Lab 1: Understanding Network Layers and TCP/IP Model using OWASP Broken Web Application IP

Objective:

In this lab, you will explore the OSI and TCP/IP models using the IP address of the OWASP Broken Web Application virtual machine. This practical exercise will help you understand how data is transmitted across networks, the key protocols involved, and how to use essential network commands for troubleshooting and analysis.

Learning Outcomes:

By the end of this lab, students will:

- Understand the OSI and TCP/IP models and their functions.
- Use essential network tools to explore and analyse network traffic.
- Apply knowledge of TCP/IP layers in real-world scenarios using OWASP Broken

Web Application IP.

Materials Needed:

- Linux-based system (Kali, Ubuntu, etc.)
- OWASP Broken Web Application VM (running and reachable on your network)
- Terminal access with networking tools
- IP address of the OWASP Broken Web Application (e.g., 192.168.X.X)

Lab Exercises:

Exercise 1: Understanding OSI and TCP/IP Models

1. Task: Review the OSI and TCP/IP models and their layers.

What is the OSI Model?

OSI stands for Open Systems Interconnection model. It is a conceptual framework used to understand and standardize how different networking systems communicate with each other. Developed by ISO (International Organization for Standardization).

It has 7 layers, each with specific functions to support communication over a network. Think of it like a layered cake where each layer depends on the one below it and supports the one above it.

OSI Model Layers (7 Layers - from bottom to top):

Physical: Transmits raw bits over the physical medium (cables, signals, voltages).

Data Link: Responsible for node-to-node communication, MAC addressing, and error detection (frames).

Network: Handles logical addressing and routing (IP addressing, packets).

Transport: Provides reliable data transfer, error recovery, and flow control (segments).

Session: Manages, establishes, and terminates sessions between applications.

Presentation: Translates, encrypts, and compresses data for the application layer.

Application: Closest to the user; provides network services to applications (HTTP, FTP, DNS, etc.).

What is the TCP/IP Model?

TCP/IP stands for Transmission Control Protocol / Internet Protocol: It is the real-world protocol suite that governs how data is sent over the Internet and most networks today. Developed by the U.S. Department of Défense in the 1970s. It has 4 layers, and is more practical and widely implemented than the OSI model.

TCP/IP Model Layers (4 Layers - from bottom to top):

Link: Combines OSI's Physical and Data Link layers; handles local network communication.

Internet: Corresponds to OSI's Network layer; manages IP addressing and routing.

Transport: Same as OSI's Transport layer; ensures reliable communication (TCP/UDP).

Application: Merges OSI’s Application, Presentation, and Session layers; provides application services.

2. **Question:**

List the OSI model layers and describe the function of each

	OSI Layer	Function
7	Application	Interfaces with end-user software and provides network services (e.g., HTTP, SMTP).
6	Presentation	Translates data formats, encrypts, and compresses data (e.g., SSL, JPEG).
5	Session	Manages sessions or connections between applications (e.g., API sessions).
4	Transport	Ensures reliable data transfer with error checking and flow control (e.g., TCP/UDP).
3	Network	Handles routing, IP addressing, and delivery between networks (e.g., IP, ICMP).
2	Data Link	Provides node-to-node delivery, framing, and MAC addressing (e.g., Ethernet).
1	Physical	Transmits raw bits over physical medium (e.g., cables, signals).

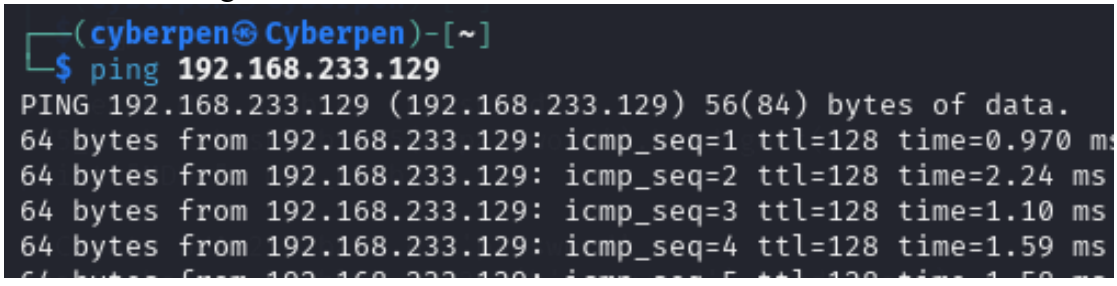
Match each OSI layer with its corresponding TCP/IP layer.

OSI Layer	Corresponding TCP/IP Layer
Application	Application
Presentation	Application
Session	Application
Transport	Transport
Network	Internet
Data Link	Link
Physical	Link

Exercise 2: Pinging the OWASP Broken Web Application

3. Task: Use the ping command to check the reachability of the OWASP Broken Web Application using its IP address.

   Command: Ping 192.168.233.129



4. **Question:**

What happens when you ping the OWASP application? Describe the process.

1. **What happens when you ping the OWASP application? Describe the process.**

When you ping the OWASP application using its IP address, your computer sends ICMP Echo Request packets to that IP, and the OWASP machine replies with ICMP Echo Reply packets. In this case, all 55 packets were successfully received, meaning the application is reachable and responsive over the network. The result also shows no packet loss, and the response times (latency) are very low around 1.5 m/s on average, which indicates a healthy, fast connection on the local network.

So, the ping confirms that:

- The OWASP application is up and running.
- The network connection between my machine and the application is stable.

Which OSI layer does the ping command operate in? Explain.

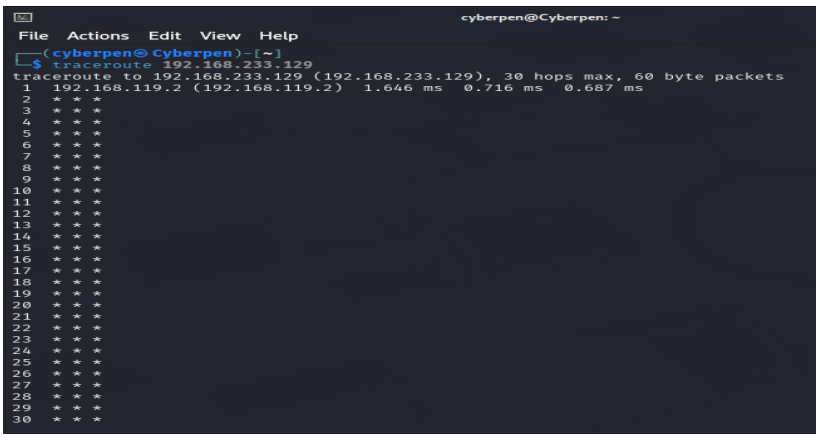
The ping command operates at the **Network layer (Layer 3**) of the OSI model.

Specifically, it uses the ICMP protocol (Internet Control Message Protocol), which is part of the IP protocol suite. ICMP is used for network diagnostics and error reporting, not for data transfer. Since it's all about addressing and routing packets between devices using IP addresses, it clearly belongs in the Network layer.

Exercise 3: Tracing the Path to the OWASP Application

5. Task: Use the traceroute command to trace the route packets take to reach the OWASP Broken Web Application.

   Command: traceroute 192.168.233.129



6. **Question:**

How many hops did it take to reach the OWASP VM?

- Based on the traceroute result, you cannot determine the exact number of hops because: Only The traceroute did not successfully reach the destination, so the number of hops to reach the OWASP VM is unknown from this result.

Describe the significance of each hop and what role traceroute plays in network troubleshooting.

Each "hop" represents a router or gateway that the packet passes through on the way to the destination.

- Hop 1: (192.168.119.2) is likely your default gateway or a local virtual network adapter
- Hop 2 to 30: The entries for this indicate that no response was received from those routers.

Role of Traceroute in Network Troubleshooting

- Traceroute helps identify where in the path a network issue is occurring.
- It shows the route packets take from your machine to a destination.
- If packets are being dropped or delayed, you can pinpoint the hop where the problem starts.

It's especially useful for:

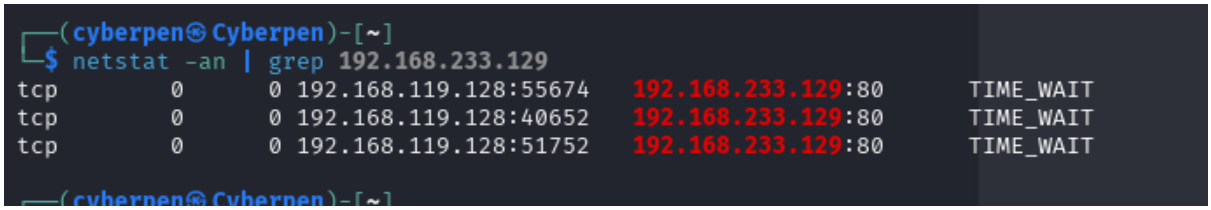
- Detecting routing loops
- Diagnosing latency or packet loss
- Understanding the network topology

Even when traceroute fails to complete, it still provides useful information. You now know the first hop is reachable and that something beyond it is blocking or not responding to traceroute probes.

Exercise 4: Viewing Active Connections to OWASP VM

7. Task: Use netstat to view active network connections between your system and the OWASP application.

Command: netstat -an | grep 192.168.233.129



8. **Question:**

- **What connections do you see? Identify the source and destination IP addresses.**

I saw three TCP connections that were recently established and are now in the `TIME_WAIT` state.

- Source IP address: `192.168.119.128`: This is my local Kali machine.
- Destination IP address: `192.168.233.129`: This is the OWASP Broken Web Application VM
- Destination port: `80`: Standard port for HTTP web traffic.

Explain how the Transport Layer (TCP/UDP) is involved in this communicate on

The Transport Layer (Layer 4 of the OSI model) is responsible for reliable delivery of data between devices, and it uses protocols like TCP and UDP.

Here's what TCP does in this process:

1. Connection Establishment:

TCP uses a 3-way handshake (SYN → SYN-ACK → ACK) to establish a reliable connection between your Kali machine and the OWASP VM.

2. Data Transfer:

After the handshake, HTTP data (like web page requests/responses) is transmitted over the TCP connection. TCP ensures that:

- Data arrives in order
- No data is lost or duplicated
- Errors are detected and corrected

3. Connection Termination:

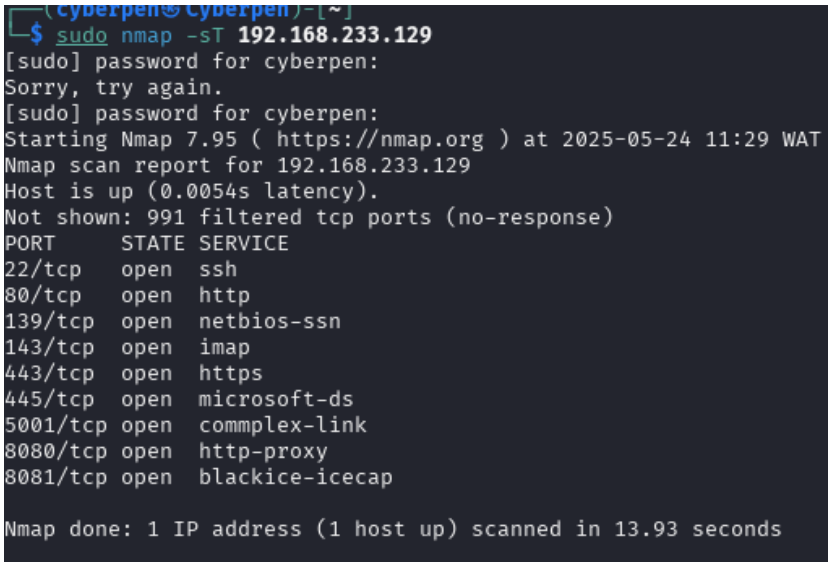
When the session ends, TCP performs an orderly connection teardown, leading to the `TIME_WAIT` state which you show in `netstat`. This state ensures that any delayed packets are handled safely before fully closing the connection.

