the form of packets. It encapsulates the data into frames suitable for transmission based on the network protocol (e.g., Ethernet, Wi-Fi).
2. **Addressing:**
   The interface appends necessary information like source and destination MAC addresses to the frames for proper routing at the data link layer.
3. **Medium Access:**
   For shared network mediums (like Wi-Fi or Ethernet), the interface uses protocols such as CSMA/CD (Ethernet) or CSMA/CA (Wi-Fi) to determine when to send data without causing collisions.

4. **Signal Conversion:**
   The network interface converts the digital data into appropriate signals (electrical, optical, or wireless) for transmission over the physical medium.
5. **Error Detection:**
   It generates checksums or other error-detection codes to ensure data integrity during transmission.

Receiving Packets:

1. **Signal Reception and Decoding:**
   The interface receives signals from the physical medium and converts them back into digital data.
2. **Error Checking:**
   It checks for errors in the received frames using checksums or error-detection codes. Corrupted frames are typically discarded.
3. **Frame Decapsulation:**
   The interface removes the frame headers and trailers, extracting the packets to pass them to higher layers in the networking stack.
4. **Filtering and Forwarding:**
   The interface ensures that only packets addressed to the local device or broadcast packets are processed, discarding irrelevant traffic.
5. **Acknowledgments:**
   In some protocols, the interface sends acknowledgments for successfully received packets to maintain reliability.

Exercise 3: Examining Network Statistics



Current Status of Network Interfaces

The output from netstat -i provides information about the network interfaces on the system. Here's a breakdown:

1. **Interface enp0s3**:
    - **MTU**: 1500 (standard for Ethernet connections).
    - **RX-OK**: 4011 packets received successfully.
    - **RX-ERR, RX-DRP, RX-OVR**: 0, indicating no errors, drops, or overruns during reception.
    - **TX-OK**: 4003 packets transmitted successfully.
    - **TX-ERR, TX-DRP, TX-OVR**: 0, indicating no errors, drops, or overruns during transmission.
    - **Flags**: BMRU indicates that this is a broadcast, multicast, running, and up interface. It is operational and actively used.
2. **Interface enp0s9**:
    - **MTU**: 1500.
    - **RX-OK**: 1687 packets received successfully.
    - **TX-OK**: 633 packets transmitted successfully.
    - No errors or drops, similar to enp0s3.
    - Flags: BMRU, indicating it is also operational and available for communication.
3. **Interface lo** (loopback interface):
    - **MTU**: 65536 (high to accommodate internal traffic efficiently).
    - **RX-OK/TX-OK**: 1983 packets received and sent internally.
    - Flags: LRU, indicating it is a loopback, running, and up interface, used for internal communications like testing or local processes.

Active Connections

The output from ss -S shows no active connections at this time:

- **State**: No established states like ESTABLISHED or LISTEN are displayed, which means there are currently no visible active or listening sockets on the system.

Significance of Statistics for Monitoring Network Performance

1. **Packet Reception and Transmission**:
    - The RX-OK and TX-OK values indicate how many packets have been successfully received and transmitted by the interfaces.
    - Higher values suggest that the interface is actively handling network traffic.
2. **Error-Free Operation**:

- The absence of errors (RX-ERR, TX-ERR) and drops (RX-DRP, TX-DRP) shows the interfaces are functioning reliably without issues in processing or buffer overruns.

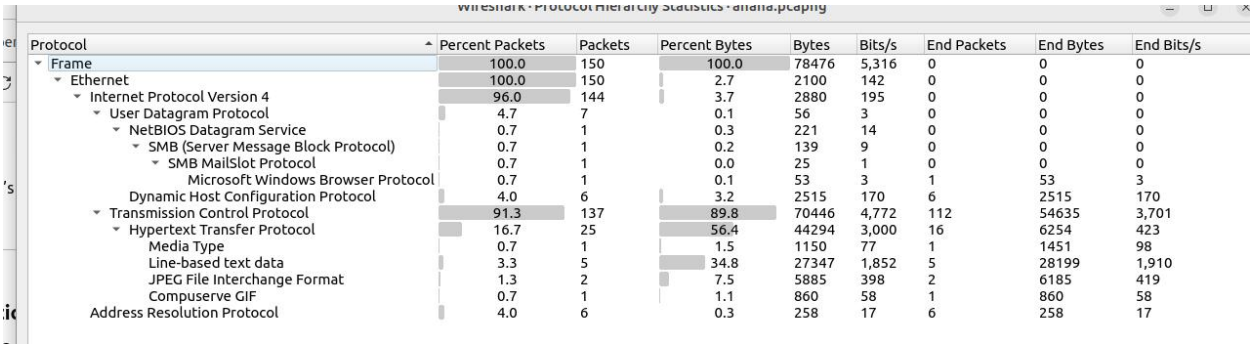
3. **Loopback Interface (lo)**:
   - Its activity (RX-OK/TX-OK of 1983) indicates that applications or services on the system are using internal communication effectively.
4. **Interface Utilization**:
   - Comparing traffic levels across interfaces (enp0s3 vs. enp0s9) shows enp0s3 is handling more traffic, suggesting it might be the primary external interface.
5. **Active Connections**:
   - No connections in the ss -S output means the system isn't currently engaged in external communications. This could indicate an idle state or no running services requiring network access.

How These Metrics Help in Monitoring

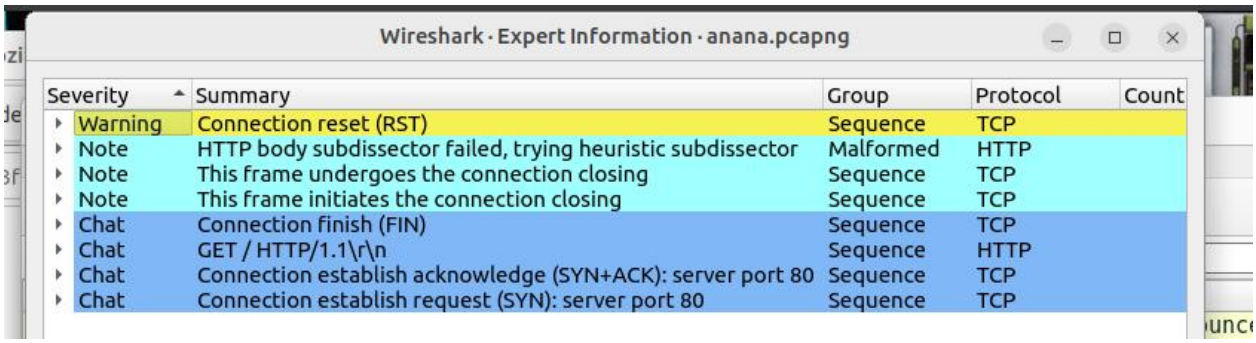
- **Baseline Performance**: Regularly observing these stats helps establish a normal traffic pattern for the system.
- **Troubleshooting**: Errors or drops (non-zero RX-ERR, TX-ERR, RX-DRP, TX-DRP) can indicate issues like hardware problems, misconfigured MTUs, or overloaded buffers.
- **Load Balancing**: Monitoring traffic levels across multiple interfaces (enp0s3 and enp0s9) helps in assessing the distribution of network load.
- **System Health**: Loopback traffic (lo) activity ensures that internal services are functioning correctly.

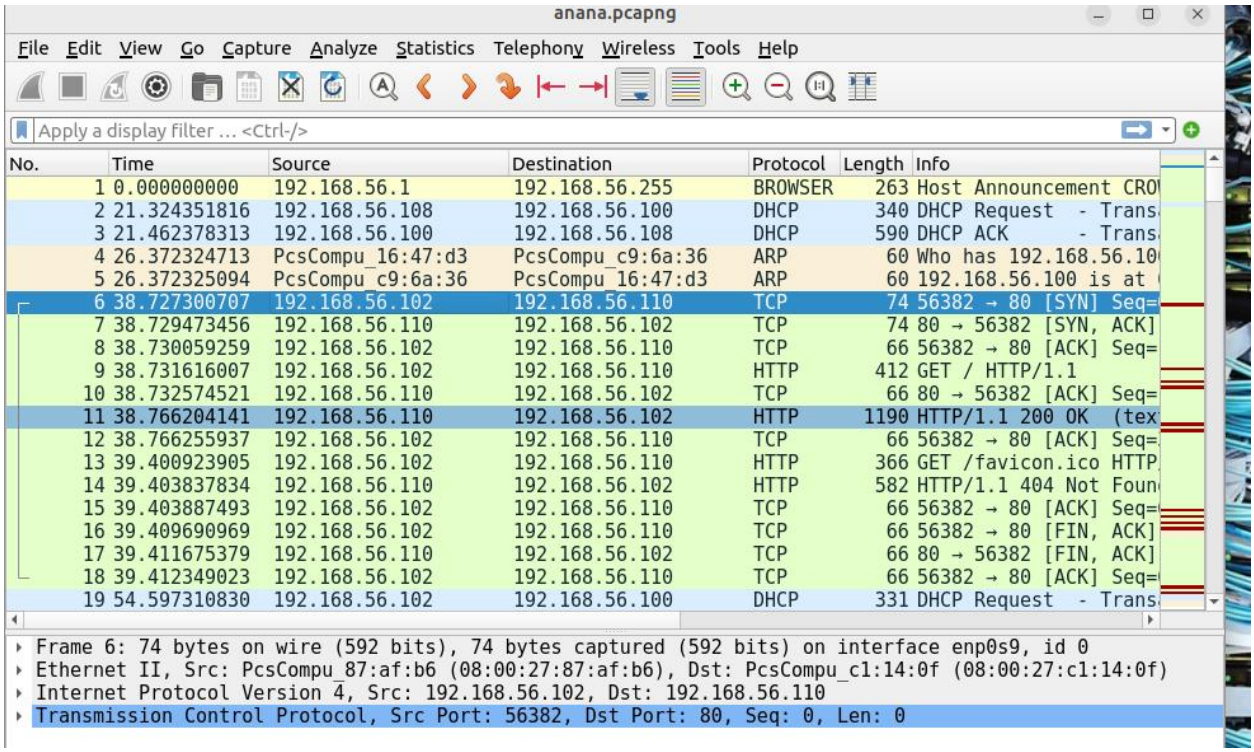
: Exercise 4: Monitoring Network Traffic with Wireshark

Protocols



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	150	100.0	78476	5,316	0	0	0
Ethernet	100.0	150	2.7	2100	142	0	0	0
Internet Protocol Version 4	96.0	144	3.7	2880	195	0	0	0
User Datagram Protocol	4.7	7	0.1	56	3	0	0	0
NetBIOS Datagram Service	0.7	1	0.3	221	14	0	0	0
SMB (Server Message Block Protocol)	0.7	1	0.2	139	9	0	0	0
SMB MailSlot Protocol	0.7	1	0.0	25	1	0	0	0
Microsoft Windows Browser Protocol	0.7	1	0.1	53	3	1	53	3
Dynamic Host Configuration Protocol	4.0	6	3.2	2515	170	6	2515	170
Transmission Control Protocol	91.3	137	89.8	70446	4,772	112	54635	3,701
Hypertext Transfer Protocol	16.7	25	56.4	44294	3,000	16	6254	423
Media Type	0.7	1	1.5	1150	77	1	1451	98
Line-based text data	3.3	5	34.8	27347	1,852	5	28199	1,910
JPEG File Interchange Format	1.3	2	7.5	5885	398	2	6185	419
Compuserve GIF	0.7	1	1.1	860	58	1	860	58
Address Resolution Protocol	4.0	6	0.3	258	17	6	258	17