Exercise 5: TCP vs. UDP

9. Task: Investigate the difference between TCP and UDP by scanning the OWASP Broken Web Application.

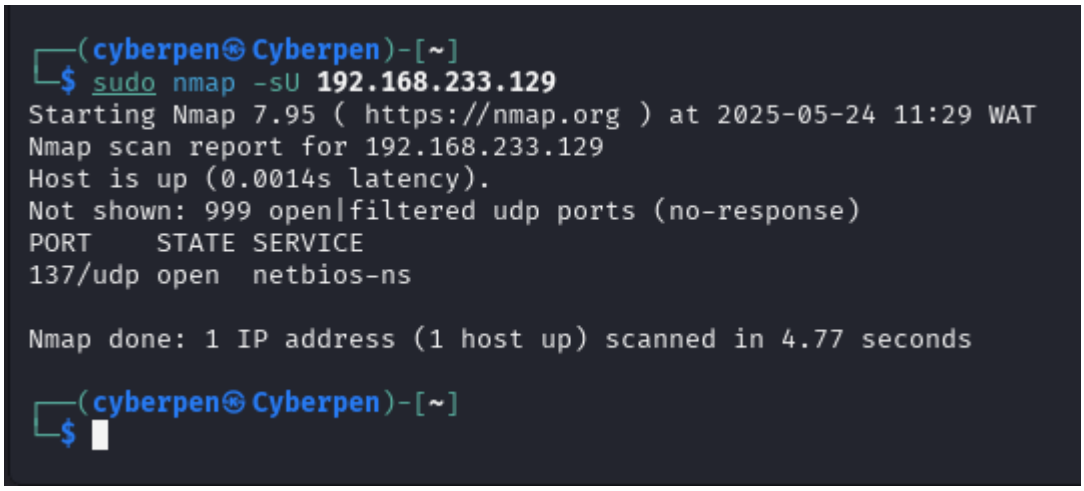
- Run a TCP scan using nmap.

Command: nmap -sT 192.168.233.129



- Run a UDP scan.

Command: nmap -sU <OWASP_IP>



10. Question:

What are the key differences between TCP and UDP in terms of reliability and speed?

- TCP (Transmission Control Protocol)
- Reliable: Establishes a connection (3-way handshake)
- Ordered: Guarantees that data arrives in order
- Error-checked: Detects and retransmits lost packets
- Slower: Because of its reliability features
- Use case: SSH, HTTP, HTTPS, Email, SMB

UDP (User Datagram Protocol)

- Unreliable: No connection or acknowledgment
- Faster: Less overhead, lower latency
- No ordering or guarantees
- Use case: DNS, VoIP, video streaming, NetBIOS

Based on the scan results, list which services on the OWASP application are using

TCP and which are using UDP.

Services using TCP are

- SSH (22)
- HTTP (80)
- NetBIOS-SSN (139)
- IMAP (143)
- HTTPS (443)
- Microsoft-DS / SMB (445)
- Commplex-Link (5001)
- HTTP-proxy (8080)
- Possibly Admin Interface (8081)

Services using UDP is:

- NetBIOS Name Service (137)

Other UDP ports might be in use but were not confirmed due to lack of response or filtering.

Exercise 6: Discovering MAC Addresses with ARP

11. Task: Use the arp command to view the MAC address of the OWASP Broken Web Application.

Command: arp -a | grep 192.168.233.129

12. Question:

- **What is the MAC address associated with the OWASP VM's IP?**

  The mac address associated tithe OWASP VM's IP of (192.168.233.129) is : 00:0c:29:8f:ca:00

- **Explain the significance of ARP in the Data Link Layer and how it contributes to successful communication.**

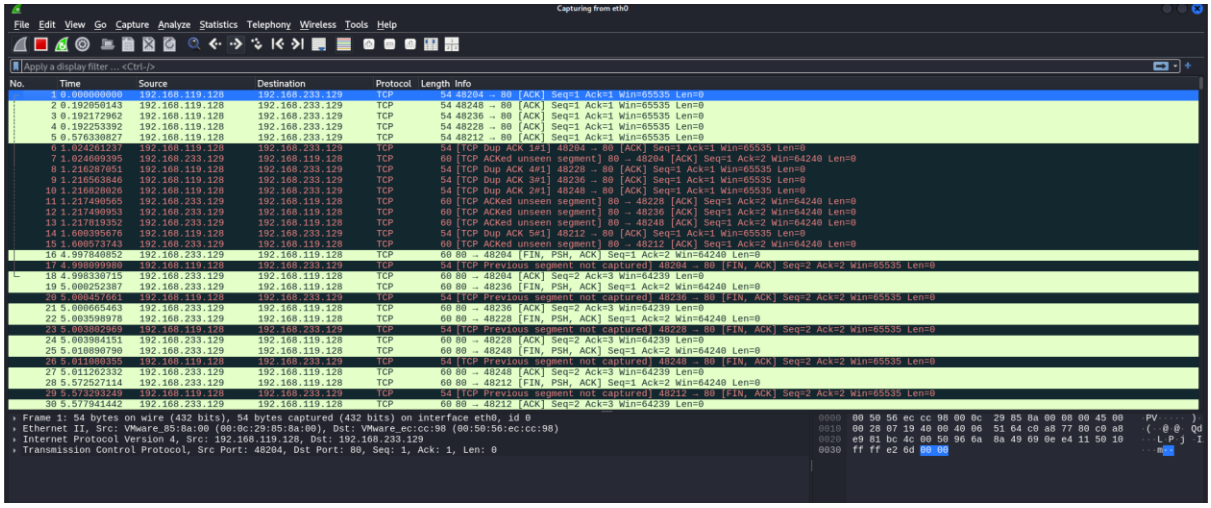
Significance of ARP in the Data Link Layer

- ARP (Address Resolution Protocol) is used to map IP addresses to MAC addresses within a local network.
- It operates between the Network Layer (Layer 3) and the Data Link Layer (Layer 2).
- When a device wants to communicate with another device on the same LAN, it needs the MAC address of the destination IP.
- If the MAC address is not already known, the device sends a broadcast ARP Request asking "Who has this IP?"
- The device with the matching IP responds with its MAC address via an ARP Reply.
- ARP enables devices to send Ethernet frames to the correct destination based on MAC address.
- The resolved MAC addresses are stored in the ARP cache to speed up future communication.
- Without ARP, devices wouldn't be able to communicate on a local Ethernet network using IP addresses.

Exercise 7: Capturing Network Traffic with Wireshark

13. Task: Use Wireshark or tshark to capture network packets between your machine and the OWASP Broken Web Application.

Command: wireshark



13. Question: Analyze the captured traffic. What protocols are in use?

i used Wireshark to capture traffic between your machine (`192.168.119.128`) and the OWASP Broken Web Application (`192.168.233.129`). The interaction occurred over port 80, which is commonly used for HTTP.

The primary protocols observed in the capture are:

- TCP (Transmission Control Protocol): This is responsible for reliable communication and is used to transmit data between the client and server.
- IP (Internet Protocol): Used for addressing and routing packets between the two IP addresses.
- Ethernet II: This protocol is part of the data link layer and is used to encapsulate the IP packets.
- Although HTTP traffic is expected since port 80 is in use, the capture doesn't clearly show HTTP packets possibly

Can you identify the handshake process or other significant events in the captured packets?

TCP Handshake Process:

- The client (`192.168.119.128`) sends a SYN packet to initiate a connection.
- The server (`192.168.233.129`) replies with a SYN-ACK packet.
- The client completes the handshake with an ACK packet.

This confirms that the TCP session was successfully established.

Significant Events Identified:

- Duplicate ACKs and Retransmissions: After the handshake, there are multiple duplicate ACK packets, which indicate potential packet loss or delay in the network. The client is repeatedly acknowledging the same segment, expecting retransmission.

- Unseen Segment Acknowledgments: Some packets show “ACKed unseen segment,” meaning packets that should have arrived weren’t captured or were lost. This can occur if Wireshark starts capturing after the session has already begun, or if packets were dropped during transmission.

- Connection Termination: The capture includes `FIN, PSH, ACK` packets from both client and server. This shows that both sides properly closed the TCP connection, following the TCP four-way termination process

Lab 2: Exploring Network Interfaces and Packet Transmission Using OWASP Broken Web Application IP

Objective:

In this lab, you will dive deeper into the concept of network interfaces and packet transmission by analyzing the network interface configuration on your system. You will explore how to capture and analyze network packets using various tools and understand the process of packet transmission to and from the OWASP Broken Web Application VM.

Learning Outcomes:

By the end of this lab, students will:

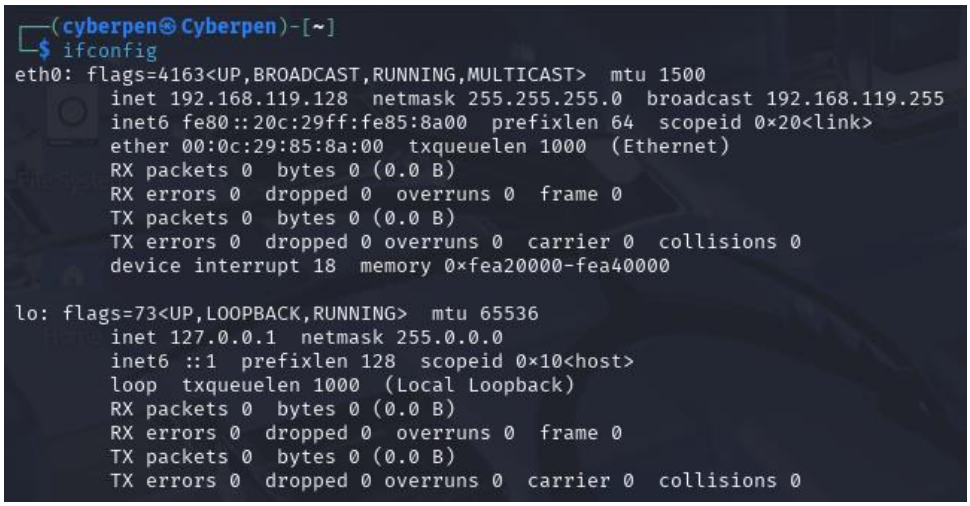
- Understand the role and configuration of network interfaces.
- Use packet capture tools to analyze network traffic and understand how data is transmitted.
- Gain experience in monitoring and troubleshooting network interfaces and traffic. Materials Needed:
- Linux-based system (Kali, Ubuntu, etc.)
- OWASP Broken Web Application VM (running and reachable on your network)
- IP address of the OWASP Broken Web Application (e.g., 192.168.X.X)
- Wireshark or tshark for packet capture

Lab Exercises:

Exercise 1: Viewing and Configuring Network Interfaces

1. Task: View your network interfaces using the ifconfig or ip command.

Command: ifconfig



2. **Question:**

- What network interfaces are available on your system?

the following network interfaces are available on my system:

  - eth0 (Ethernet interface)
  - lo (Loopback interface)

- Identify the IP address assigned to each interface.
  IP Addresses Assigned to eth0 (external network interface) are:

    - IPv4: 192.168.119.128
    - Netmask: 255.255.255.0
    - Broadcast: 192.168.119.255
    - IPv6: fe80::20c:29ff:fe85:8a00 (link-local)

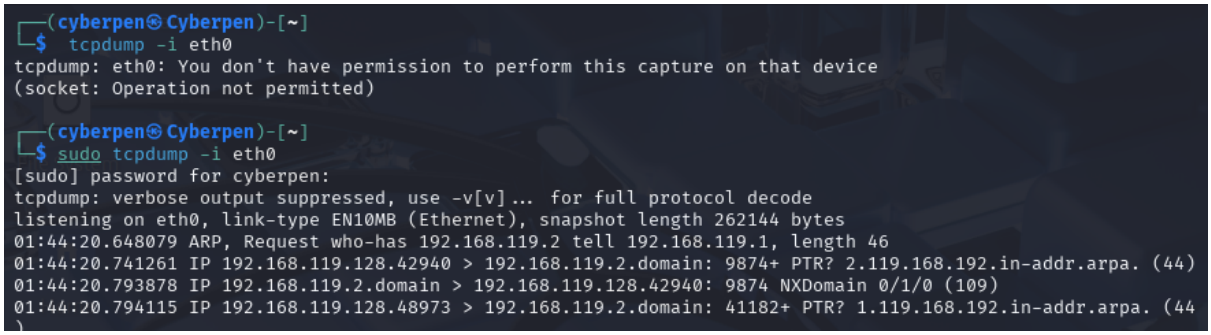
- Ip addresses assigned to lo (loopback interface):
    - IPv4: 127.0.0.1
    - Netmask: 255.0.0.0
    - IPv6::1

- Describe the difference between a loopback interface and an external network interface.
    - Loopback Interface (lo) is Used for internal communication within the host while eth0 Connects the system to external networks (e.g., LAN, internet).
    - Lo has an IP address: 127.0.0.1 (localhost) while Eth0 Has an IP address assigned by DHCP or configured statically.
    - Lo allows software on the same system to communicate with itself using networking protocols While is used for actual data transmission to/from other devices.