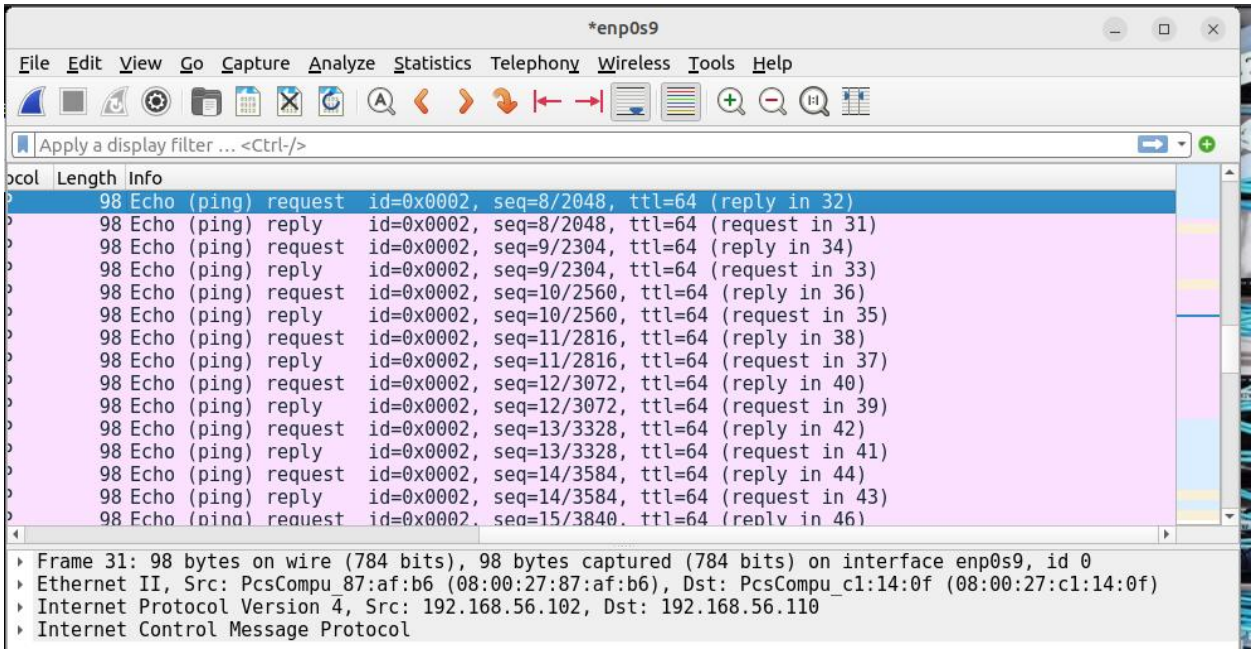


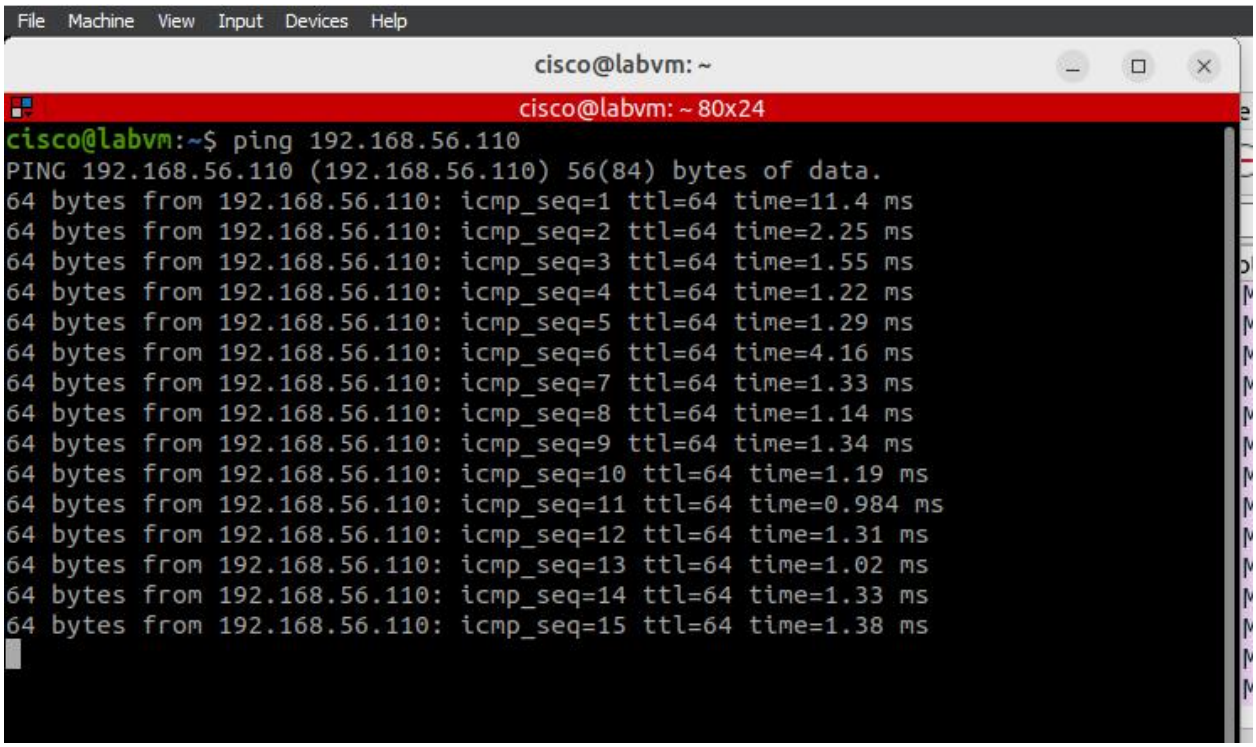
For the packets going to and from OWASP IS TCP “Transmission Control Protocol” I also noticed handshake in the communication betweenOEASP and vm It clearly indicates the ip address of of OWASP and that of the host machine. Port 80 was identified as the destination port while port 56382 was the source port. Protocols als identified include ARP, Address Resolution Protocols, DHCP etc.

Exercise 5: Packet Transmission Analysis

Task: Perform a ping or TCP connection request to the OWASP Broken Web ApplicationIPandcapture the packet details using tcpdump or Wireshark.

Pinging of OWASP captured in the wireshark





Observations During Packet Transmission:

When pinging an OWASP vulnerable machine from your VM, you'll observe the **ICMP (Internet Control Message Protocol)** packet exchange. Here's the process:

1. **ICMP Echo Request:**
   The VM sends an ICMP Echo Request packet to the target (OWASP vulnerable machine).

2. **Packet Transmission:**
   The packet travels through multiple network layers, being encapsulated at each step.
3. **ICMP Echo Reply:**
   The target machine responds with an ICMP Echo Reply packet, indicating successful communication.
4. **Round-Trip Time (RTT):**
   The time taken for the packet to travel to the target and back is measured, providing latency details.

Handshake Process (TCP Connection):

For TCP-based communication (not ICMP/ping), a **three-way handshake** occurs:

1. **SYN (Synchronize):**
   The VM sends a TCP segment with the SYN flag set to initiate a connection.
2. **SYN-ACK (Synchronize-Acknowledge):**
   The target replies with a segment having both SYN and ACK flags set, acknowledging the request.
3. **ACK (Acknowledge):**
   The VM responds with a segment with the ACK flag set, confirming the connection establishment.

Round-Trip for Ping (ICMP):

- Ping operates at the **Network Layer (OSI Layer 3)** using ICMP.
- The Echo Request and Reply packets traverse lower layers for transmission:
  - **Data Link Layer (Layer 2):** Encapsulation into frames for the physical medium.
  - **Physical Layer (Layer 1):** Transmission as signals over the medium.

Layers Involved in TCP Handshake:

- **Application Layer (OSI Layer 7 / TCP/IP Application Layer):** Initiates the request (e.g., through an application like a browser or script).
- **Transport Layer (OSI Layer 4 / TCP/IP Transport Layer):** Manages the handshake using TCP for reliable communication.
- **Network Layer (OSI Layer 3 / TCP/IP Internet Layer):** Routes packets using IP addresses.
- **Data Link Layer (OSI Layer 2 / TCP/IP Link Layer):** Handles frames and physical addressing (MAC).

- **Physical Layer (OSI Layer 1):** Converts data into signals for transmission.

Exercise 6: Troubleshooting Network Interface Issues

Observations and Explanations:

1. What happens when you disable the network interface?

- **Loss of Connectivity:**
  Disabling the network interface (sudo ifconfig <interface> down or sudo ip link set <interface> down) immediately cuts off all network communication for that interface. The VM cannot send or receive packets, effectively severing its connection to the OWASP Broken Web Application VM.
- **Route Table Changes:**
  The operating system removes routes associated with the disabled interface from the routing table, rendering the system unable to resolve or communicate with external or connected hosts via that interface.
- **Application Impact:**
  Ongoing connections (e.g., SSH, HTTP sessions) using the disabled interface are terminated or timeout due to the lack of network response.

---

2. How does your system respond when the interface is re-enabled?

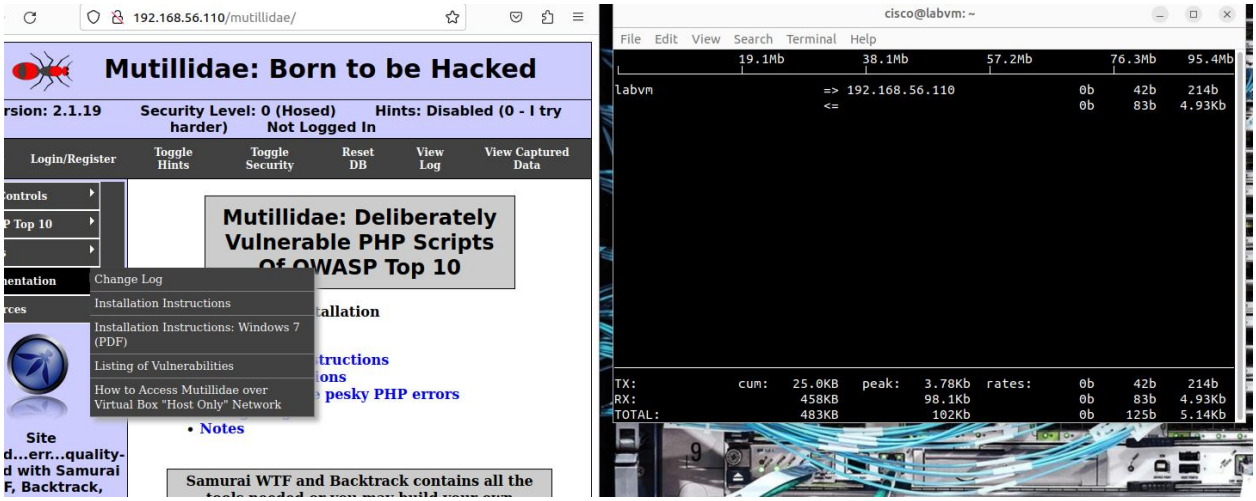
- **Interface Initialization:**
  When the interface is brought back up (sudo ifconfig <interface> up or sudo ip link set <interface> up), it reinitializes, including reapplying its IP configuration (static or via DHCP).
- **Routing Table Update:**
  Routes associated with the interface are restored to the system's routing table, enabling communication to resume.
- **Connectivity Restoration:**
  New connections can be established, and applications relying on network access may recover automatically if they support reconnection.
- **Latency in Recovery:**
  Depending on the network setup (e.g., DHCP lease renewal, ARP table updates), some delay might occur before full connectivity is restored.

---

3. How can network administrators use this knowledge for troubleshooting connectivity issues?

- **Interface Health Checks:**
  Administrators can disable and re-enable network interfaces to test whether connectivity issues are due to hardware failures, driver problems, or network misconfigurations.
- **IP Reassignment:**
  Restarting an interface often triggers DHCP lease renewal or re-applies a static IP configuration, which can resolve misconfigured or stale IP settings.
- **Routing Debugging:**
  Flushing and rebuilding routing tables by reinitializing interfaces can address issues where incorrect or outdated routes prevent proper communication.
- **Driver or Firmware Testing:**
  Temporarily disabling and re-enabling an interface can help identify driver-related instability or issues.
- **Layer Isolation:**
  Administrators can isolate problems to specific OSI layers (e.g., data link or physical layer issues) when disabling and re-enabling interfaces impacts connectivity.

Exercise 7: Bandwidth Monitoring



Bandwidth image capture clearly shown below please note

Outgoing (TX): Data sent to the OWASP VM.

Incoming (RX): Data received from the OWASP VM.

Total bandwidth usage on the interface

Identify the impact of network traffic on your interface. Is there any traffic congestion? NO

Identify the impact of network traffic on your interface. Is there any traffic congestion? How does this help in monitoring network performance?

Impact of Network Traffic on the Interface

The impact of network traffic on your interface can be assessed as follows:

1. **Traffic Load**:
    - If you see consistent, moderate traffic (e.g., 1–10 Mbps), the interaction is likely normal.
    - High or spiking traffic (e.g., >50 Mbps) could indicate heavy activity, such as large file uploads, downloads, or a high volume of requests to/from the OWASP VM.
2. **Potential Congestion**:
    - **No congestion**: If the observed bandwidth usage is significantly below the interface capacity (e.g., for a 1 Gbps interface).
    - **Possible congestion**: If bandwidth usage is close to the interface limit or if packet retransmissions are visible in tools like iftop.

---

3. Traffic Congestion Signs

You may notice the following issues if there's congestion:

- **Increased latency**: Slower response times when interacting with the OWASP VM.
- **Packet drops or retransmissions**: Visible in iftop or other tools.
- **Reduced throughput**: Lower actual data transfer rates compared to the potential bandwidth.

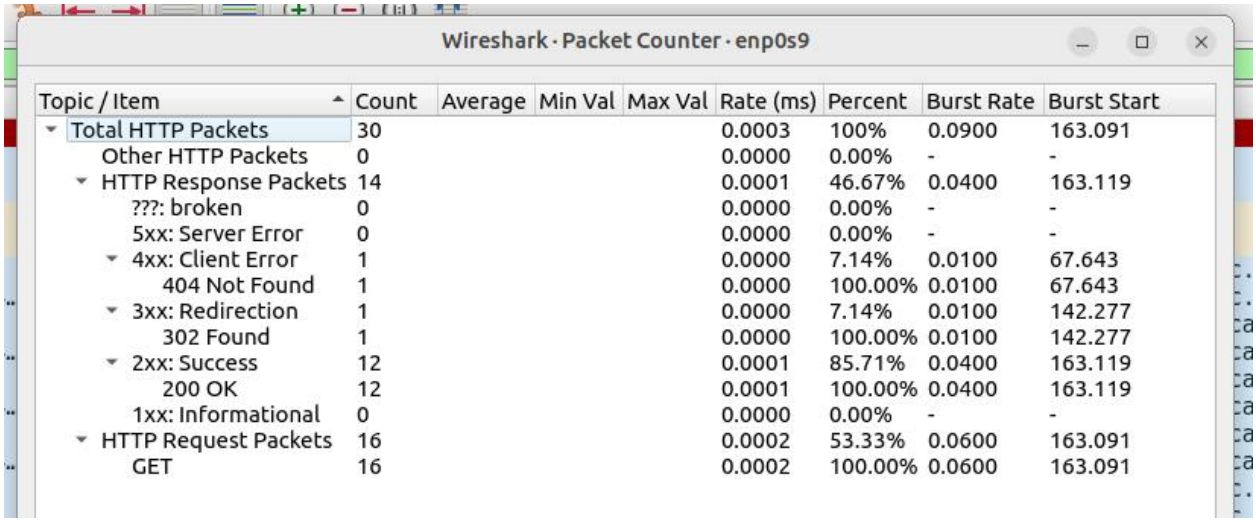
---

4. How This Helps in Monitoring Network Performance

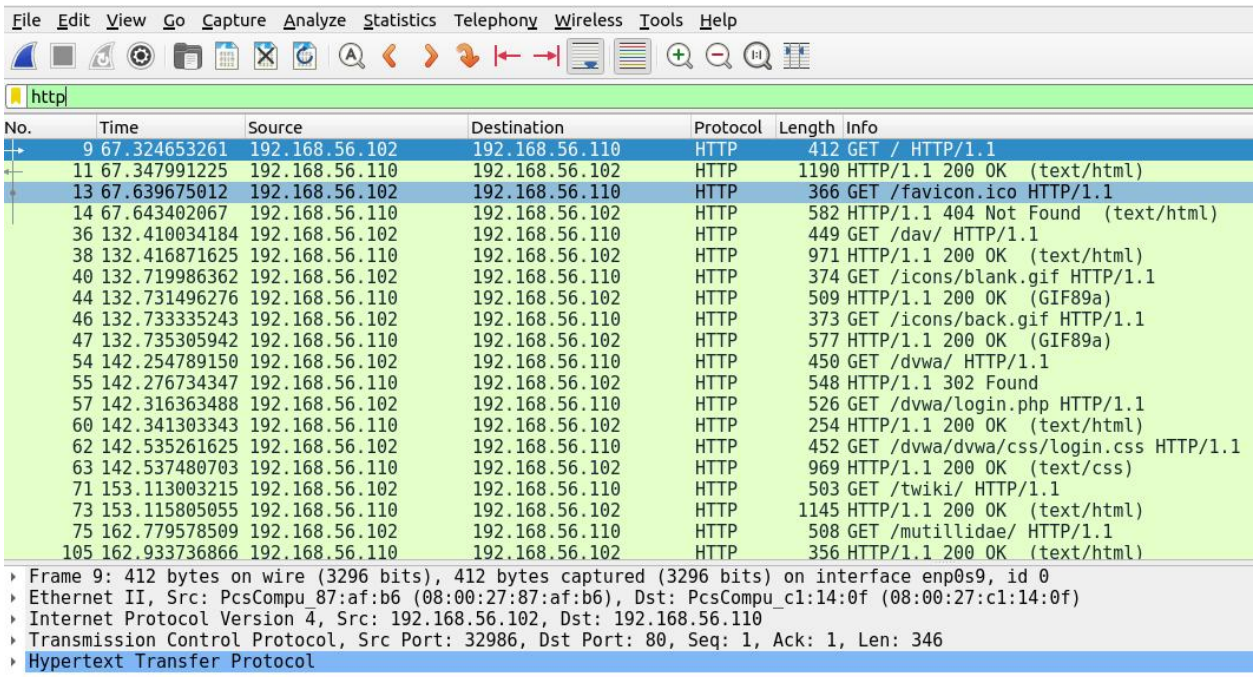
- **Real-Time Bandwidth Monitoring**: iftop and nload provide instant feedback on the current usage of the interface, helping identify active communication flows and bandwidth-hogging activities.
- **Identifying Bottlenecks**: If bandwidth usage nears capacity or congestion is visible, it indicates that the network infrastructure may require optimization or upgrading.
- **Traffic Insights**: iftop displays source and destination IPs, allowing you to confirm the traffic is related to the OWASP VM and not an unknown source.
- **Diagnosing Issues**: Helps in detecting and troubleshooting performance issues like slow application responses or high latency.
- **Resource Planning**: By observing typical bandwidth usage patterns, you can make informed decisions for resource allocation and scaling.

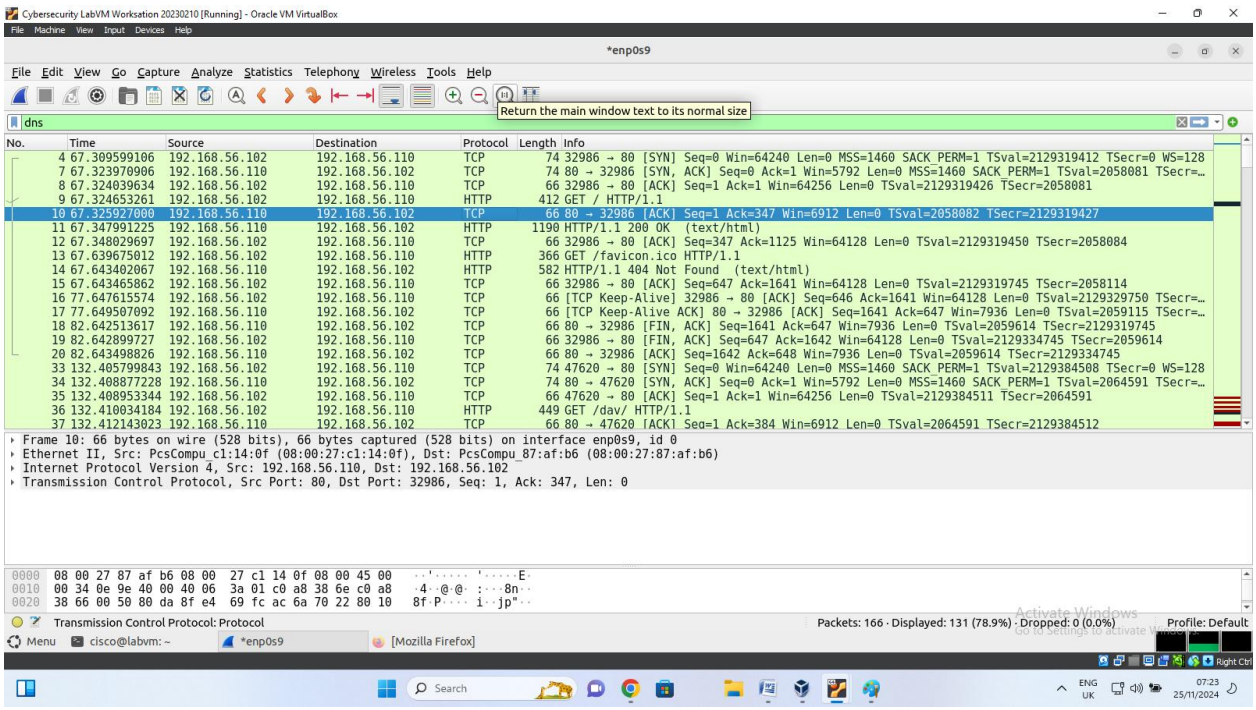
Exercise 8: Advanced Packet Capture Filters

HTTP filtering on the specific interface (enp0s9)



Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Total HTTP Packets	30				0.0003	100%	0.0900	163.091
Other HTTP Packets	0				0.0000	0.00%	-	-
HTTP Response Packets	14				0.0001	46.67%	0.0400	163.119
???: broken	0				0.0000	0.00%	-	-
5xx: Server Error	0				0.0000	0.00%	-	-
4xx: Client Error	1				0.0000	7.14%	0.0100	67.643
404 Not Found	1				0.0000	100.00%	0.0100	67.643
3xx: Redirection	1				0.0000	7.14%	0.0100	142.277
302 Found	1				0.0000	100.00%	0.0100	142.277
2xx: Success	12				0.0001	85.71%	0.0400	163.119
200 OK	12				0.0001	100.00%	0.0400	163.119
1xx: Informational	0				0.0000	0.00%	-	-
HTTP Request Packets	16				0.0002	53.33%	0.0600	163.091
GET	16				0.0002	100.00%	0.0600	163.091





INT303: Networking Fundamentals – Lab 3 Lab 3: TCP/IP Protocol Stack and Packet Inspection Using OWASP BrokenWebApplication IP

Exercise 1: Understanding the TCP/IP Model

TCP/IP Model Layers Overview

1. **Application Layer**
   - **Purpose:** Provides high-level protocols for end-user applications (e.g., HTTP, FTP, SMTP).
   - **Function:** Handles data presentation, session management, and user interface communication.
   - **OSI Correspondence:** Combines OSI's Application (Layer 7), Presentation (Layer 6), and Session (Layer 5) layers.
2. **Transport Layer**
   - **Purpose:** Ensures reliable data transfer between devices using protocols like TCP and UDP.
   - **Function:** Manages flow control, error detection, and data segmentation.
   - **OSI Correspondence:** Matches OSI's Transport Layer (Layer 4).
3. **Network Layer**
   - **Purpose:** Handles logical addressing and routing of packets between devices.
   - **Function:** Uses IP for packet delivery across networks.
   - **OSI Correspondence:** Aligns with OSI's Network Layer (Layer 3).
4. **Link Layer**
   - **Purpose:** Facilitates physical transmission of data over the network medium.
   - **Function:** Manages frame creation, MAC addressing, and error detection.
   - **OSI Correspondence:** Combines OSI's Data Link (Layer 2) and Physical (Layer 1) layers.