Exercise 2: Capturing Packets on a Specific Interface

3. Task: Use tcpdump capture packets on your primary network interface.

Command: tcpdump -i eth0



4. **Question:**

- What kind of packets are being captured?
  The packets being captured are:
  1. ARP (Address Resolution Protocol) packets:
  2. DNS (Domain Name System) reverse lookup packets

- **Are there any packets related to communication with the OWASP Broken Web Application VM? Yes, there is strong evidence and they are:**
  - All captured traffic is between your system (192.168.119.128) and 192.168.119.2.
  - It's likely that 192.168.119.2 is the IP of your OWASP BWA VM.
  - The traffic includes:
    - Reverse DNS queries directed to 192.168.119.2
    - ARP traffic attempting to resolve 192.168.119.2

- Describe the role of this network interface in transmitting and receiving packets.

  Role of eth0 Network Interface:
  The eth0 interface is the system's primary external (Ethernet) network interface. It serves the following roles:

  1. Transmitting Packets: Sends DNS queries and ARP requests to find other devices or services on the network.

  2. Receiving Packets: Listens for responses from those devices, such as ARP replies and DNS answers.

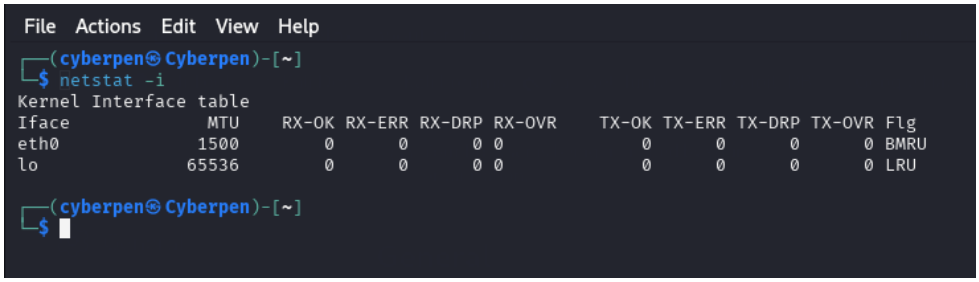
  3. Enabling Communication: Facilitates interaction between your system and the OWASP BWA VM or any other device on the same virtual network.

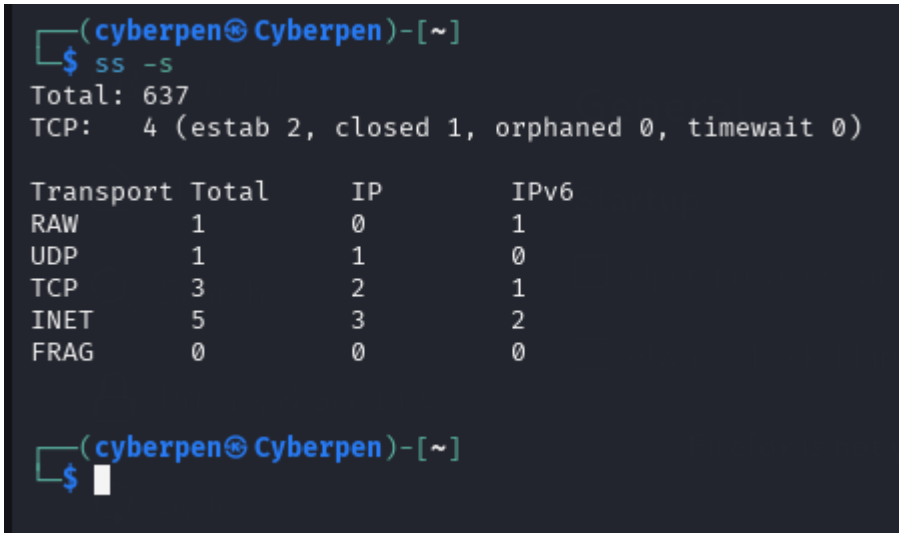
  4. In this case, eth0 is critical for the host's communication with the OWASP VM for tasks such as DNS resolution, service discovery, and web app penetration testing.

Exercise 3: Examining Network Statistics

5. Task: Use the netstat or ss command to view current network statistics and connections.

Command: netstat -i or ss -s





6. **Question:**

- **What is the current status of your network interfaces?**

The network interfaces are up, but there has been no network traffic at the moment. Interfaces listed: eth0 and lo (loopback).

MTU:

- eth0: 1500 (standard Ethernet MTU).
- lo: 65536 (for loopback, normal).

- **What active connections are visible?**
  From the ss -s output:
  - TCP connections:
  - 4 total
  - 2 established (actively communicating)
  - 1 closed
  - 0 orphaned or in time wait

So, there are 2 currently active TCP connections.

- **Explain the significance of these statistics in monitoring network performance.**
  Significance of Network Statistics in Monitoring Performance
  - RX-OK / TX-OK: Indicates the number of successfully received and transmitted packets — helps verify normal network operation.
  - RX-ERR / TX-ERR: Shows packet errors during receiving or sending can point to faulty network hardware, bad cables, or driver issues.
  - RX-DRP / TX-DRP: Dropped packets may indicate buffer overflows, congestion, or performance bottlenecks.
  - RX-OVR / TX-OVR: Overruns happen when the network stack can't keep up with the packet rate can lead to performance degradation.

- Established Connections: Shows how many connections are currently active useful for spotting unusual activity or potential attacks.
- Closed or Time wait Connections: Helps assess how the system is handling connection cleanup, which can affect resource usage.
- MTU (Maximum Transmission Unit): Affects how much data can be sent in one packet incorrect MTU can cause fragmentation and slowdowns.
- Protocol Usage (TCP/UDP/RAW): Helps identify what types of communication are active excessive raw socket use might be suspicious.
- IPv4 vs IPv6 Distribution: Helps monitor the type of network traffic, which can be useful in dual-stack environments.

Exercise 4: Monitoring Network Traffic with Wireshark

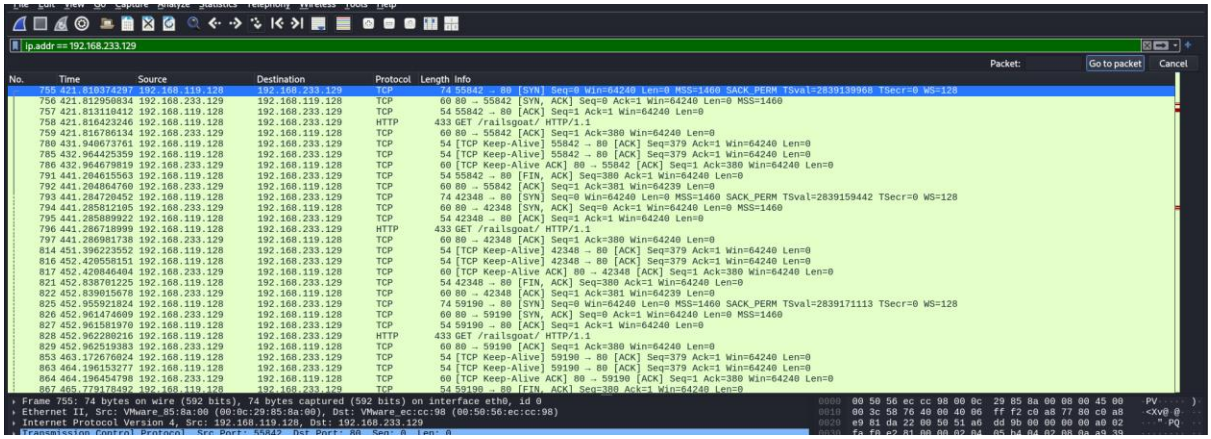
7. Task: Use Wireshark to monitor network traffic on your primary interface and filter traffic going to and from the OWASP Broken Web Application VM.

Command: Open Wireshark, choose your network interface, and set the

filter to ip.addr == <OWASP_IP> (replace <OWASP_IP> with the actual IP,

8. **Question:**

- Analyze the packets going to and from the OWASP VM. What types of protocols are in use?



Can you identify any key packet details such as source/destination IP addresses, port numbers, and flags?

Yes. Some examples:

- Source IP: 192.168.119.128 (attacker or user's machine)
- Destination IP: 192.168.233.129 (OWASP BWA VM)
- Port Numbers:
    - Source Ports: Varying ephemeral ports (e.g., 55842, 42348, 59190)
    - Destination Port: 80 (standard for HTTP)

- TCP Flags Observed:
    - SYN: Initiating a new TCP connection
    - ACK: Acknowledging received data
    - FIN: Gracefully closing a connection
    - SYN, ACK: Part of TCP handshake
    - Keep-Alive: Maintains idle connections

- **How does the TCP/IP model apply to the data captured?**

The TCP/IP model can be applied to the captured packets as follows:

1. Application Layer
    - HTTP is used for communication with the web application.
    - Visible in packets like GET /railgoat/ HTTP/1.1.
2. Transport Layer
    - TCP provides reliable delivery and connection control (SYN, ACK, FIN).
    - Port numbers help identify services (port 80 = HTTP).
3. Internet Layer
    - IP addresses used for routing (e.g., 192.168.119.128 ↔ 192.168.233.129).
    - Packets are filtered based on IP (ip.addr filter).
4. Network Interface Layer (Link Layer)
    - While not detailed in the screenshot, Ethernet frames are the actual medium-level transport.
    - Shown in the frame header info (e.g., MAC addresses like 00:0c:29:85:8a:00).

Summary:

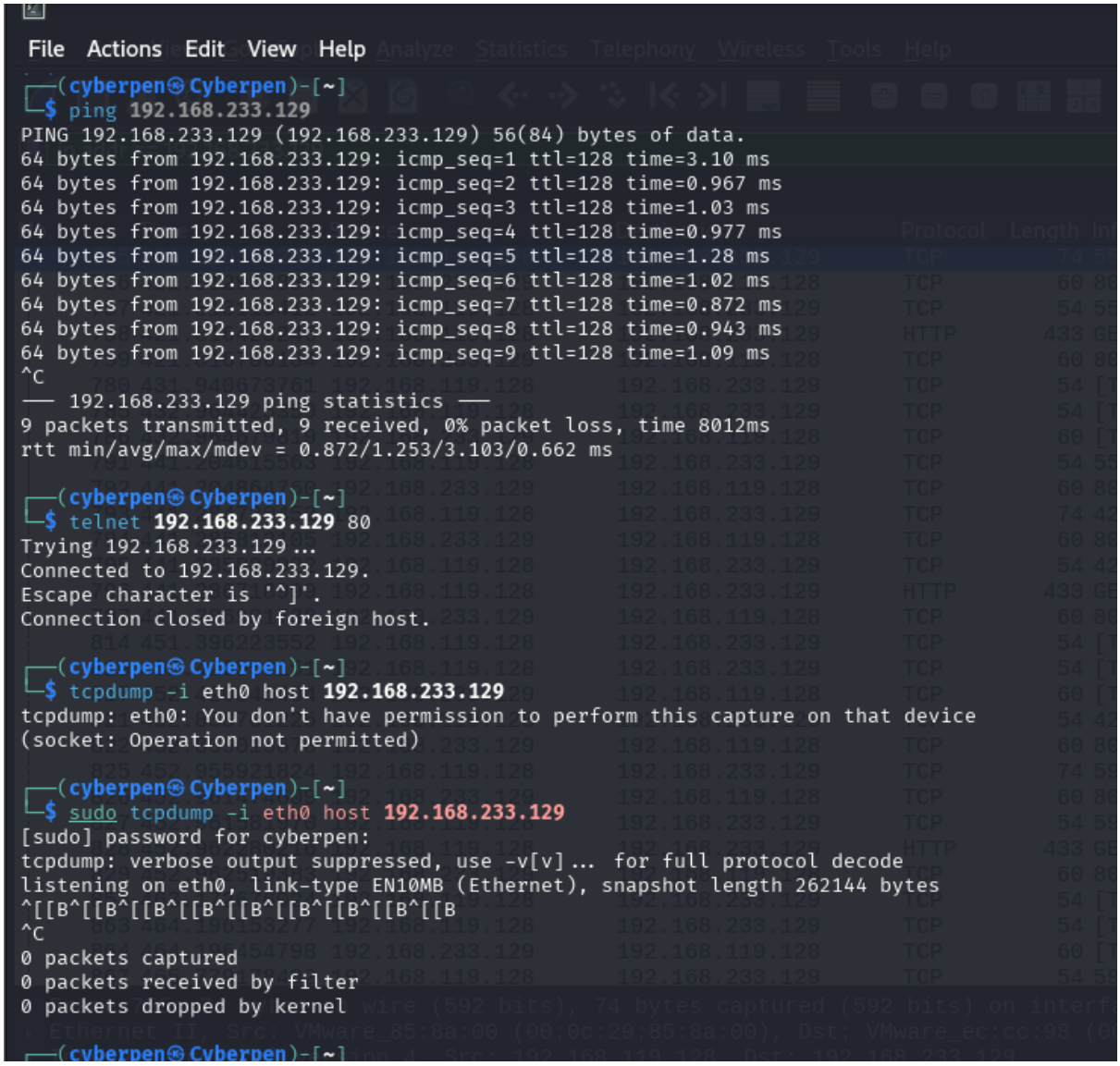
- Protocols: TCP and HTTP.
- IP Communication: From host 192.168.119.128 to OWASP VM 192.168.233.129.
- Common Flags: SYN, ACK, FIN, Keep-Alive — showing full connection setup and teardown.
- TCP/IP Model Use:
    - Application Layer: HTTP
    - Transport Layer: TCP
    - Internet Layer: IP
    - Network Layer: Ethernet (implied)