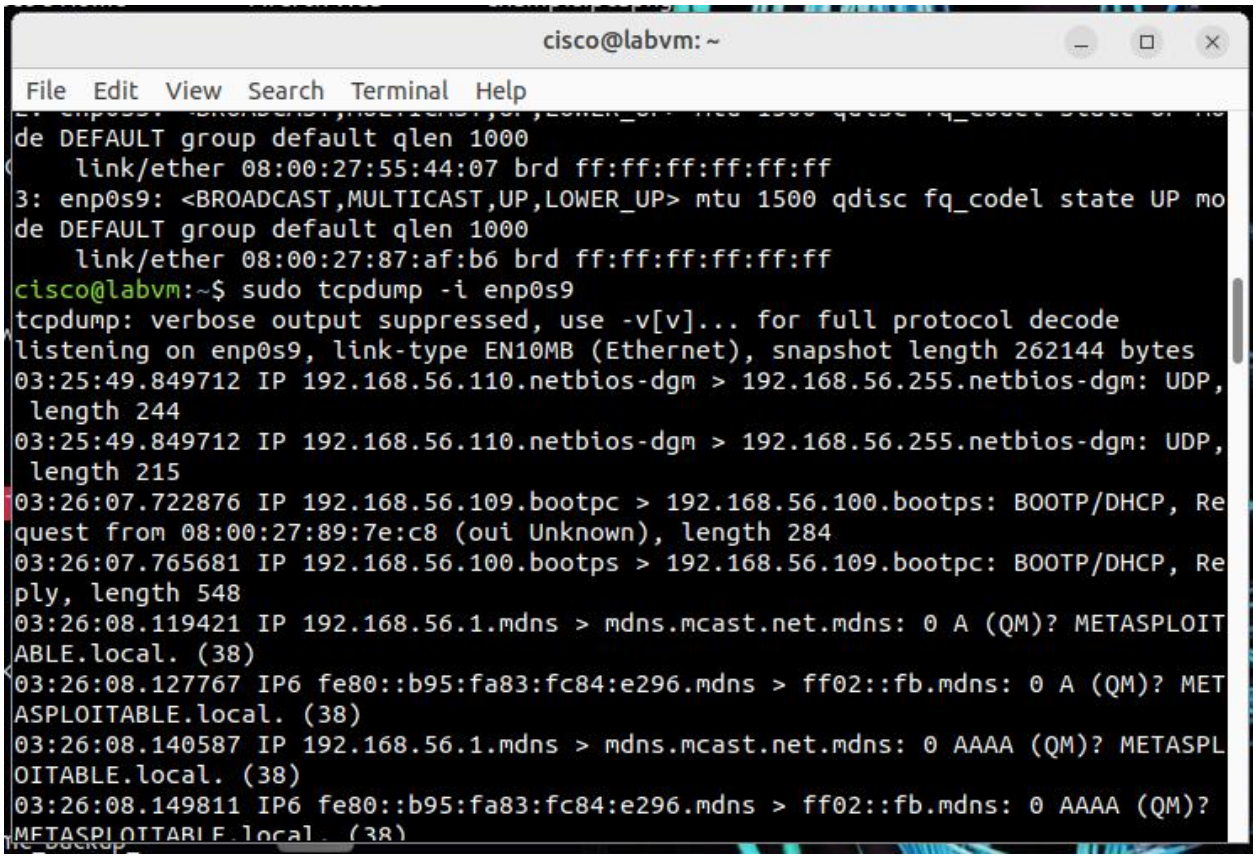
Differences Between TCP/IP and OSI Models

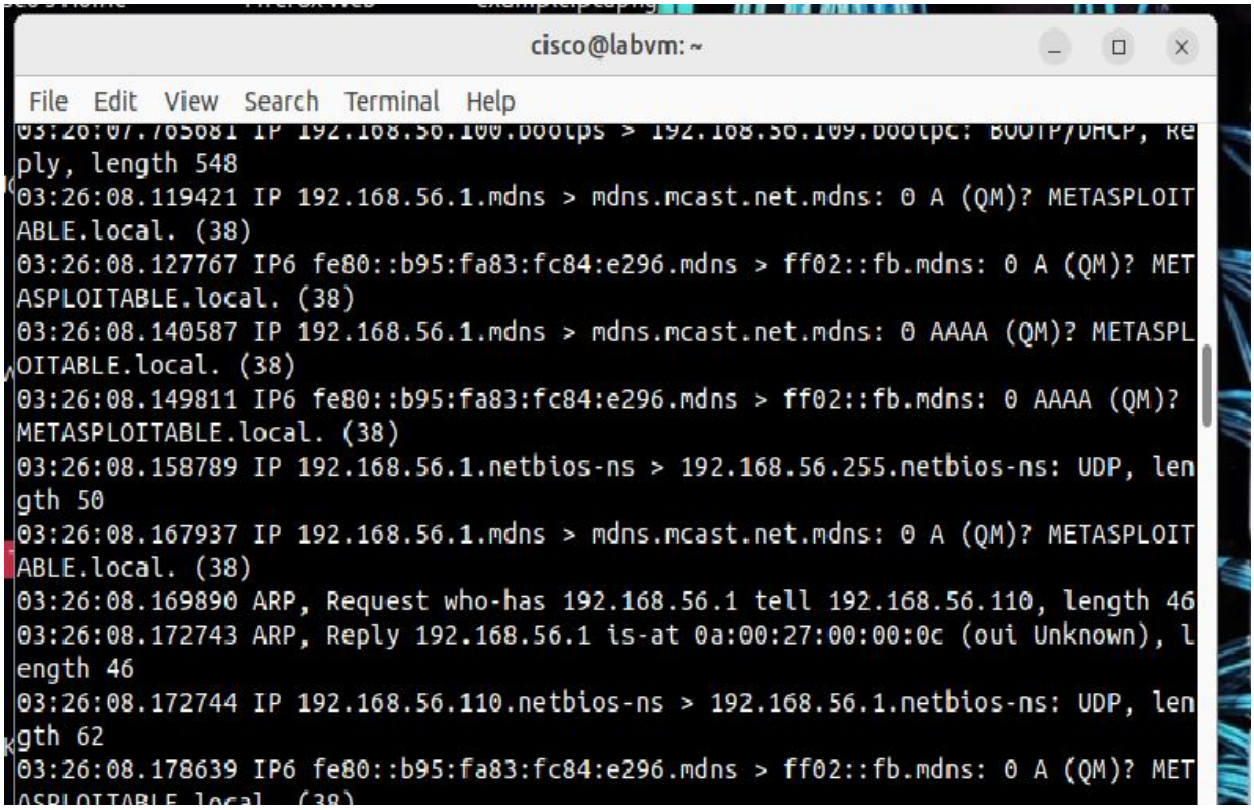
Aspect	TCP/IP Model	OSI Model
Development	Created as a practical, implementation-driven model for networking.	Developed as a conceptual reference model for standardization.
Layers	4 Layers: Application, Transport, Network, Link.	7 Layers: Application, Presentation, Session, Transport, Network, Data Link, Physical.
Focus	Emphasizes functionality and real-world protocols.	Provides a theoretical framework for interoperability.
Layer Details	Combines some OSI layers (e.g., no Presentation or Session layers).	Clearly separates all seven layers.
Protocol Dependence	Protocol-oriented (e.g., TCP, IP).	Protocol-agnostic, providing a flexible framework.

Correspondence Between TCP/IP and OSI Layers

TCP/IP Layer	Corresponding OSI Layers
Application	Application (7), Presentation (6), Session (5).
Transport	Transport (4).
Network	Network (3).
Link	Data Link (2), Physical (1).

Exercise 2: Capturing and Analyzing TCP Packets





TCP Handshake Process

The TCP handshake establishes a reliable connection between two devices. It involves three steps:

1. **SYN**: The client sends a synchronization request to initiate a connection.
2. **SYN-ACK**: The server acknowledges the request and sends its own synchronization.
3. **ACK**: The client acknowledges the server's SYN, completing the handshake.

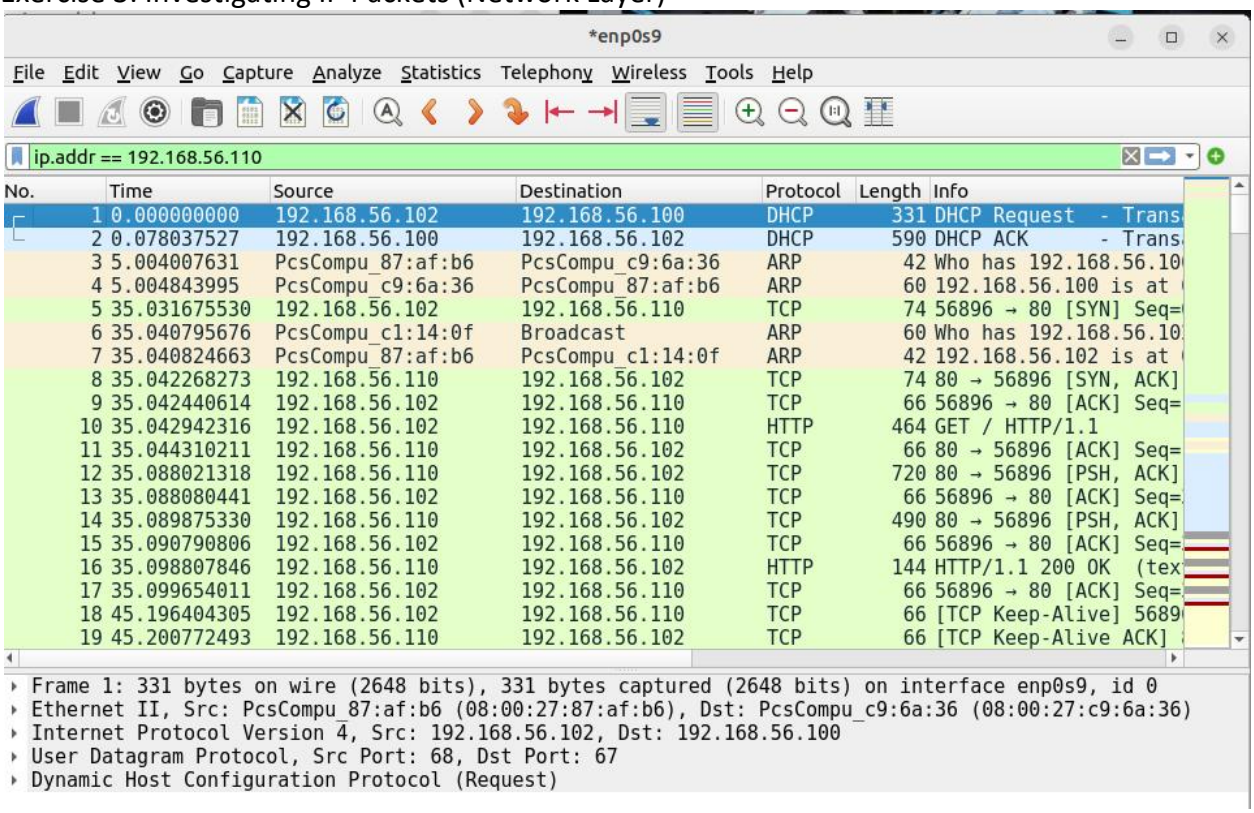
TCP Flags

- **SYN**: Initiates a connection.
- **ACK**: Acknowledges received data.
- **FIN**: Signals the end of a connection.
- **RST**: Resets a connection.
- **PSH**: Indicates immediate delivery of data.
- **URG**: Marks data as urgent.

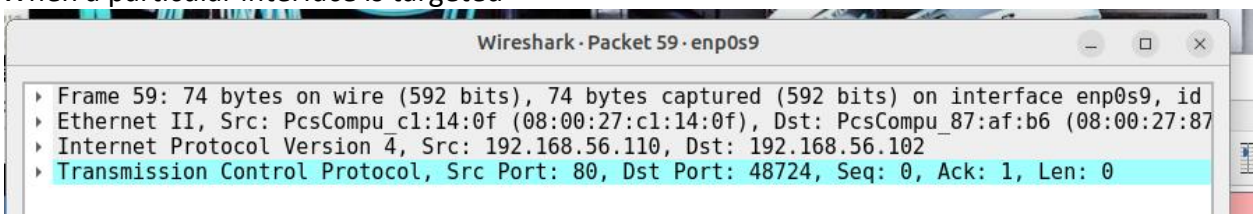
Reliability

TCP ensures reliability through acknowledgments, sequence numbers, retransmissions for lost packets, and ordered delivery.

Exercise 3: Investigating IP Packets (Network Layer)



When a particular interface is targeted



Fields in the IP Packet Header

When analyzing the **IP packet header** in Wireshark, you can see several important fields. Here's a breakdown of common fields and their significance:

1. **Version**:
   - **IPv4** (4) or **IPv6** (6).
   - **Significance**: Indicates the version of the Internet Protocol being used. IPv4 is more commonly seen in networks today, though IPv6 is gaining adoption.
2. **Header Length**:
   - The length of the IP header in 32-bit words.
   - **Significance**: Specifies how many 32-bit chunks are used by the header, which can vary depending on options.
3. **Total Length**:

- The total length of the IP packet, including both header and data.
- **Significance**: Helps determine the overall size of the packet. For IPv4, this is a 16-bit field, so it can range from 0 to 65535 bytes.

4. **Identification**:
   - A unique identifier assigned to each packet to help reassemble fragmented packets.
   - **Significance**: Used for fragment reassembly. When an IP packet is fragmented into multiple smaller packets, all fragments will have the same identification value.
5. **Flags**:
   - **Don't Fragment (DF)**, **More Fragments (MF)**.
   - **Significance**:
     - **DF**: Indicates the packet cannot be fragmented.
     - **MF**: Shows that the packet is part of a series of fragmented packets.
6. **Fragment Offset**:
   - This field is used if the packet is fragmented.
   - **Significance**: It helps in determining where a fragment belongs in the original packet.
7. **Time to Live (TTL)**:
   - The number of hops (routers) the packet can traverse before being discarded.
   - **Significance**: Prevents packets from endlessly circulating in the network. Each router decreases TTL by 1. If TTL reaches 0, the packet is discarded.
8. **Protocol**:
   - Indicates the upper-layer protocol used (e.g., TCP = 6, UDP = 17, ICMP = 1).
   - **Significance**: Helps the receiving system understand which protocol should be used to process the data (e.g., TCP, UDP, or ICMP).
9. **Header Checksum**:
   - A checksum for error-checking of the IP header.
   - **Significance**: Ensures the integrity of the header during transmission. If the checksum does not match on receipt, the packet is discarded.
10. **Source IP Address**:
    - The IP address of the sender (your system or any host sending the packet).
    - **Significance**: Identifies the origin of the packet and is used to route the reply back to the sender.
11. **Destination IP Address**:
    - The IP address of the receiver (the OWASP VM in this case).
    - **Significance**: Identifies the final destination of the packet.
12. **Options (if present)**:
    - Extra options, like routing, timestamp, etc.
    - **Significance**: Can be used for special routing or handling. This is rarely used in typical packets.

How Does IP Routing Work?

IP routing is the process of forwarding packets from the source to the destination across multiple network devices (routers). Each router in the path looks at the **Destination IP Address** and uses a **routing table** to forward the packet to the next hop (another router or the destination). The **TTL** field ensures that if the packet gets stuck in a routing loop, it is discarded after a set number of hops.

Steps in IP Routing:

1. **Source Device**: The source device (your system) creates the packet and sends it out with a destination IP (e.g., the IP address of the OWASP VM).
2. **Routing Decision**: Each router in the network examines the destination IP address.
    - The router looks up the destination IP in its routing table.
    - If a match is found, it forwards the packet to the next hop or the destination.
    - If there's no match, it drops the packet or forwards it to the default route.
3. **TTL Decreases**: As the packet passes through routers, the **TTL** value decreases. If TTL reaches 0, the packet is discarded.
4. **Destination Reached**: Eventually, the packet reaches the destination IP address (the OWASP VM in this case).

Are There Any Hops Between Your System and the OWASP VM?

To determine if there are any hops, you can use a traceroute **tool to see the route packets take to reach the OWASP VM. 1. Fields in the IP Packet Header**

When capturing IP packets using tools like Wireshark or tcpdump, the following key fields from the IP header can be observed:

1. **Source IP Address**
    - The IP address of the device sending the packet.
    - Significance: Identifies where the packet originated, enabling return communication.
2. **Destination IP Address**
    - The IP address of the target device receiving the packet.
    - Significance: Indicates where the packet is intended to go.
3. **Version**
    - Indicates the IP protocol version (IPv4 or IPv6).
    - Significance: Ensures devices interpret the packet correctly based on the protocol.
4. **Header Length (IHL)**
    - Specifies the length of the IP header in 32-bit words.

    - Significance: Helps identify where the payload begins.
5. **Total Length**
    - The entire packet size, including header and data, in bytes.
    - Significance: Ensures the receiver can reassemble the entire packet properly.
6. **Identification**
    - A unique identifier for each packet.
    - Significance: Useful for reassembling fragmented packets.
7. **Flags**
    - Controls and identifies fragmentation behavior.
    - Significance: Ensures packets are fragmented and reassembled as needed.
8. **Fragment Offset**
    - Indicates the position of a fragment in the original packet.
    - Significance: Helps in reconstructing fragmented packets.
9. **Time to Live (TTL)**
    - Limits the packet's lifespan in terms of hops.
    - Significance: Prevents packets from looping indefinitely in the network.
10. **Protocol**
    - Identifies the protocol (e.g., TCP, UDP, ICMP) encapsulated in the packet.
    - Significance: Directs packets to the appropriate handler at the destination.
11. **Header Checksum**
    - A checksum of the header to detect errors.
    - Significance: Ensures the integrity of the header during transmission.
12. **Options (if any)**
    - Provides additional functionality (e.g., security, routing).
    - Significance: Extends the capabilities of the IP header.

---

2. Significance of Each Field

- The fields collectively ensure that packets are delivered to the correct destination, are intact, and can be processed by the receiving system.
- Fields like TTL manage network efficiency, while checksum validates header integrity, ensuring robust communication.

---

3. IP Routing in This Scenario

IP routing involves directing packets from the source to the destination IP address across potentially multiple network hops. In this context:

Steps:

1. **Packet Generation:** Your system creates and sends packets to the OWASP VM's IP address.
2. **Local Routing Table:** The operating system checks its routing table to determine the next hop (gateway or directly connected).
3. **Intermediate Hops (if any):**
    - If the OWASP VM is on the same subnet, packets are sent directly.
    - If the VM is on a different subnet, packets are sent to the default gateway, which routes them to the target subnet.

Hops:

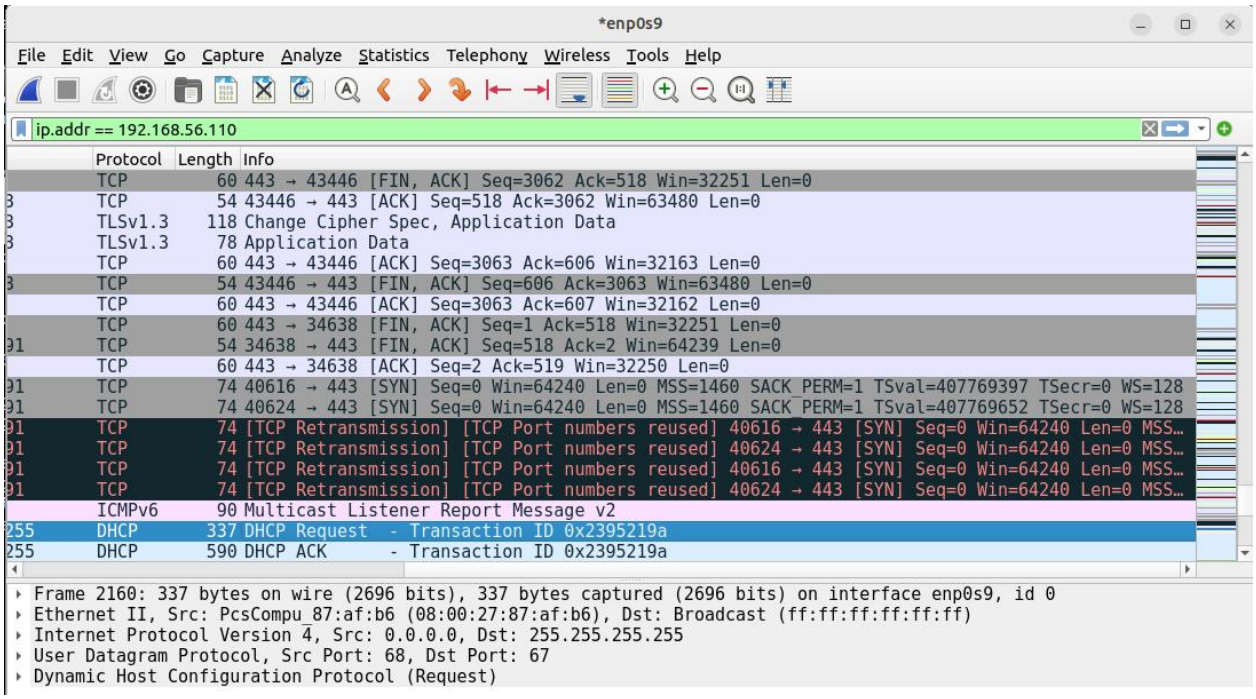
- **Same Subnet:** There will be no intermediate hops; the communication occurs directly.
- **Different Subnet:** The packet traverses routers, which decrement the TTL by one at each hop and forward it based on the destination.

To check the exact routing:

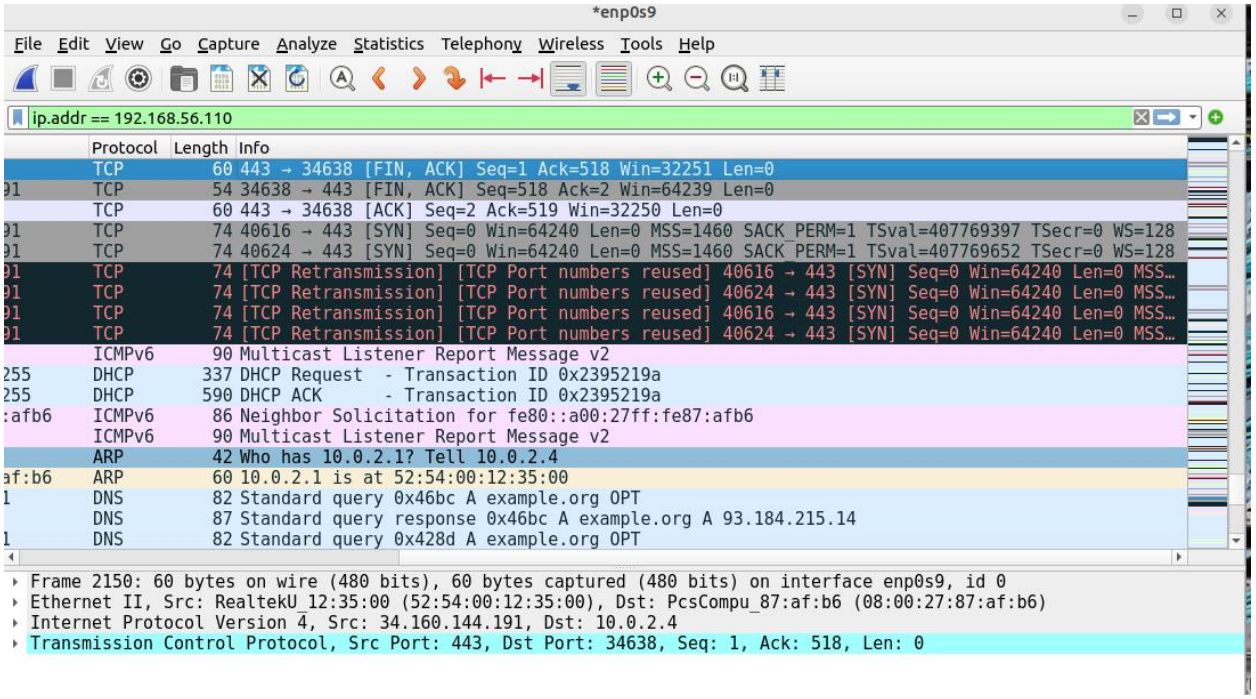
- Use traceroute <OWASP_IP> on Linux or tracert <OWASP_IP> on Windows to see the intermediate hops.

Exercise 5: Error Handling in TCP/IP Transmission

Below is the screenshot when interface enp0s9, where the data is transmitted was stopped. You can see series of SYN without response



Below is the response immediately I enabled the the interface on which I am capturing the data



Please NOTE that all my explanation below can be verified by the two screenshot above where I intententionally shutdown the interface enp0s9 and captured the reaction on the wireshark

What Happens When Packets Are Dropped or Delayed?

1. **Packet Dropped**:

- If a packet is dropped (lost in transit), the receiver does not acknowledge it.
- TCP detects the loss through mechanisms like **timeout** or duplicate acknowledgments.
- Loss can occur due to network congestion, hardware failure, or misconfigurations.