Exercise 5: Packet Transmission Analysis

9. Task: Perform a ping or TCP connection request to the OWASP Broken Web

Application IP and capture the packet details using tcpdump or Wireshark.

- Command: ping <OWASP_IP> or telnet <OWASP_IP> 80 (to initiate a simple connection)
- Capture the traffic using tcpdump -i <interface> host <OWASP_IP>, or set the appropriate filter in Wireshark.



10. Question:

- What do you observe during the packet transmission process?

Ping Process

During the ping process, we observe a series of ICMP Echo Request packets being sent to 192.168.233.129, and ICMP Echo Reply packets being received back. Each line of the ping output shows the number of bytes received, the source IP, icmp_seq (sequence number), ttl (Time To Live), and time (round-trip time). The ping statistics summarize the number of packets transmitted, received, packet loss percentage, and round-trip time statistics (min/avg/max/mdev).

Telnet Process

For the telnet process, we observe an attempt to establish a TCP connection. The "Trying 192.168.233.129..." indicates the client is trying to connect. "Connected to 192.168.233.129." confirms that a TCP connection was successfully established. "Escape character is '^]'." is a telnet-specific message. "Connection closed by foreign host." indicates that the server on 192.168.233.129 (likely a web server not expecting a raw telnet session) closed the connection after it was established.

tcpdump

For tcpdump, we observe that it requires elevated privileges (sudo) to capture packets directly from the network interface. Once started with sudo, it successfully initiates listening on the specified interface (eth0) with the host filter. If traffic were to occur at that moment, it would display detailed information about the captured packets.

Describe the handshake process or the round-trip of the packets for ping or TCP connection.

- Ping (ICMP Round-Trip):

The ping command uses the ICMP (Internet Control Message Protocol) to check network connectivity. The process is a simple request-reply mechanism:

1. The client (your machine) sends an ICMP Echo Request packet to the target IP address (192.168.233.129).
2. The target host, upon receiving the Echo Request, responds with an ICMP Echo Reply packet back to the client. This entire process (sending an Echo Request and receiving an Echo Reply) constitutes one round-trip. The ping command measures the time taken for this round-trip for each packet and reports it as the time value.

- TCP Connection (Three-Way Handshake):

The telnet command (when connecting to port 80) initiates a TCP three-way handshake to establish a reliable connection:

1. SYN (Synchronize Sequence Numbers): The client sends a TCP segment with the SYN flag set to the server. This segment also includes an initial sequence number (ISN) from the client.
2. SYN-ACK (Synchronize-Acknowledgment): The server, upon receiving the SYN from the client, responds with a TCP segment that has both the SYN and ACK flags set. The SYN flag indicates the server's own ISN, and the ACK flag acknowledges the client's ISN (by sending client's ISN + 1).
3. ACK (Acknowledgment): The client, upon receiving the SYN-ACK from the server, sends a final TCP segment with the ACK flag set. This segment acknowledges the server's ISN (by sending server's ISN + 1). Once this three-way handshake is complete, a full-duplex TCP connection is established, and data can be exchanged between the client and the server. In the telnet example, data exchange was brief before the server closed the connection.

- **Which layers of the OSI and TCP/IP models are involved in this transmission?**
- Let's break it down for both `ping` and `telnet`:

  For `ping` (ICMP):

  - **OSI Model:**
    - **Layer 1: Physical Layer:** Responsible for the physical transmission of bits over the network medium (e.g., Ethernet cable, Wi-Fi signals).
    - **Layer 2: Data Link Layer:** Responsible for framing data into frames, MAC addressing, and error detection within the local network segment. (e.g., Ethernet for wired connection)
    - **Layer 3: Network Layer:** This is where ICMP and IP operate. IP provides logical addressing (192.168.233.129) and routing of packets. ICMP messages (Echo Request/Reply) are encapsulated within IP packets.
    - **(Implicitly) Layer 4: Transport Layer:** While ICMP doesn't use TCP or UDP, it's often considered to be at the Network Layer (Layer 3). However, some interpretations place it conceptually "above" IP for control messages. For practical purposes, it's firmly routed by IP.
  - **TCP/IP Model:**
    - **Network Access Layer (or Link Layer):** Combines OSI Layers 1 and 2 (e.g., Ethernet). Handles the physical transmission and local network addressing.
    - **Internet Layer:** This is where IP and ICMP reside. IP handles logical addressing and routing, while ICMP carries the ping messages.

  For `telnet` (TCP Connection to HTTP port 80):

  - **OSI Model:**
    - **Layer 1: Physical Layer:** Physical transmission of bits.
    - **Layer 2: Data Link Layer:** Framing, MAC addressing (e.g., Ethernet).
    - **Layer 3: Network Layer:** IP addressing and routing of packets from source to destination.
    - **Layer 4: Transport Layer:** This is where TCP operates. TCP provides reliable, connection-oriented communication, handles segment sequencing, flow control, and error recovery (e.g., the three-way handshake occurs here). Telnet as an application uses TCP as its transport protocol.
    - **Layer 5: Session Layer:** Manages communication sessions, establishing, managing, and terminating connections. (Telnet itself implicitly handles some session aspects).
    - **Layer 6: Presentation Layer:** Deals with data formatting, encryption, and compression. (Less prominent in a raw telnet to HTTP, but still conceptually present).
    - **Layer 7: Application Layer:** This is where HTTP (the protocol for web servers) and the Telnet client itself operate. The Telnet application provides a user interface to interact with network services, and in this case, it's attempting to speak HTTP.

- **TCP/IP Model:**
  - **Network Access Layer (or Link Layer):** Combines OSI Layers 1 and 2 (e.g., Ethernet).
  - **Internet Layer:** IP addressing and routing.
  - **Transport Layer:** This is where TCP operates, providing reliable connections and the three-way handshake.
  - **Application Layer:** This combines OSI Layers 5, 6, and 7. HTTP (the service on port 80) and the Telnet client application reside here, using the underlying TCP connection for communication.

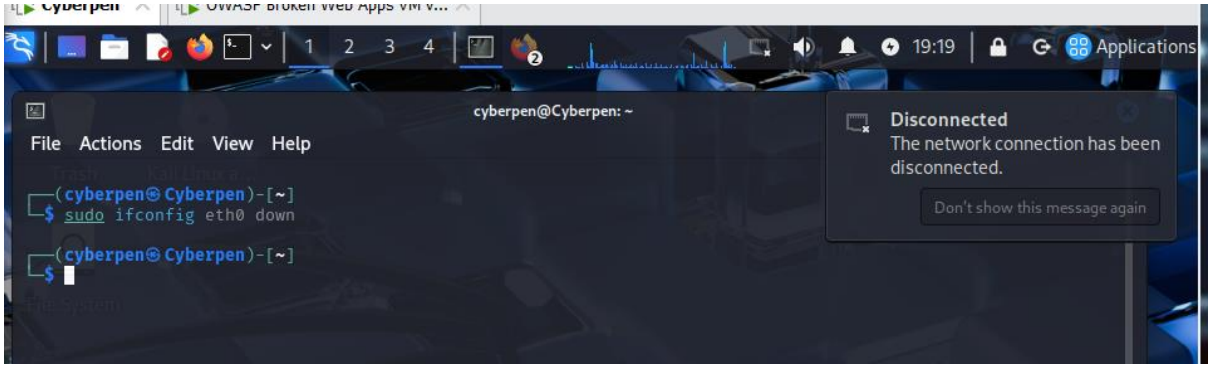
Exercise 6: Troubleshooting Network Interface Issues

11. Task: Simulate a network interface failure by disabling and then re-enabling an interface. Observe how this affects network connectivity to the OWASP Broken Web Application VM.

    Command: sudo ifconfig <interface> down and sudo ifconfig <interface> up (or use ip commands)

12. Question:

- What happens when you disable the network interface?



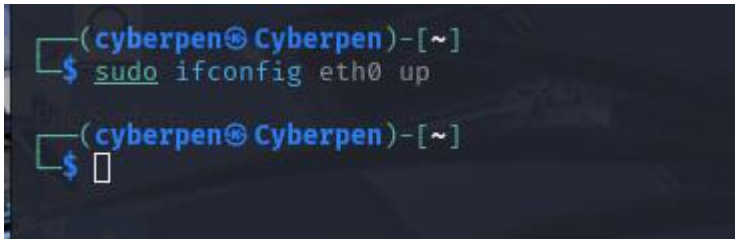
The terminal shows the command `sudo ifconfig eth0 down` being executed. This command is used to disable the `eth0` network interface.

System Response: Immediately after the command is executed (or very shortly thereafter), a notification bubble appears on the top right of the screen stating:

- **"Disconnected"**
- **"The network connection has been disconnected."**

This visual confirmation directly demonstrates what happens when you disable a network interface. It confirms the loss of network connectivity that was described in the answer to Question 12, specifically: "Loss of Network Connectivity" and "No Internet Access (if it's your primary interface)." The system's graphical environment promptly notifies the user about the loss of connection.

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



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

When sudo ifconfig eth0 up is executed:

1. Network Connectivity is Restored: The most critical response is the immediate restoration of network connectivity. The eth0 interface is brought back online, allowing your system to send and receive network traffic.
2. IP Address Re-acquisition (if DHCP): If the interface is configured to obtain an IP address via DHCP (which is common for virtual machines like Cyberpen), the system will automatically initiate a DHCP request. It will obtain an IP address, subnet mask, default gateway, and DNS server information from the DHCP server.
3. Network Services Resume: Any network-dependent applications or services that were failing due to the disabled interface will now be able to function correctly. For instance, if you then tried to ping the OWASP Broken Web Application VM (as in the previous task), it would succeed, assuming the VM is still accessible.
4. No Immediate Visual Notification (often): Unlike disabling, which often triggers a "Disconnected" notification, re-enabling might not always display an explicit "Connected" notification in all desktop environments, especially if the connection was quickly re-established. However, network indicators (like a network icon in the system tray) would typically reflect the connected state.
5. ARP Cache Update: The system will rebuild its ARP (Address Resolution Protocol) cache as it communicates with other devices on the network.

Explain how network administrators can use this knowledge for troubleshooting connectivity issues.

Network administrators leverage the ability to disable and re-enable interfaces as a fundamental troubleshooting technique for several reasons:

1. Resetting the Network Stack (The "Off-and-On Again" Rule): This is perhaps the most common use. Sometimes, network drivers or the operating system's network stack can get into a glitched or unresponsive state. Disabling and then re-enabling the interface effectively performs a "soft reset" of the network configuration for that specific interface. This can resolve transient issues like:
    - Stuck network connections.
    - Incorrect IP address assignments (forces a new DHCP request).
    - Problems with DNS resolution that are client-side.
    - Minor driver issues that don't require a full reboot.

2. Verifying Interface Functionality: If a system is experiencing connectivity problems, an administrator can disable and re-enable an interface to see if it comes back online and acquires a valid IP address. If it doesn't come up, it points to a deeper issue such as:
    - A faulty network cable (Layer 1 problem).
    - A bad network port on the switch or router.
    - A problem with the network card's driver or hardware.
    - Incorrect static IP configuration.
3. Applying Network Configuration Changes: After making manual changes to network configuration files (e.g., /etc/network/interfaces on Linux, or similar configurations on other OSes), administrators often disable and re-enable the interface to ensure the new settings are applied without requiring a system reboot. This is quicker and less disruptive.
4. Isolating Network Problems: In environments with multiple network interfaces (e.g., a server with interfaces for management, production, and storage networks), an administrator might temporarily disable specific interfaces to:
    - Determine which interface is causing a routing conflict.
    - Isolate traffic for testing.
    - Troubleshoot firewall rules affecting a particular interface.
5. Security Incident Response: In some extreme cases, if a system is suspected of being compromised or is generating malicious traffic, an administrator might temporarily disable its network interface to immediately cut off its network access and contain the threat, allowing them time to investigate without the system affecting the rest of the network.

In summary, the ifconfig down and ifconfig up commands provide a quick and powerful way for network administrators to diagnose, reset, and manage the state of network interfaces, making them invaluable tools in day-to-day network troubleshooting.

Exercise 7: Bandwidth Monitoring

13. Task: Use the iftop or nload command to monitor the bandwidth usage on your system while interacting with the OWASP Broken Web Application VM.

    nload eth0

Command: sudo iftop -i eth0

14. Question:

- **What is the current bandwidth usage while communicating with the OWASP VM?**

  For Incoming Traffic:
  - Curr (Current): 496.00 Bit/s
  - Avg (Average): 176.00 Bit/s
  - Min (Minimum): 0.00 Bit/s
  - Max (Maximum): 1.09 KBit/s (which is 1090 Bit/s)
  - Total (Ttl): 1.21 MByte

  For Outgoing Traffic:
  - Curr (Current): 0.00 Bit/s
  - Avg (Average): 192.00 Bit/s
  - Min (Minimum): 0.00 Bit/s
  - Max (Maximum): 1.03 KBit/s (which is 1030 Bit/s)
  - Total (Ttl): 346.23 Kbyte

  These values represent the real-time and aggregated bandwidth usage during the period iftop was running. The "Curr" values show the instantaneous rate, while "Avg" provides a smoothed average. The "Max" value indicates the highest burst of traffic observed. The "Ttl" shows the total data transferred.

Identify the impact of network traffic on your interface. Is there any traffic congestion?

Based on the observed bandwidth usage:

- Impact: The current traffic levels (in the hundreds of bits per second, with peak Kbits per second) are extremely low. For typical modern network interfaces (Ethernet, Wi-Fi), which usually operate at speeds of 100 Mbps (Megabits per second) to 1 Gbps (Gigabits per second) or more, these traffic levels are negligible.
- Traffic Congestion: Absolutely no traffic congestion is indicated by these numbers. Traffic congestion occurs when the network interface or pathway is nearing its capacity, leading to delays, packet drops, and slow performance. Here, the usage is orders of magnitude below typical network capacities. The "0.00 Bit/s" for current outgoing traffic suggests that at the moment the screenshot was taken, no data was actively being sent, while a very small amount was being received. This low traffic

likely indicates that you are not actively interacting with the OWASP VM (e.g., Browse pages, performing scans) while this specific iftop screenshot was captured. If you were, you would expect to see higher "Curr" and "Avg" values.

- **How does this help in monitoring network performance?**

Tools like iftop are incredibly valuable for network performance monitoring and troubleshooting:

1. Real-time Visibility: iftop provides immediate, real-time insights into what's happening on your network interface. This is crucial for quickly identifying sudden spikes in traffic or unexpected data flows.

2. Identifying Bandwidth Hogs: It displays individual connections (source/destination IP addresses and ports) and their respective bandwidth consumption. This allows administrators to quickly pinpoint which applications, users, or devices are consuming the most bandwidth. For example, if the OWASP VM was actively being scanned, you would see high traffic to/from its IP address.

3. Detecting Anomalous Traffic: Unusual spikes in traffic, or consistent high traffic from unexpected sources, can indicate various issues, such as:
    - Malware activity (e.g., botnet communication, data exfiltration).
    - Misconfigured applications or services.
    - Network loops or broadcast storms (though iftop primarily shows unicast/multicast).
    - Denial-of-Service (DoS) attacks.

4. Troubleshooting Slowdowns: When users complain about slow network performance, iftop can help determine if the bottleneck is local (e.g., a process on the machine itself hogging bandwidth) or if the issue lies further upstream in the network. If the local interface isn't saturated but performance is poor, the issue is elsewhere.

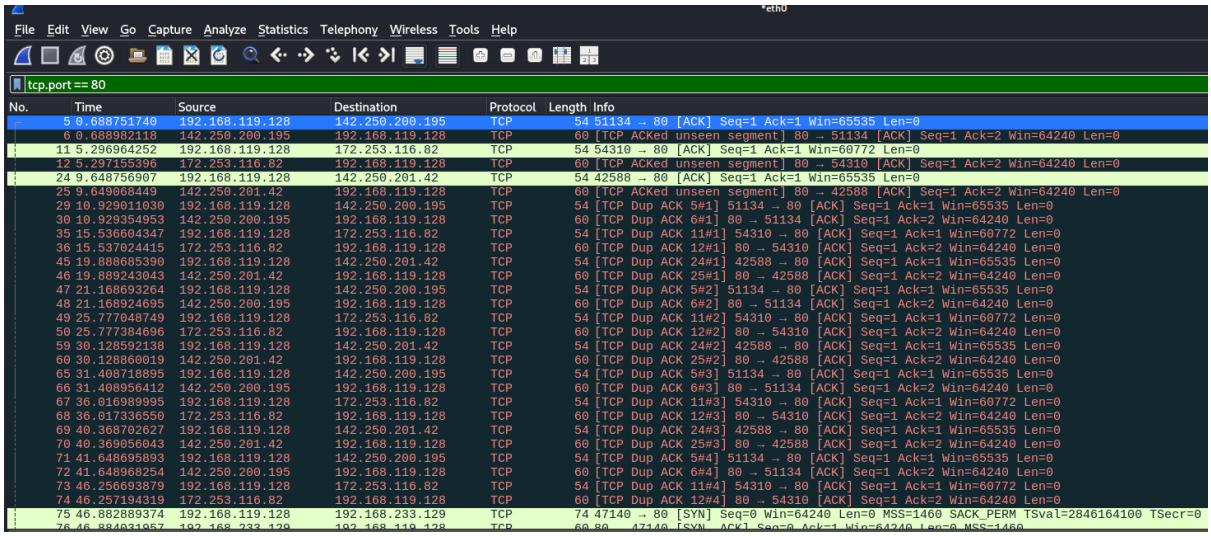
5. Capacity Planning: By monitoring peak and average bandwidth usage over time (often using more advanced tools than iftop for logging, but iftop gives a snapshot), administrators can make informed decisions about upgrading network infrastructure to meet demand.
6. Verifying Connectivity and Traffic Flow: Seeing traffic to/from expected IPs confirms that network paths are operational and data is flowing as anticipated.

In essence, iftop provides a "network top" view, showing bandwidth consumption in a similar way top or htop shows CPU and memory usage, making it an indispensable tool for diagnosing and managing network performance.

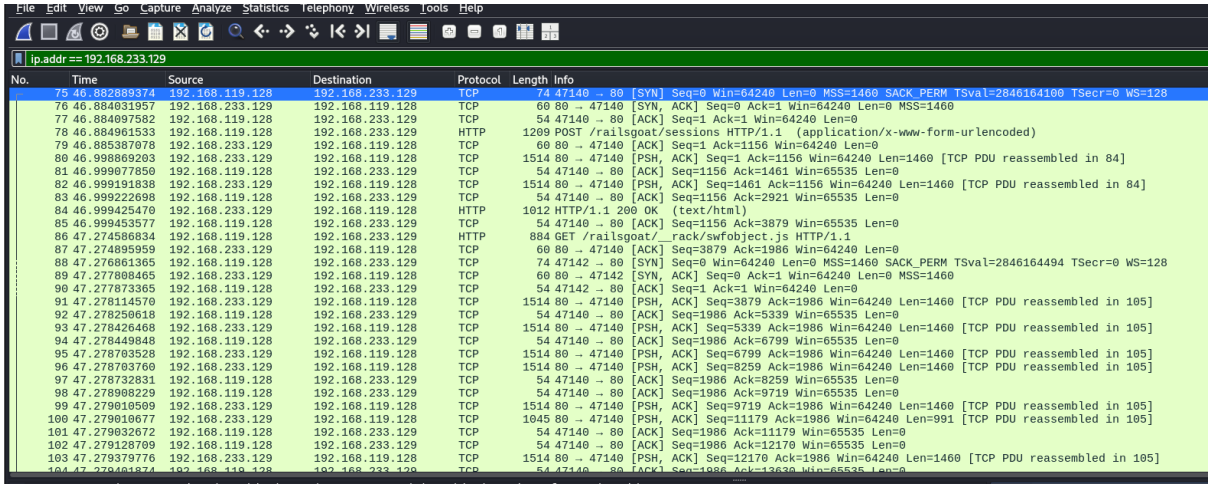
Exercise 8: Advanced Packet Capture Filters

15. Task: Create custom filters in Wireshark or tcpdump to capture only specific types of traffic, such as TCP or HTTP traffic, between your system and the OWASP VM.

Tcp port == 80



Ip.addr == 192.168.233.129



16. Question:

What is the significance of filtering specific traffic?

The significance of filtering specific traffic in packet capture tools like Wireshark and tcpdump is profound for several reasons:

1. Reduced Noise and Focus: Raw packet captures can quickly become overwhelmingly large and contain vast amounts of irrelevant data. Filtering allows you to cut through the "noise" and focus only on the packets relevant to your investigation, making analysis much faster and more efficient.

2. Performance and Storage: Capturing all traffic, especially on busy networks, consumes significant CPU resources and disk space. Filters reduce the volume of data being processed and saved, leading to smaller capture files and less performance impact.
3. Targeted Analysis: Filters enable you to perform targeted analysis. For example, if you're troubleshooting a web server issue, filtering for HTTP traffic to that server immediately shows you the relevant requests and responses. If you're looking for a specific malicious connection, filtering by IP address or port can quickly isolate it.
4. Security and Privacy: In environments with sensitive data, filtering ensures that only the necessary traffic is captured, minimizing the exposure of confidential information that might be present in unrelated packets.
5. Reconstruction of Specific Conversations: Filters allow you to reconstruct specific conversations or sessions (e.g., all TCP packets belonging to a single web request-response cycle) without being distracted by other ongoing network activity.

How can advanced filters help network engineers diagnose and resolve issues ?.

Advanced packet capture filters are indispensable for network engineers in diagnosing and resolving a wide array of network issues:

1. Root Cause Analysis:
    - Application Performance Issues: Filter for HTTP, DNS, database (e.g., MySQL, PostgreSQL), or other application-specific protocols to see if delays are at the network layer (e.g., high latency, retransmissions) or the application layer (e.g., slow server responses, database queries). Image 2, for example, shows HTTP GET requests and 200 OK responses, allowing an engineer to see the HTTP conversation flow.
    - Connectivity Problems: Filter by IP address or port to see if traffic is reaching the destination, if firewalls are blocking connections (by seeing SYN but no SYN-ACK, for instance, as shown in Image 1 indicating TCP activity but potentially not successful full handshakes with all IPs), or if routing issues prevent packets from reaching their target.
    - Authentication Failures: Filter for authentication protocols (e.g., Kerberos, LDAP, RADIUS) to identify where the authentication process is breaking down.

2. Security Investigations:
    - Malware Analysis: Filter for suspicious ports, unusual protocols, or communication with known malicious IPs to identify compromised systems or data exfiltration.
    - Intrusion Detection: Look for specific attack signatures or patterns of traffic (e.g., excessive SYN packets indicating a SYN flood).
    - Policy Enforcement: Verify that firewall rules and security policies are correctly applied by checking if expected traffic is allowed and blocked traffic is indeed blocked.