2. **Packet Delayed**:

- If a packet is delayed, it may arrive out of order.
- TCP reorders packets upon receipt using their sequence numbers.
- Excessive delays can trigger retransmissions if TCP's timeout threshold is exceeded.

3. **Impact on Communication**:

- Increased **latency** (due to retransmissions).
- Possible **throughput reduction** if TCP's congestion control reduces the transmission rate.

How Does TCP Ensure Data Reliability in the Presence of Errors?

TCP uses several mechanisms to ensure data reliability:

1. **Acknowledgments (ACKs)**:

- For every packet (or segment) received, the receiver sends an acknowledgment.
- If an acknowledgment is not received within a certain timeframe, TCP assumes the packet was lost and retransmits it.

2. **Sequence Numbers**:

- Each byte of data in a TCP segment is assigned a unique sequence number.
- Sequence numbers ensure that the receiver can reassemble packets in the correct order, even if they arrive out of sequence.

3. **Checksums**:

- TCP includes a checksum in every segment to detect corruption during transmission.
- If a checksum mismatch occurs, the packet is discarded, and retransmission is initiated.

4. **Retransmissions**:

- When packet loss is detected (e.g., through timeout or duplicate ACKs), TCP retransmits the missing data.
- Retransmission is key to recovering from errors and ensuring reliable delivery.

5. **Flow Control**:
- TCP uses a **sliding window protocol** to manage how much data can be sent before receiving an acknowledgment.
- This prevents overwhelming the receiver.
6. **Congestion Control**:
- Algorithms like **TCP Reno** or **TCP Cubic** adjust the sending rate based on network conditions to minimize packet loss due to congestion.

How Do Retransmissions and Sequence Numbers Work in TCP?

1. **Sequence Numbers**:
- Each byte in the data stream is assigned a sequence number.
- The first byte's sequence number is randomly chosen during the connection handshake.
- The receiver uses these numbers to:
- Identify missing or out-of-order packets.
- Acknowledge the next expected byte.
2. **Retransmissions**:
- **Timeout-Based Retransmission**:
- If an acknowledgment is not received before the retransmission timeout (RTO) expires, the sender retransmits the unacknowledged packet.
- **Duplicate ACK-Based Retransmission**:
- If the receiver detects a missing packet, it sends duplicate ACKs for the last successfully received sequence number.
- When the sender receives three duplicate ACKs, it performs **fast retransmission** without waiting for the timeout.
3. **Example Workflow**:
- Sender transmits packets: Sequence numbers 1, 2, 3.
- Packet 2 is dropped.
- Receiver sends ACK for 1 repeatedly (duplicate ACKs).
- After three duplicate ACKs, the sender retransmits packet 2.
- Once packet 2 is received, the receiver acknowledges up to 3.
4. **Out-of-Order Handling**:

- If packets arrive out of order, the receiver buffers them temporarily and waits for the missing packet to reassemble the stream.

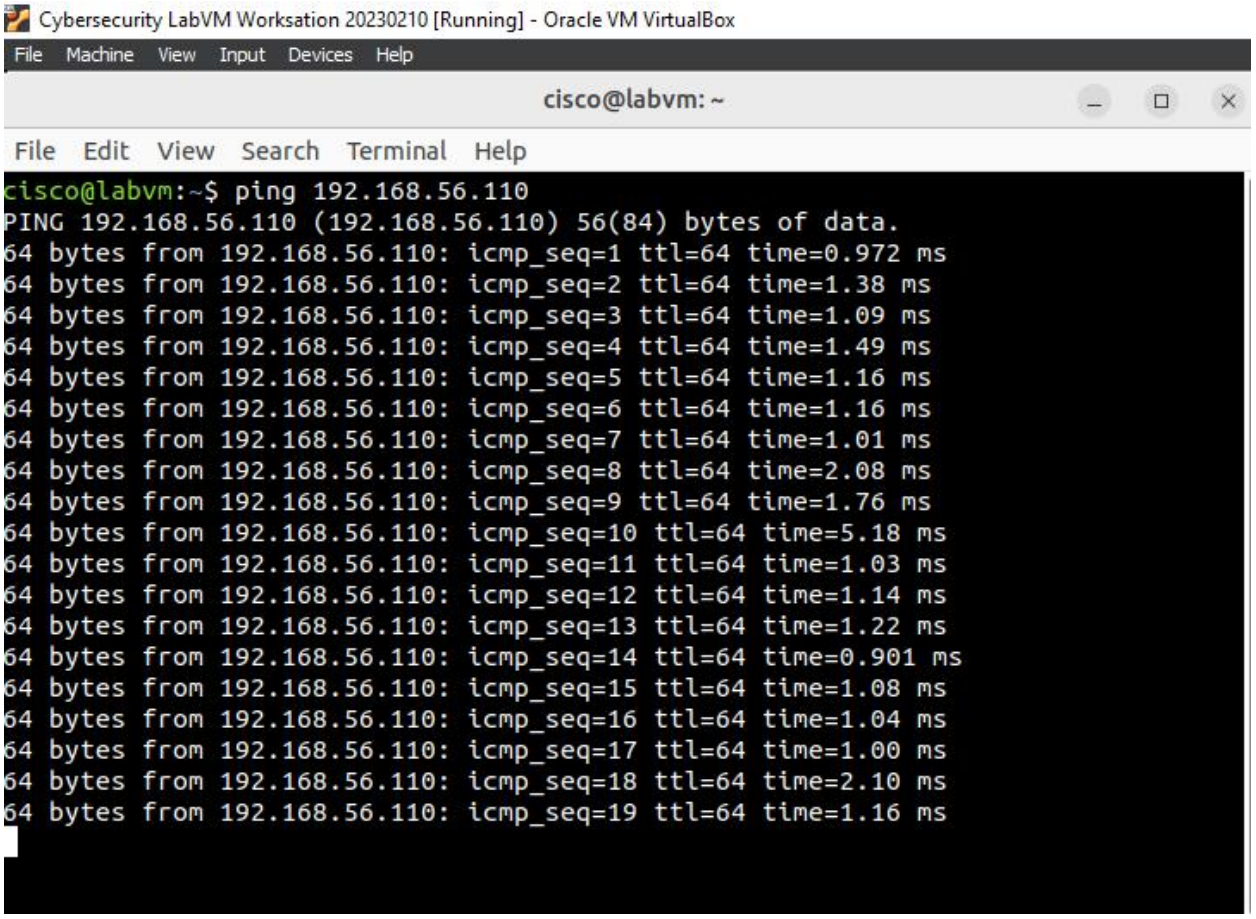
Summary of TCP Reliability Mechanisms

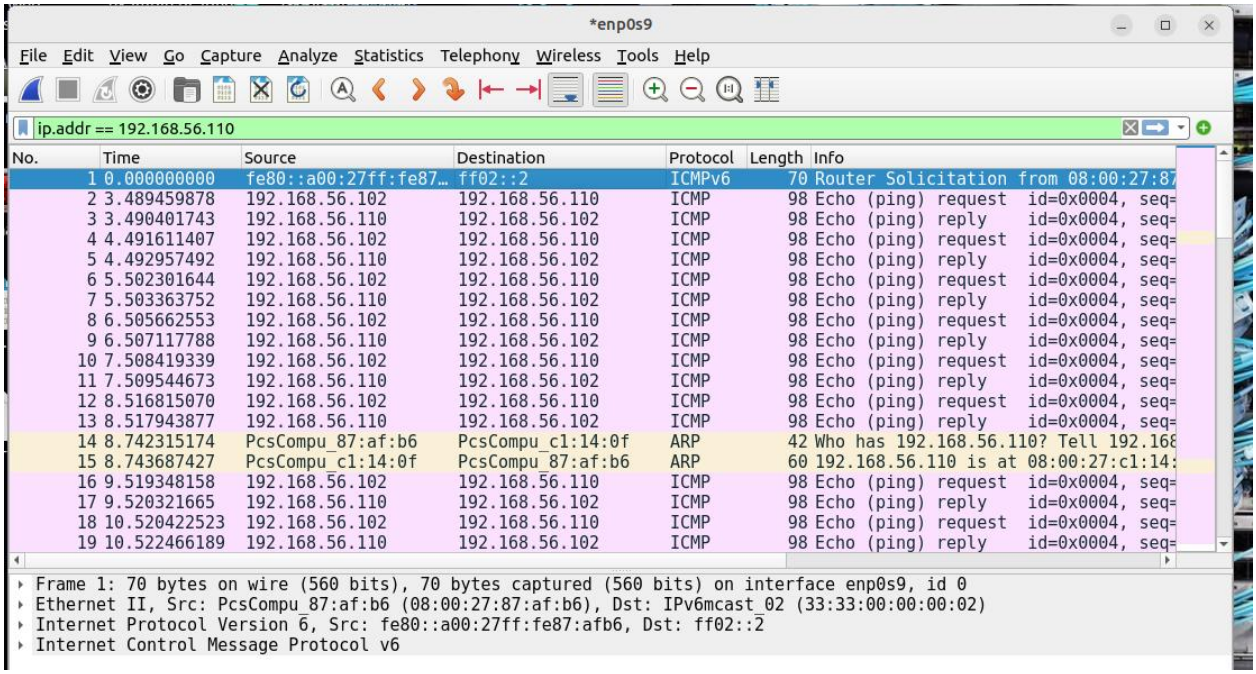
- **Acknowledgments** track received data.
- **Sequence numbers** ensure proper ordering.
- **Retransmissions** recover lost data.
- **Checksums** verify integrity.
- **Flow and congestion control** optimize performance under varying network conditions.

These features together make TCP a robust protocol for reliable communication.

Exercise 6: ICMP and Ping Inspection (Network Layer)

Command: ping and use Wireshark to capture ICMP packets with the filter icmp.





Key Fields in an ICMP Packet

ICMP (Internet Control Message Protocol) packets consist of specific fields that convey diagnostic and error messages. The key fields include:

1. **Type**:
- Indicates the purpose of the ICMP message (e.g., 0 for Echo Reply, 8 for Echo Request, 3 for Destination Unreachable).
- Each type serves a distinct diagnostic or error reporting function.
2. **Code**:
- Provides additional granularity to the Type field, specifying the reason for the message.
- For example, in a "Destination Unreachable" message (Type 3), codes differentiate between "Network Unreachable," "Host Unreachable," etc.
3. **Checksum**:
- Used for error checking the ICMP header and payload to ensure data integrity.
- If the checksum doesn't match, the packet is discarded.
4. **Identifier** (for Echo Request/Reply):
- Used to match requests and replies, often to differentiate between multiple pings.
5. **Sequence Number** (for Echo Request/Reply):
- Helps track the order of sent packets, useful in determining packet loss or latency.
6. **Data Section**:
- Contains additional information or payload data, such as a timestamp for round-trip time (RTT) calculations.

How Does ICMP Assist in Diagnosing Network Connectivity Issues?

ICMP is integral to network diagnostics, as it provides tools for identifying and addressing issues. Key uses include:

1. **Ping (Echo Request/Reply)**:
- Sends ICMP Echo Request packets to a target host, which replies with Echo Reply packets.
- Measures round-trip time (RTT) and detects packet loss, indicating latency or connectivity issues.
2. **Traceroute**:

- Uses ICMP Time Exceeded messages to map the path packets take to a destination.
- Identifies intermediate routers, potential bottlenecks, or points of failure.

3. **Error Reporting**:

- ICMP communicates network errors back to the sender:

- **Destination Unreachable**: Indicates routing issues or firewall blocks.
- **Time Exceeded**: Signals that a packet's Time-to-Live (TTL) expired, often revealing routing loops.
- **Redirect Message**: Suggests a better route for the sender.

4. **Network Congestion Insights**:

- Tools using ICMP can reveal packet drops or delays, often associated with congestion or misconfigurations.

Significance of TTL in ICMP Packets and General IP Packets

1. **TTL (Time-to-Live)**:

- A field in the IP header that specifies the maximum number of hops (routers) a packet can traverse.
- Decremented by one at each hop; if it reaches zero, the router discards the packet and sends an ICMP Time Exceeded message back to the sender.

2. **Role in ICMP**:

- **Traceroute**: Relies on TTL. By incrementally increasing the TTL value in packets, it maps the route to a destination.
- **Loop Prevention**: ICMP Time Exceeded messages triggered by TTL expiration help identify and resolve routing loops.