3. Network Performance Tuning:
    - Bandwidth Usage: Identify the top talkers and biggest bandwidth consumers on a segment by filtering by IP address or subnet, helping to optimize network resources (as seen with iftop in the previous exercise, but tcpdump/Wireshark provide more detail).
    - Latency and Jitter: Analyze timestamps between packets to identify sources of network delay.
    - Packet Loss/Retransmissions: Filters for TCP retransmissions (like the "TCP DUP ACK" seen in Image 1) or out-of-order packets can pinpoint congestion or unreliable links.

4. Protocol Analysis:
    - Protocol Compliance: Verify that applications and devices are adhering to protocol standards.
    - Debugging Protocol Implementations: Developers use filters to debug their network-aware applications by examining the exact sequence of packets exchanged.

5. Configuration Verification:
    - DHCP/DNS Issues: Filter for DHCP or DNS traffic to ensure clients are getting correct IP assignments or resolving hostnames correctly.
    - Routing Issues: Observe routing protocol updates (e.g., OSPF, BGP) or traceroute packets to diagnose routing table problems.

By using precise filters, network engineers can transform a chaotic stream of network data into actionable intelligence, significantly accelerating the diagnostic and resolution process for complex network problems.1.

Lab 3: TCP/IP Protocol Stack and Packet Inspection Using OWASP Broken Web Application IP

Objective:

In this lab, students will delve into the Transmission Control Protocol/Internet Protocol (TCP/IP) stack, dissecting and inspecting the various layers involved in packet transmission. Students will focus on understanding how data travels across a network analyzing packets, and troubleshooting issues related to the TCP/IP model.

Learning Outcomes:

By the end of this lab, students will:

- Understand the fundamentals of the TCP/IP model.
- Be able to analyze packets at different layers (Application, Transport, Network, and Link).
- Gain hands-on experience in dissecting TCP/IP traffic using packet inspection tools.
- Learn how to apply practical troubleshooting techniques based on TCP/IP behavior.

Materials Needed:

- Linux-based system (Kali, Ubuntu, etc.)
- OWASP Broken Web Application VM (running and reachable on your network)
- IP address of the OWASP Broken Web Application (e.g., 192.168.X.X)
- Wireshark or tshark for packet capture and analysis

Lab Exercises:

Exercise 1: Understanding the TCP/IP Model

1. **Task:** Briefly review the layers of the TCP/IP model (Application, Transport, Network, and Link layers). Understand the purpose and function of each layer in data transmission.

   The TCP/IP model is a conceptual framework that describes how data is communicated over a network. It's often compared to a "military-grade" approach to networking, designed for robustness and internetworking. It consists of four (or sometimes five) layers:

   1. **Application Layer:**
      - Purpose: This is the top layer, closest to the end user. It provides services directly to the applications that the user interacts with.
      - Function: Defines the protocols and services for specific network applications. It handles communication for specific programs.

- Examples: HTTP (Web Browse), FTP (File transfer), SMTP (Email), DNS (Domain Name Resolution), Telnet, SSH, SNMP.

2. **Transport Layer:**
   - Purpose: Responsible for end-to-end communication between applications on different hosts. It ensures reliable and ordered delivery of data between processes.
   - Function: Manages flow control (preventing a fast sender from overwhelming a slow receiver), error checking (detecting corrupted data), and multiplexing (allowing multiple applications to share the same network connection). It segments data from the Application layer and reassembles it at the destination.
   - Key Protocols:
     - TCP (Transmission Control Protocol): Connection-oriented, reliable, ordered, and error-checked delivery. Used for applications that require high reliability (e.g., web Browse, email, file transfer).
     - UDP (User Datagram Protocol): Connectionless, unreliable, faster delivery. Used for applications where speed is more critical than guaranteed delivery (e.g., streaming video, online gaming, DNS queries).

3. **Internet Layer (or Network Layer):**
   - Purpose: Responsible for logical addressing (IP addresses) and routing packets across different networks (internetworking).
   - Function: Determines the best path for data packets to travel from the source host to the destination host, even if they are on different physical networks. It adds IP headers to segments from the Transport layer, creating IP datagrams.
   - Key Protocols:
     - IP (Internet Protocol): The primary protocol for addressing and routing packets.
     - ICMP (Internet Control Message Protocol): Used for error reporting and network diagnostics (e.g., ping).
     - ARP (Address Resolution Protocol): Maps IP addresses to MAC addresses.

4. Network Access Layer (or Link Layer / Data Link Layer):

- Purpose: Deals with the physical transmission of data frames over a local network.
- Function: Defines how data is physically sent over the network medium (e.g., Ethernet cable, Wi-Fi). It handles physical addressing (MAC addresses), framing (packaging data into frames), error detection within the local link, and control over physical access to the network.
- Examples: Ethernet, Wi-Fi (802.11), PPP (Point-to-Point Protocol), ATM. This layer often combines aspects of the OSI model's Physical and Data Link layers.

2. Question:

- Explain the differences between the TCP/IP and OSI models.

The TCP/IP model and the OSI (Open Systems Interconnection) model are both conceptual frameworks that describe how network communication works. While they share similar goals, they differ in their structure, origins, and practical application:

1. **Number of Layers:**
   - OSI Model: Has 7 distinct layers.
   - TCP/IP Model: Traditionally has 4 layers, though some modern interpretations expand it to 5 by explicitly separating the Physical and Data Link layers within the Network Access layer.

2. **Origin and Purpose:**
   - OSI Model: Developed by ISO (International Organization for Standardization) as a theoretical and standardized blueprint for network communication, intended to be a universal model before TCP/IP gained prominence. It is more prescriptive.
   - TCP/IP Model: Developed by the Department of Defense (DoD) in the U.S. for the ARPANET (predecessor to the Internet). It was designed to be a practical, working model that would facilitate robust and fault-tolerant communication, even in the event of network failures. It is more descriptive of what actually works on the internet.

3. **Relationship to Protocols:**
   - OSI Model: Protocols were developed after the model was defined, making it a "model first, then protocols" approach. Many OSI protocols didn't achieve widespread adoption.
   - TCP/IP Model: The model was developed from existing protocols (like TCP and IP) that were already in use. It's a "protocols first, then model" approach, making it a more accurate representation of how the internet actually functions.

4. **Strictness of Layering:**
   - OSI Model: More rigid and strictly defines the functions of each layer, with clear boundaries between them.
   - TCP/IP Model: More flexible and less strict about its layering. Some layers in the TCP/IP model perform functions that span multiple OSI layers (e.g., TCP/IP's Network Access layer combines OSI's Physical and Data Link layers).

5. **Connection-Oriented vs. Connectionless:**
   - OSI Model: All layers can be either connection-oriented or connectionless.
   - TCP/IP Model: The Internet layer (IP) is inherently connectionless, while the Transport layer offers both connection-oriented (TCP) and connectionless (UDP) services.

In essence, the OSI model is often used for theoretical understanding and teaching due to its detailed breakdown, while the TCP/IP model is more widely used for practical implementation and understanding how the Internet works.

Which layers of the TCP/IP model correspond to specific OSI layers?

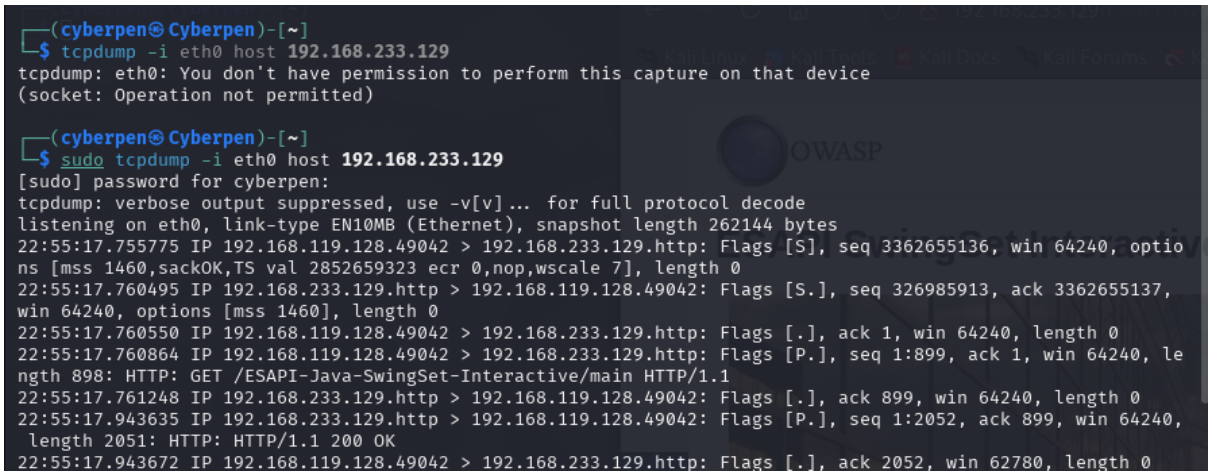
Here's the common correspondence between the 4-layer TCP/IP model and the 7-layer OSI model:

- TCP/IP Model: Application Layer
  - OSI Model:
    - Layer 7: Application Layer
    - Layer 6: Presentation Layer
    - Layer 5: Session Layer (The TCP/IP Application layer combines the functions of these top three OSI layers.)
- TCP/IP Model: Transport Layer
  - OSI Model:
    - Layer 4: Transport Layer (These layers are very similar in their functions.)
- TCP/IP Model: Internet Layer
  - OSI Model:
    - Layer 3: Network Layer (These layers are also very similar, with IP being the primary protocol in both.)
- TCP/IP Model: Network Access Layer
  - OSI Model:
    - Layer 2: Data Link Layer
    - Layer 1: Physical Layer (The TCP/IP Network Access layer bundles these two lowest OSI layers.)

Exercise 2: Capturing and Analyzing TCP Packets

3. Task: Initiate a connection to the OWASP Broken Web Application VM via HTTP or SSH. Capture TCP packets related to this connection using tcpdump or Wireshark.

Command: tcpdump -i eth0 tcp and host 192.168.56.101



4. Question:

What happens during the TCP handshake (SYN, SYN-ACK, ACK)?

TCP Handshake Process (SYN, SYN-ACK, ACK) according to my result, In the image, we can observe the three-way handshake used to initiate a TCP connection:

1. SYN – A packet with the Flags [S] is sent from client 10.0.2.15 to server 172.20.10.4 on port 80 (HTTP), initiating a connection.
2. SYN-ACK – The server responds with a Flags [S.] packet (SYN + ACK) acknowledging the SYN request and sending its own SYN.
3. **ACK** – The client responds with `Flags [.]`, acknowledging the SYN-ACK from the server. This completes the handshake.

Identify and describe the flags used in TCP communication (SYN, ACK, FIN, etc.).

Common TCP Flags

Flag	Description
S	SYN – Initiates a connection.
.	ACK – Acknowledges received data or connection response.
S.	SYN-ACK – Server response to SYN.
P	PSH – Push data to the receiving application immediately
F	FIN – Terminate connection.
R	RST – Reset the connection (not seen in your screenshot).

How does the TCP connection maintain reliability during transmission?

TCP uses multiple mechanisms to maintain reliable communication:

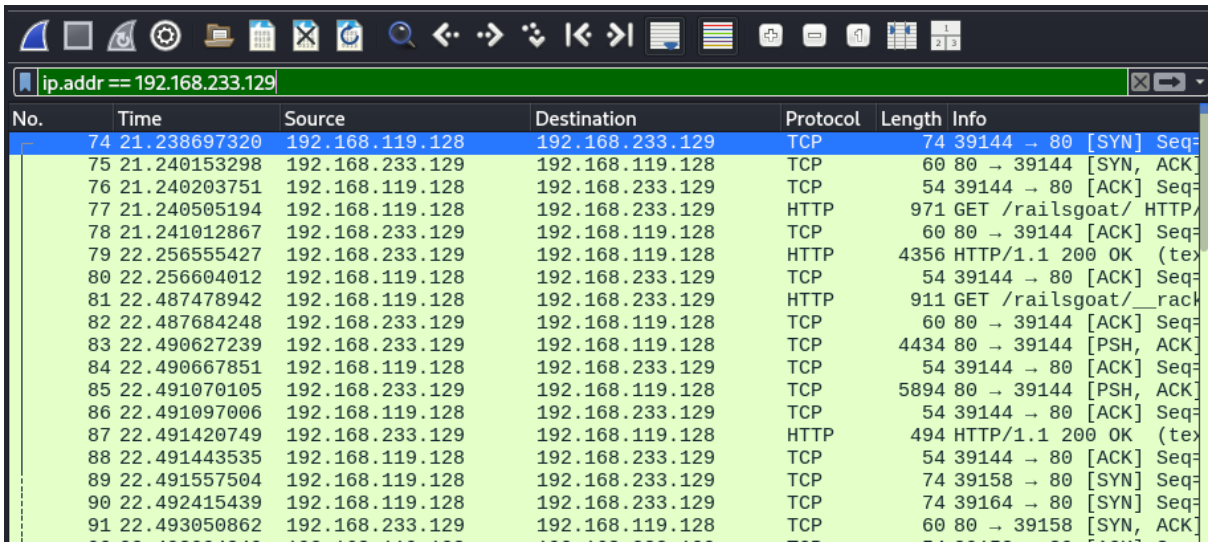
1. Acknowledgments (ACKs) – Each packet received must be acknowledged by the recipient.
2. Sequence Numbers – These help reassemble data in the correct order.
3. Retransmission – Lost packets (not acknowledged) are resent.
4. Checksums – Ensure data integrity during transit.
5. Flow Control (Window Size) – Prevents overwhelming the receiver.
6. Congestion Control – Avoids network congestion through slow start, congestion avoidance, etc.

Exercise 3: Investigating IP Packets (Network Layer)

5. Task: Capture and analyze the IP packets from your connection to the OWASP

Broken Web Application.

Command: Use Wireshark with the filter ip.addr == 192.168.233.129



6. Question:

What fields can you see in the IP packet header (e.g., Source IP, Destination IP, TTL, etc.)?

While the Wireshark summary pane (as shown in the image) doesn't display all IP header fields explicitly in each line, a detailed view of any individual packet (by clicking on it) in Wireshark would reveal the following common and significant IP packet header fields:

1. Version: (Usually 4 for IPv4 or 6 for IPv6) Indicates the version of the IP protocol.
2. Header Length (IHL): Specifies the length of the IP header in 32-bit words.

3. Differentiated Services Code Point (DSCP) / Type of Service (ToS): Used for Quality of Service (QoS), indicating how the packet should be handled (e.g., priority).
4. Total Length: The total length of the IP packet, including both header and data, in bytes.
5. Identification: A unique identifier assigned to each IP packet. Used for reassembling fragmented packets.
6. Flags: Control flags related to fragmentation (e.g., "Don't Fragment," "More Fragments").
7. Fragment Offset: Indicates the position of a fragment within the original unfragmented IP packet.
8. Time To Live (TTL): A counter that decrements by one each time the packet passes through a router (hop). When TTL reaches zero, the packet is discarded.
9. Protocol: Indicates the protocol of the data carried in the IP packet's payload (e.g., 6 for TCP, 17 for UDP, 1 for ICMP).
10. Header Checksum: Used to detect errors in the IP header.
11. Source IP Address: The IP address of the sender of the packet (e.g., 192.168.119.128 or 192.168.233.129 in your capture).
12. Destination IP Address: The IP address of the intended recipient of the packet (e.g., 192.168.233.129 or 192.168.119.128).
13. Options (Optional): Optional fields that provide additional features, though rarely used in standard IP traffic.

What is the significance of each of these fields?

1. Version: Essential for the receiving host to interpret the rest of the IP header correctly (IPv4 vs. IPv6 header formats are different).
2. Header Length (IHL): Allows the receiving device to determine where the IP header ends and the data payload begins.
3. DSCP / ToS: Enables network devices (routers) to prioritize certain types of traffic (e.g., voice or video traffic over regular web Browse) for better performance.
4. Total Length: Informs the receiver about the total size of the datagram, which is crucial for buffer allocation and processing.
5. Identification, Flags, Fragment Offset: These fields are vital for IP fragmentation. If a packet is too large for a network segment's Maximum Transmission Unit (MTU), it can be broken into smaller fragments. These fields allow the destination host to correctly reassemble the fragments into the original packet.
6. Time To Live (TTL): Prevents packets from looping indefinitely in a network due to routing errors. When TTL reaches 0, the packet is dropped, and an ICMP "Time Exceeded" message is usually sent back to the source. This is also how traceroute works, by manipulating TTL.
7. Protocol: Crucial for the IP layer to know which higher-layer protocol (e.g., TCP, UDP, ICMP) the data payload belongs to, so it can hand the payload up to the correct process on the receiving host.
8. Header Checksum: Provides basic error detection for the IP header. If the checksum doesn't match, the packet is usually discarded, assuming the header is corrupted.

9. Source IP Address: Identifies the originating host, allowing the recipient to know who sent the packet and for replies to be routed back correctly.
10. Destination IP Address: Identifies the intended recipient host, enabling routers to forward the packet along the correct path across the network.
11. Options: Allows for extensibility and provides additional functions like security, record route, and timestamping (though rarely used now compared to standard headers).

How does IP routing work in this scenario?

How IP Routing Works Generally:

IP routing is the process of moving IP packets from a source host to a destination host, potentially across multiple interconnected networks. When a host wants to send an IP packet:

1. It checks its routing table.
2. If the destination IP is on the same local network segment (determined by comparing the destination IP with its own IP and subnet mask), the packet is sent directly to the destination's MAC address (resolved via ARP).
3. If the destination IP is on a different network, the packet is sent to its default gateway (a router).
4. The router receives the packet, inspects the destination IP address, and uses its own routing table to determine the next hop (another router or the final destination) that brings the packet closer to its destination.
5. This process of forwarding packets from router to router continues until the packet reaches the network segment of the destination host, where it is then delivered directly.

In this Scenario (from my tcpdump):

- system's IP: 192.168.119.128 (based on Source IP of initial packets)
- OWASP VM's IP: 192.168.233.129 (based on Destination IP)

Notice the difference in the third octet of the IP addresses: 119 vs. 233. This strongly suggests that my system (192.168.119.128) and the OWASP VM (192.168.233.129) are on different IP subnets.

Are there any hops between your system and the OWASP VM?

Yes, based on the differing subnets, it is almost certain there is at least one hop (router) between your system and the OWASP VM.

- Explanation: Since they are in different 192.168.x.x subnets, they cannot communicate directly at Layer 2 (Ethernet). Your system (192.168.119.128) would send packets destined for 192.168.233.129 to its configured default gateway (router).

This router would then forward the packets to the subnet where 192.168.233.129 resides, possibly through another router.

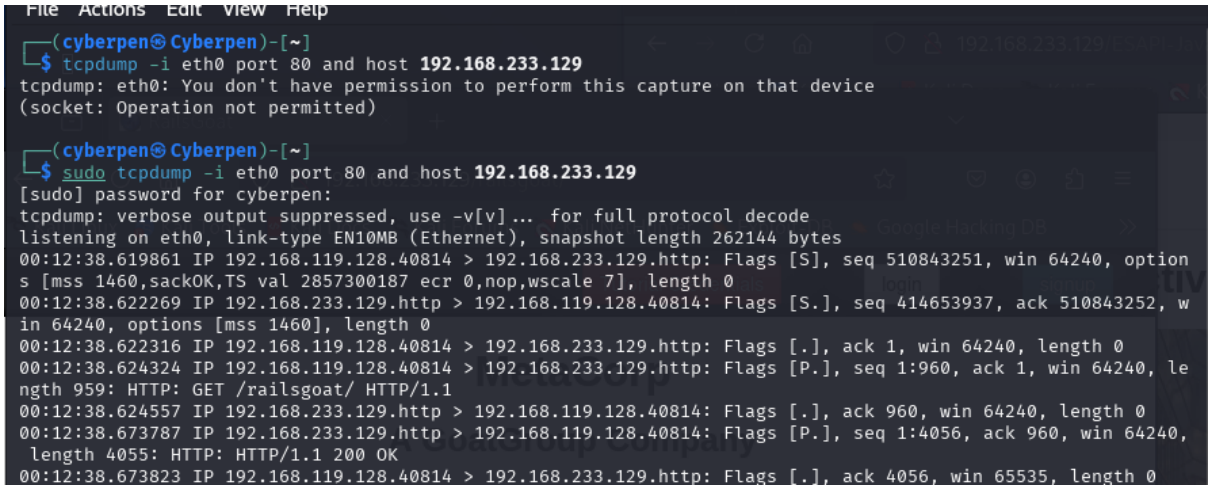
- Proof (without traceroute): While tcpdump itself doesn't explicitly show hop counts or intermediate routers (that's traceroute's job using TTL), the mere fact that 192.168.119.128 can communicate with 192.168.233.129 despite them being in different subnets confirms that routing is taking place, and thus, at least one router (hop) is involved. Each time a packet crosses a router, its TTL (Time To Live) value is decremented. If you were to look at the full Wireshark packet details, you would see the TTL values in both directions, and a traceroute command would explicitly list the intermediate routers (hops) and their latency.

In a virtualized lab environment, this single hop is often the virtual network router provided by the virtualization software (like VirtualBox or VMware) that connects different virtual networks.

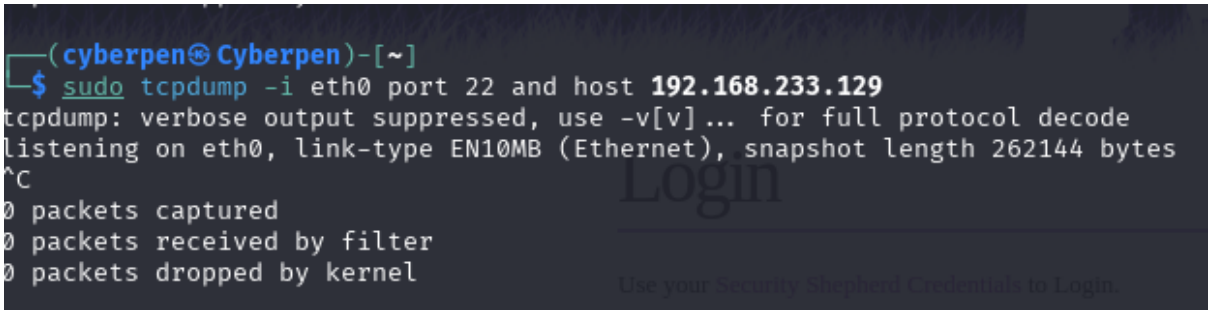
Exercise 4: Application Layer Analysis (HTTP/SSH Traffic)

7. Task: If you're using HTTP, filter the packets to capture only HTTP traffic using Wireshark or tcpdump. If you're using SSH, capture and analyze the encrypted traffic.

o Command for HTTP traffic: tcpdump -i <interface> port 80 and host <OWASP_IP>



o Command for SSH traffic: tcpdump -i <interface> port 22 and host <OWASP_IP>



8. Question:

Analyze the HTTP packets. What information is available in the HTTP request and response?

Based on the HTTP tcpdump output, we can extract significant information from both HTTP requests and responses:

Information from HTTP Request Packets (Client 192.168.119.128 to Server 192.168.233.129):

Let's take this example line:00:12:38.624324 IP 192.168.119.128.40814 > 192.168.233.129.http: Flags [P.], seq 1:960, ack 1, win 64240, length 959: HTTP: GET /railsgoat/ HTTP/1.1

1. Source and Destination:
   - Source IP: 192.168.119.128 (your system's IP address).
   - Source Port: 40814 (the ephemeral port chosen by your client for this connection).
   - Destination IP: 192.168.233.129 (the OWASP Broken Web Application VM's IP address).
   - Destination Port: http (which resolves to TCP port 80, the standard for HTTP).
2. TCP Flags: [P.] indicates a PUSH flag (meaning the sender wants the data delivered to the application immediately) and an ACK (acknowledgment) flag.
3. Sequence and Acknowledgment Numbers: seq 1:960, ack 1. These are TCP sequence and acknowledgment numbers used for reliable data transfer and ordering. seq 1:960 indicates the byte range of the data in this segment. ack 1 acknowledges data received from the server.
4. Payload Length: length 959 bytes. This indicates the size of the HTTP request payload being sent.
5. HTTP Method: GET. This specifies the action the client wants to perform on the server. GET is used to request data.
6. Requested URI/Path: /railsgoat/. This is the specific resource (e.g., a web page, an image, a script) that the client is requesting from the server.
7. HTTP Version: HTTP/1.1. The version of the HTTP protocol used for communication.We also see subsequent GET requests for other resources like

/railsgoat/__rack/swfobject.js, /railsgoat/__rack/web_socket.js, and /railsgoat/__rack/livereload.js, showing how a web page loads multiple components.