3. **Role in General IP Packets**:

- Prevents infinite loops in the network by ensuring packets are discarded after a finite number of hops.
- Helps maintain network stability by discarding misrouted or delayed packets.

4. **Diagnostics with TTL**:

- A high TTL indicates a direct or optimized route, while a low TTL signals more hops or potential inefficiencies.
- TTL values can be used to approximate the distance (in hops) to a destination.

Exercise 7: Analyzing UDP Packets (Transport)

```
(socket: operation not permitted)
cisco@labvm:~$ sudo tcpdump -i enp0s9 udp and host 192.168.56.110
[sudo] password for cisco:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s9, link-type EN10MB (Ethernet), snapshot length 262144 bytes
09:27:00.302799 IP 192.168.56.110.netbios-ns > 192.168.56.255.netbios-ns: UDP, length 50
09:27:02.304126 IP 192.168.56.110.netbios-ns > 192.168.56.255.netbios-ns: UDP, length 50
09:27:02.304126 IP 192.168.56.110.netbios-ns > 192.168.56.255.netbios-ns: UDP, length 50
09:27:04.302298 IP 192.168.56.110.netbios-ns > 192.168.56.255.netbios-ns: UDP, length 50
09:27:15.301570 IP 192.168.56.110.netbios-dgm > 192.168.56.255.netbios-dgm: UDP, length 197
09:27:17.301120 IP 192.168.56.110.netbios-dgm > 192.168.56.255.netbios-dgm: UDP, length 197
09:27:19.300249 IP 192.168.56.110.netbios-dgm > 192.168.56.255.netbios-dgm: UDP, length 197
09:27:21.300309 IP 192.168.56.110.netbios-dgm > 192.168.56.255.netbios-dgm: UDP, length 197
09:27:23.300757 IP 192.168.56.110.netbios-dgm > 192.168.56.255.netbios-dgm: UDP, length 197
09:27:23.301469 IP 192.168.56.110.netbios-ns > 192.168.56.255.netbios-ns: UDP, length 68
09:27:25.336234 IP 192.168.56.110.netbios-ns > 192.168.56.255.netbios-ns: UDP, length 68
09:27:25.336955 IP 192.168.56.110.netbios-ns > 192.168.56.255.netbios-ns: UDP, length 68
09:27:27.330027 IP 192.168.56.110.netbios-ns > 192.168.56.255.netbios-ns: UDP, length 68
09:27:27.330753 IP 192.168.56.110.netbios-ns > 192.168.56.255.netbios-ns: UDP, length 68
09:27:29.330059 IP 192.168.56.110.netbios-ns > 192.168.56.255.netbios-ns: UDP, length 68
09:27:29.330909 IP 192.168.56.110.netbios-ns > 192.168.56.255.netbios-ns: UDP, length 68
09:27:31.330543 IP 192.168.56.110.netbios-ns > 192.168.56.255.netbios-ns: UDP, length 68
09:27:31.331237 IP 192.168.56.110.netbios-dgm > 192.168.56.255.netbios-dgm: UDP, length 185
09:27:31.331983 IP 192.168.56.110.netbios-dgm > 192.168.56.255.netbios-dgm: UDP, length 244
09:27:31.332793 IP 192.168.56.110.netbios-dgm > 192.168.56.255.netbios-dgm: UDP, length 215
09:31:31.504354 IP 192.168.56.110.netbios-dgm > 192.168.56.255.netbios-dgm: UDP, length 244
09:31:31.505195 IP 192.168.56.110.netbios-dgm > 192.168.56.255.netbios-dgm: UDP, length 215
09:33:41.738529 IP 192.168.56.110.netbios-dgm > 192.168.56.1.netbios-dgm: UDP, length 189
09:33:43.983342 IP 192.168.56.110.netbios-ns > 192.168.56.1.netbios-ns: UDP, length 62
09:33:45.085119 IP 192.168.56.110.netbios-ns > 192.168.56.1.netbios-ns: UDP, length 62
09:33:45.091667 IP 192.168.56.110.netbios-ns > 192.168.56.1.netbios-ns: UDP, length 62
09:33:45.097811 IP 192.168.56.110.netbios-dgm > 192.168.56.1.netbios-dgm: UDP, length 189
09:33:47.351330 IP 192.168.56.110.netbios-dgm > 192.168.56.1.netbios-dgm: UDP, length 189
```

Comparison of UDP and TCP

1. Packet Structure Differences:

Aspect	UDP (User Datagram Protocol)	TCP (Transmission Control Protocol)
Header Size	8 bytes	20-60 bytes
Fields	Contains fewer fields:	Contains more fields for reliability:
	- Source Port	- Source Port
	- Destination Port	- Destination Port
	- Length	- Sequence Number
	- Checksum	- Acknowledgment Number
		- Flags (e.g., SYN, ACK)
		- Window Size, Urgent Pointer
Complexity	Simple	More complex

2. Behavior Differences:

Aspect	UDP	TCP
Connection	Connectionless	Connection-oriented
Reliability	No guarantees (best-effort)	Reliable delivery (acknowledgments, retransmissions)
Flow Control	None	Implements flow control (sliding window)
Error Checking	Checksum only	Checksum, retransmissions, error recovery
Speed	Faster (minimal overhead)	Slower due to added reliability
Data Integrity	No sequence guarantee	Ensures ordered and complete delivery

Why Does UDP Not Ensure Reliability?

- **No Handshake:** UDP doesn't establish a connection between sender and receiver, skipping the overhead of connection setup and teardown.
- **No Acknowledgments:** It doesn't require the receiver to send acknowledgments for received packets.
- **No Retransmissions:** Lost or corrupted packets are not automatically resent.
- **Stateless Protocol:** Each datagram is independent, with no tracking of sent or received data.

UDP prioritizes simplicity and low latency, which naturally sacrifices reliability.

Scenarios Where UDP is Preferred Over TCP

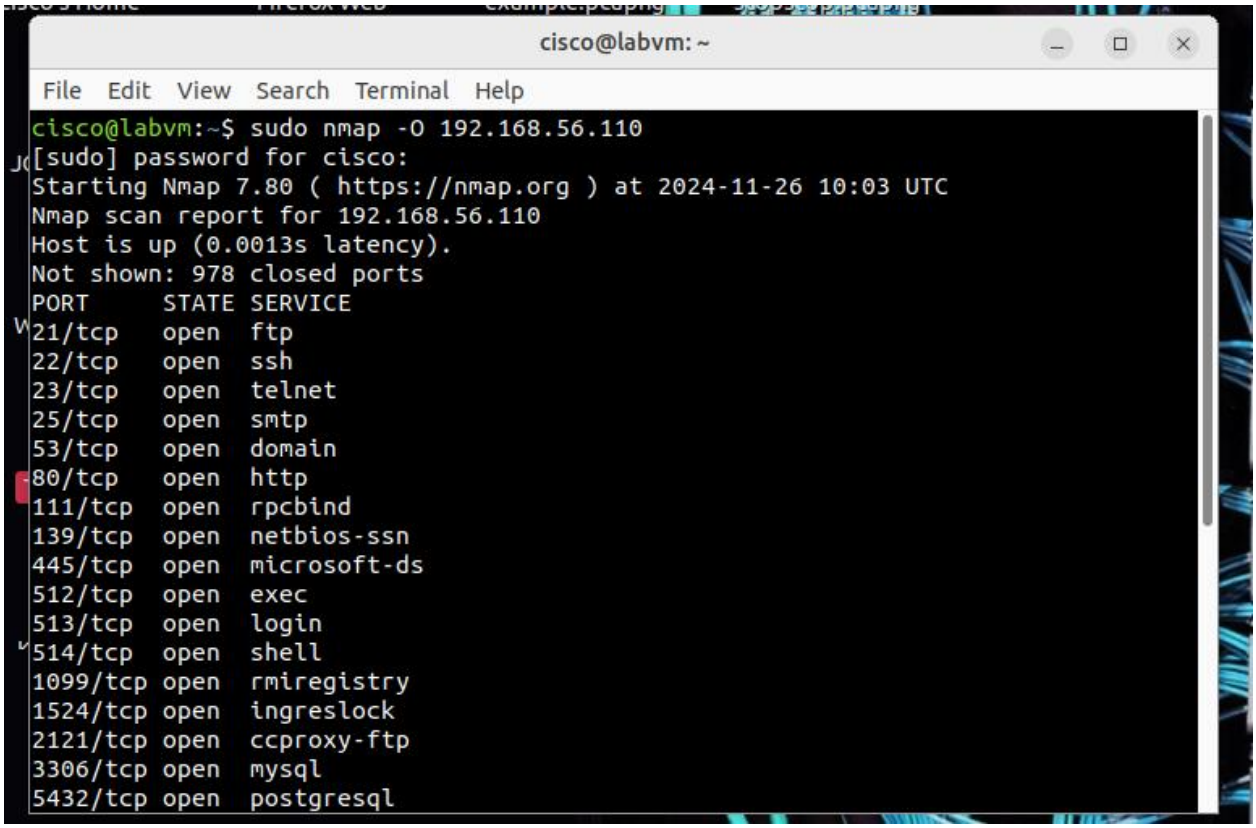
- **Real-Time Applications:** Audio/video streaming, VoIP, gaming, and conferencing prioritize low latency over perfect data reliability.
- **Broadcast/Multicast:** Scenarios like DNS queries, DHCP, and IPTV where broadcasting to multiple receivers is efficient.
- **Simple Query-Response Protocols:** Applications like SNMP and TFTP that don't require high reliability or order.
- **Fast Transmission Needs:** UDP is used in applications where speed and minimal overhead are critical, like financial systems or IoT.

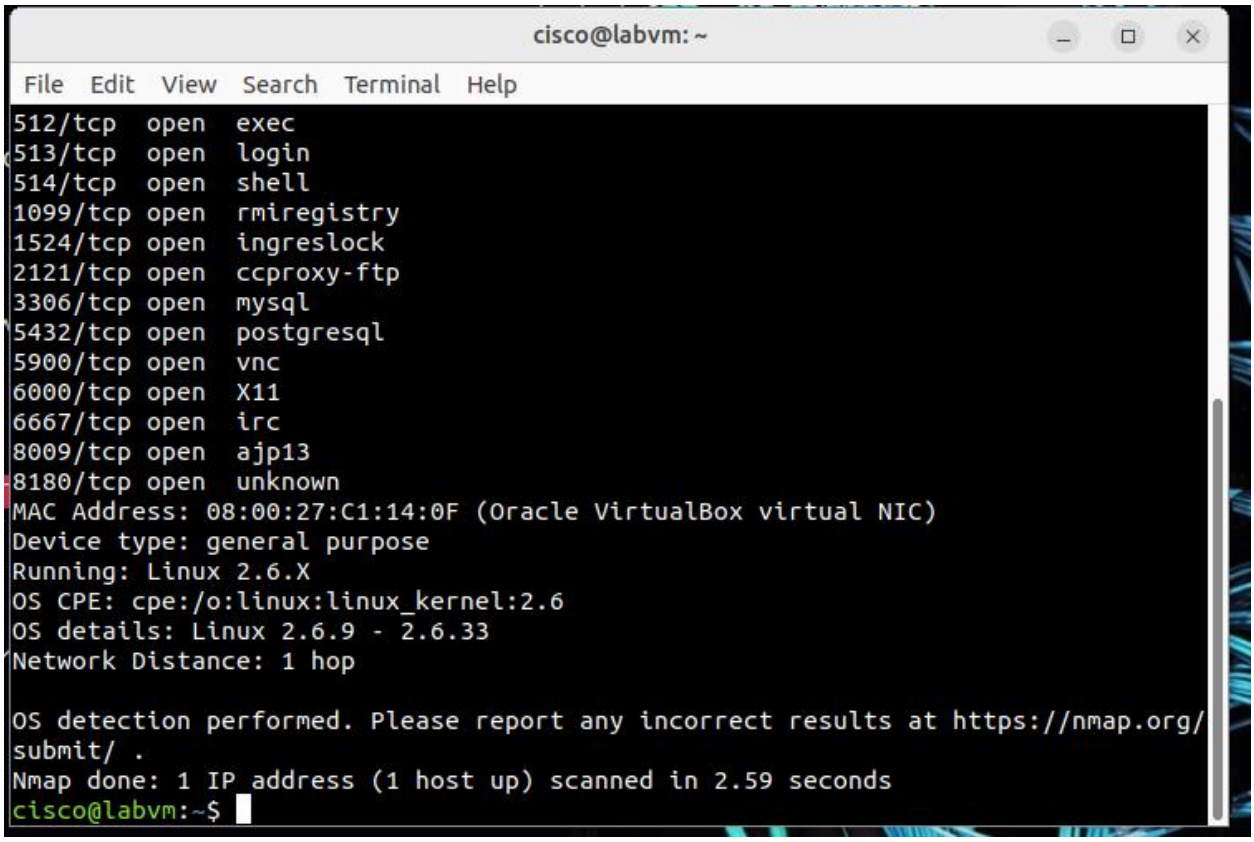
How UDP Manages Data Transmission Without Acknowledgments or Retransmissions

1. **Independent Packets:** Each UDP packet (datagram) is self-contained, carrying all necessary addressing and data.
2. **Minimal Overhead:** The lack of sequence numbers and acknowledgments reduces protocol complexity and transmission time.
3. **Error Checking:** A checksum is used to detect errors, but recovery is left to the application.
4. **Application Responsibility:** The application layer implements its own reliability mechanisms, if needed, such as retransmissions or packet ordering.

In short, UDP shifts the responsibility for handling reliability to the application, making it lightweight but suitable for specific use cases.

Exercise 8: OS Detection via Nmap (Network and Transport Layers)





How Nmap Detects the OS Based on Captured Packets

Nmap (Network Mapper) uses **OS fingerprinting** to identify the operating system of a remote device. This process involves analyzing the responses to specifically crafted packets. There are two types of OS detection methods in Nmap:

1. **Active OS Detection (TCP/IP Stack Fingerprinting):**

- Nmap sends a series of probes (e.g., TCP, UDP, ICMP packets) to the target.
- It observes the responses to these probes, analyzing parameters such as TTL, window size, and other low-level protocol details.
- By comparing these responses to a database of known OS fingerprints, Nmap deduces the target OS.

2. **Passive OS Detection:**

- Nmap captures existing traffic without actively probing.
- It analyzes packet characteristics from this traffic to determine the OS.

Packet Characteristics Used for OS Detection

1. **TTL (Time to Live):**
- The initial TTL value set by the OS before decrementation gives clues about the OS. For example:
- Linux: 64
- Windows: 128
- Cisco: 255
2. **Window Size:**
- The size of the TCP window (buffer) used by the OS varies and is often unique. Examples:
- Windows XP: 65535
- Linux 2.4 kernel: 5840
3. **TCP Options and Flags:**
- Sequence of TCP options (e.g., MSS, SACK, timestamps) and their order can reveal the OS.
- Specific combinations of flags in responses (SYN-ACK, RST) vary across operating systems.
4. **ICMP Responses:**
- Differences in ICMP type and code, rate-limiting behavior, and formatting in response to specific probes.
5. **OS-Specific Quirks:**
- Differences in how various OSes handle malformed or unusual packets, such as:
- Responses to FIN or NULL scans.
- Handling of overlapping fragments.
6. **Other Characteristics:**
- Presence or absence of certain protocols or headers (e.g., ICMP timestamps, IP ID generation patterns).

Importance of OS Detection in Network Analysis and Vulnerability Assessment

1. **Identifying Potential Vulnerabilities:**
- Each OS has unique vulnerabilities, and knowing the OS helps prioritize vulnerability scans and exploit testing.
- Example: If a system runs Windows Server 2008, you might check for unpatched SMB vulnerabilities like EternalBlue.

2. **Inventory and Asset Management:**

- OS detection aids in maintaining an accurate inventory of devices, their operating systems, and versions.

3. **Incident Response and Threat Analysis:**

- Understanding the OS can help assess the impact of a security breach and guide remediation efforts.

4. **Assessing Security Posture:**

- OS detection reveals devices that may be running outdated or unsupported systems, exposing the network to unnecessary risk.

5. **Red Team and Penetration Testing:**

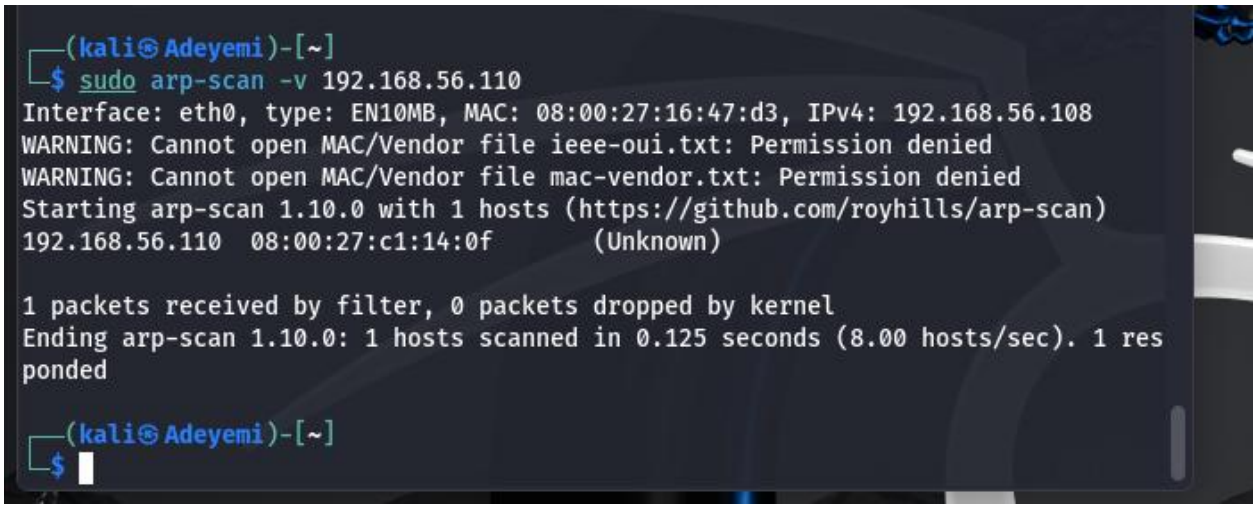
- Identifying the OS helps attackers simulate realistic attack scenarios, improving the quality of penetration tests.

6. **Defensive Measures:**

- Knowing the OS enables system administrators to configure OS-specific defenses, such as disabling unused services or applying OS-specific patches.

As a matter of fact, OS detection provides critical insights into a network's structure and potential vulnerabilities, forming the foundation for proactive defense and targeted security assessments.

Exercise 9: Analyzing ARP Traffic (Link Layer)



Information Gathered from ARP Packets

Address Resolution Protocol (ARP) packets are used to map IP addresses to MAC (Media Access Control) addresses within a local network. The following information can be gathered from ARP packets:

1. **MAC Addresses**: ARP packets include the hardware (MAC) addresses of the source and destination systems. For example:
- The source MAC address (the sender's address).
- The target MAC address (set to zero or unknown in ARP requests but provided in ARP replies).
2. **IP Addresses**: ARP packets also include the corresponding IP addresses for the source and target devices:
- The source IP address (the IP of the sender).
- The target IP address (the IP of the intended recipient).
3. **Operation Type**: ARP packets specify whether the operation is a request (asking for a MAC address corresponding to an IP) or a reply (providing a MAC address).
4. **Network Hardware Type and Protocol Type**: ARP packets indicate the type of link-layer protocol (usually Ethernet) and the network-layer protocol (usually IPv4).

ARP Protocol Function at the Link Layer

ARP operates at the **link layer** (Layer 2 of the OSI model) but facilitates communication between the link and network layers. Here’s how it functions:

1. **Broadcast Request**:
- When a device wants to communicate with another device on the same local network but doesn't know its MAC address, it sends an ARP request as a **broadcast frame** (destination MAC is FF:FF:FF:FF:FF:FF).
- The ARP request includes the IP address of the target device whose MAC address is being queried.
2. **Unicast Reply**:
- The device with the matching IP address sends an ARP reply **directly to the requesting device** (a unicast response).
- The reply includes the MAC address associated with the IP address in the ARP request.
3. **Cache Updates**:

- Both the requester and the responder update their ARP tables (a local cache) with the MAC-IP mapping, reducing the need for future ARP requests for the same target.

Role of ARP in Communication Between My System and the OWASPVM

In a scenario where your system communicates with an **OWASPVM** (a virtual machine running security tools):

1. **Discovery**: When my system attempts to send packets to the OWASPVM, it first checks its ARP table to see if it already knows the MAC address of the VM's IP address. If it doesn't:
- It sends an ARP request to discover the MAC address associated with the OWASPVM's IP.
2. **Mapping**: The OWASPVM responds with its MAC address. This allows my system to map the OWASPVM's IP address to its MAC address.
3. **Direct Communication**: After learning the MAC address, my system can encapsulate data in Ethernet frames with the correct destination MAC address, ensuring that the packets are delivered to the OWASPVM.
4. **Local Communication**: Since ARP operates only within a local network, my system and the OWASPVM must be on the same subnet or within a virtual network bridged to the host machine. Otherwise, a gateway/router must perform ARP resolution on behalf of my system.

Exercise 10: Troubleshooting Network Connectivity Using TCP/IP Knowledge

Simulated Issue: Incorrect Subnet Mask

Issue Simulated:
A device was configured with an incorrect subnet mask. Instead of the correct subnet mask 255.255.255.0, the device was set to 255.255.0.0. This misconfiguration caused communication failures with devices outside the incorrectly defined subnet.

Effects on Network Communication

- The device believed that devices outside its actual subnet were still part of its local network. As a result:

- It attempted direct communication without forwarding packets to the configured gateway (router).
- Since the intended recipient devices were not in the same local network, these packets were dropped, causing communication failures.

Diagnosis and Resolution Using TCP/IP Layers and Packet Inspection

1. **Symptoms Observed**:

- The device could communicate with some hosts but failed to reach others, especially devices outside its local network or in other subnets.
- Ping commands to external IPs failed, showing "Destination Host Unreachable" or no response at all.

2. **Troubleshooting Steps**:

- **Layer 1 (Physical Layer)**: Verified physical connectivity and link status. No issues found.
- **Layer 2 (Data Link Layer)**: Used tools like arp -a to verify that ARP resolution was successful for local devices but not for external ones.
- **Layer 3 (Network Layer)**:
- Checked the IP configuration using ipconfig (Windows) or ifconfig/ip addr (Linux). Found an incorrect subnet mask.
- Used traceroute to trace the packet path to an unreachable external device. The trace indicated that packets were not reaching the gateway.
- **Packet Inspection**:
- Captured network traffic using Wireshark.
- Observed ARP requests and ICMP packets sent to external IPs directly, bypassing the gateway. This confirmed that the device was incorrectly assuming those IPs were in the same subnet.

3. **Resolution**:

- Corrected the subnet mask to 255.255.255.0 using network configuration tools or manual editing of the network interface settings.
- Tested connectivity again. Ping and traceroute to external devices were now successful, confirming the issue was resolved.

Tools and Techniques for Real-World Network Troubleshooting

1. **Basic Tools**:
- **ping**: Test connectivity to a specific IP address and measure response times.
- **traceroute/tracert**: Trace the path packets take to a destination.
- **arp**: View and clear ARP cache to diagnose Layer 2 issues.
- **ipconfig/ifconfig/ip**: Verify IP configuration and network interfaces.
2. **Advanced Tools**:
- **Wireshark**: Inspect packets in detail for ARP, ICMP, TCP, or UDP issues.
- **Netcat**: Test connectivity and simulate client-server communication.
- **nmap**: Scan networks to identify hosts, open ports, and services.
3. **Systematic Techniques**:
- **Layered Approach**: Start at the physical layer and systematically troubleshoot upwards (Layer 1 to Layer 4 and beyond).
- **Packet Analysis**: Use packet captures to identify anomalies in ARP requests, routing, or protocol behavior.
- **Log Review**: Check system logs, router logs, and firewall logs for errors or dropped packets.
- **Simulation**: Recreate the issue in a test environment to understand its root cause.
4. **Recommendations**:
- Maintain proper documentation of IP configurations and subnetting rules.
- Use monitoring tools like Nagios, SolarWinds, or PRTG for proactive network management.
- Implement VLANs or routing configurations carefully to avoid subnet conflicts.