Information from HTTP Response Packets (Server 192.168.233.129 to Client 192.168.119.128):

Let's take this example line:

00:12:38.673787 IP 192.168.233.129.http > 192.168.119.128.40814: Flags [P.], seq 1:4056, ack 960, win 64240, length 4055: HTTP: HTTP/1.1 200 OK

1. Source and Destination:
   - Source IP: 192.168.233.129 (OWASP VM).
   - Source Port: http (TCP port 80).
   - Destination IP: 192.168.119.128 (your system).
   - Destination Port: 40814 (the client's ephemeral port).
2. TCP Flags: [P.] (Push, Acknowledgment).
3. Sequence and Acknowledgment Numbers: seq 1:4056, ack 960. The seq 1:4056 indicates the byte range of the data being sent in the response. The ack 960 acknowledges receipt of the client's previous data (the GET request, which was 959 bytes and started at seq 1, so 1+959=960).
4. Payload Length: length 4055 bytes. This is the size of the HTTP response body (e.g., the HTML content of the web page).
5. HTTP Version: HTTP/1.1.
6. Status Code and Message: 200 OK. This indicates that the server successfully processed the request and is providing the requested resource. Other common status codes include 404 Not Found, 500 Internal Server Error, etc.

In summary for HTTP: HTTP requests convey the client's intent and specific resource needs. While HTTP responses communicate the outcome of that request and deliver the requested data.

For SSH traffic, what is the significance of encrypted packets? Can you analyze the payload?

Since your tcpdump captured 0 packets for SSH, we'll answer this theoretically.

- Significance of Encrypted Packets:

    - Confidentiality: This is the paramount significance. SSH (Secure Shell) is designed for secure remote access. Encryption ensures that all data exchanged after the initial key exchange (including your username, password, commands typed, command output, and file transfers) is scrambled and unreadable to

anyone sniffing the network. This prevents eavesdropping and protects sensitive information from unauthorized disclosure.

- Integrity: Beyond confidentiality, SSH also employs cryptographic mechanisms to ensure data integrity. This means that if any part of the encrypted packet's payload is altered during transit, the receiving SSH client or server will detect the tampering and reject the data, thus protecting against malicious modification.
- Authentication: While distinct from the ongoing payload encryption, SSH's robust authentication mechanisms (like public-key authentication or password authentication) are crucial for verifying the identity of both the client and the server, ensuring you're connecting to the legitimate server and not an impostor.

Can you analyze the payload?

- No, not directly, without the decryption key. The very purpose of SSH encryption is to make the payload unreadable to passive observers. If you were to open an SSH packet in Wireshark (if you had captured one), the payload section would appear as unintelligible, random-looking binary data.
- What you can analyze (metadata): Even with encryption, you can still observe information from the lower layers, such as:
    - Source and Destination IP addresses.
    - Source and Destination ports (SSH uses TCP port 22 by default).
    - Packet sizes and timestamps (which can give clues about activity levels or types of operations, e.g., a large stream of packets might indicate a file transfer).
    - TCP flags (SYN, ACK, PUSH, FIN) related to connection establishment and termination.
    - The initial SSH handshake (before full encryption takes over) might reveal protocol versions and supported ciphers.
- Decryption (in controlled environments): In highly controlled lab settings, if you have access to the SSH server's private key (and sometimes client keys), advanced tools like Wireshark can be configured to decrypt SSH traffic after it's captured. However, this is not feasible or ethical for connections you don't own or control.

How does the application layer play a role in data exchange between your system and the OWASP VM?

The Application Layer, at the top of the TCP/IP model, is paramount to the data exchange because it's where the user's intent and the actual purpose of network communication are translated into network protocols.

1. Protocol Definition and Standardization:
    - It defines the specific rules and formats for how applications communicate. For example, HTTP (used for web Browse) specifies how a web browser requests a page (GET /railsgoat/ HTTP/1.1) and how a web server responds (HTTP/1.1 200 OK). This standardization allows different applications (e.g., Firefox, Chrome, Apache, Nginx) to interoperate seamlessly.
    - Without Application Layer protocols like HTTP, the browser wouldn't know how to ask for web content, and the web server wouldn't know how to deliver it.

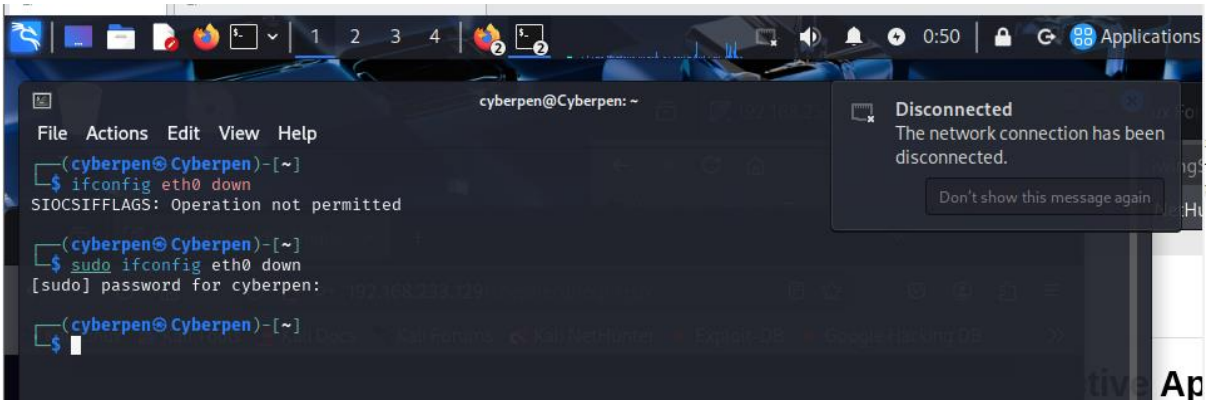
2. **User Interaction and Service Provision:**
    - The Application Layer directly interacts with the end-user applications. When you type http://192.168.233.129/railsgoat/ into your browser, it's the browser (an application) leveraging the HTTP protocol (Application Layer) to generate the GET request.
    - Similarly, if you used SSH, the SSH client application would use the SSH protocol to authenticate you and send your commands to the SSH server application on the VM.
3. Data Formatting and Payload Interpretation:
    - It's responsible for formatting the data that the application wants to send into a standard format that the receiving application can understand. For HTTP, this means structuring the GET request with its URI and version. For responses, it means ensuring the HTML, CSS, or image data is correctly formatted.
    - Conversely, on the receiving side, the Application Layer process (e.g., the web server) interprets the incoming data according to the protocol rules and passes the meaningful content (e.g., the requested file, the command to execute) to the higher-level application logic.
4. Initiation and Management of Application-Specific Sessions:
    - While the Transport Layer (TCP) manages the underlying connection, the Application Layer determines when an application-specific session begins (e.g., starting a web Browse session) and effectively ends (e.g., closing a web page, logging out of SSH).

In essence, the Application Layer bridges the gap between what a user wants to do and how that action is translated into network-understandable communication. It provides the "language" and structure for specific network services like web Browse, email, file transfer, and remote administration.

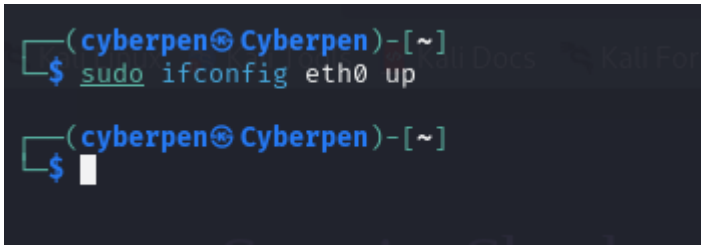
Exercise 5: Error Handling in TCP/IP Transmission

9. Task: Simulate packet loss or network issues by introducing a delay or stopping the network connection momentarily. Capture how TCP/IP handles this issue.

Command: Temporarily disable your network interface using ifconfig eth0 down



Command: To re-enable your network interface using ifconfig eth0 up



10. Question:

What happens when packets are dropped or delayed?

System Response: Immediately after the command is executed (or very shortly thereafter), a notification bubble appears on the top right of the screen stating:

- "Disconnected"
- "The network connection has been disconnected."

This visual confirmation directly demonstrates what happens when you disable a network interface. It confirms the loss of network connectivity

How does TCP ensure data reliability in the presence of errors?

TCP ensures data reliability through a combination of mechanisms:

1. Sequence Numbers: Every byte of data sent by TCP is assigned a sequence number. This allows the receiver to identify the order of bytes, detect missing segments, and correctly reassemble the data stream.

2. Acknowledgments (ACKs): The receiver sends ACKs to the sender, confirming the successful receipt of data up to a specific sequence number. This is a form of positive acknowledgment.
3. Retransmission Timers (Timeout and Retransmit): The sender maintains a timer for each segment it sends. If an ACK for that segment isn't received before the timer expires, the sender assumes the segment was lost and retransmits it.
4. Flow Control (Sliding Window): The receiver advertises its "receive window size" to the sender. This tells the sender how much buffer space the receiver has available. The sender will not send more data than the receiver's advertised window, preventing the receiver from being overwhelmed.
5. Congestion Control: TCP dynamically adjusts its transmission rate based on perceived network congestion. If packet loss (detected via retransmissions or duplicate ACKs) occurs, TCP reduces its sending rate (e.g., by decreasing its "congestion window") to alleviate network stress.
6. Checksums: TCP uses a checksum field in its header to detect corruption of the header and data during transmission. If a checksum mismatch occurs, the packet is typically discarded by the receiver, which then indirectly triggers a retransmission from the sender due to the lack of an ACK.
7. Three-Way Handshake: Establishes a reliable connection by synchronizing initial sequence numbers and agreeing on communication parameters before data transfer begins.
8. Four-Way Handshake (or RST): Provides a graceful way to terminate a connection, ensuring all data is delivered before closing.

How do retransmissions and sequence numbers work in TCP to maintain a proper data flow?

Retransmissions and sequence numbers are fundamental to TCP's reliability:

1. Sequence Numbers:
   - When a TCP sender sends a segment, it includes the sequence number of the first byte of data in that segment.
   - The sender also keeps track of the next sequence number it expects to send.
   - The receiver uses these sequence numbers to:
     - Order packets: If packets arrive out of sequence, the receiver can buffer them and reassemble them correctly.
     - Detect missing packets: If the receiver receives a segment with a sequence number that's far ahead of what it expects, it knows an earlier segment is missing.
2. Acknowledgments (ACKs) and Expected Sequence Numbers:
   - The receiver sends an ACK to the sender, which includes an acknowledgment number. This acknowledgment number signifies the next sequence number that the receiver expects to receive. Essentially, it confirms that all bytes up to (but not including) that acknowledgment number have been successfully received.
3. Retransmissions (Timeout and Fast Retransmit):
   - Timeout-based Retransmission: The sender maintains a timer for each unacknowledged segment. If this timer expires before the corresponding ACK is

received, the sender concludes that the segment (or its ACK) was lost and retransmits that segment. The RTO value is dynamically adjusted; it increases (exponentially) with successive retransmissions for the same segment.

- Fast Retransmit: This mechanism speeds up retransmission. If the sender receives three duplicate ACKs for the same segment, it assumes that the segment immediately following the acknowledged data has been lost (even if its timer hasn't expired yet). It then immediately retransmits that suspected lost segment without waiting for the RTO. This is common when one segment is lost, but subsequent segments from the sender continue to arrive, causing the receiver to repeatedly acknowledge the last in-order byte received.

Working Together for Data Flow:

- The sender continuously sends data segments, each with a sequence number.
- The receiver continuously sends ACKs, telling the sender what it has successfully received and what it expects next.
- If there's a gap in the sequence numbers or a timeout, the sender retransmits.
- This continuous interplay of sending data, acknowledging receipt, and retransmitting lost data ensures that the data stream is delivered reliably, in order, and without gaps, maintaining a proper and continuous data flow from the source to the destination, even in the face of network imperfections.

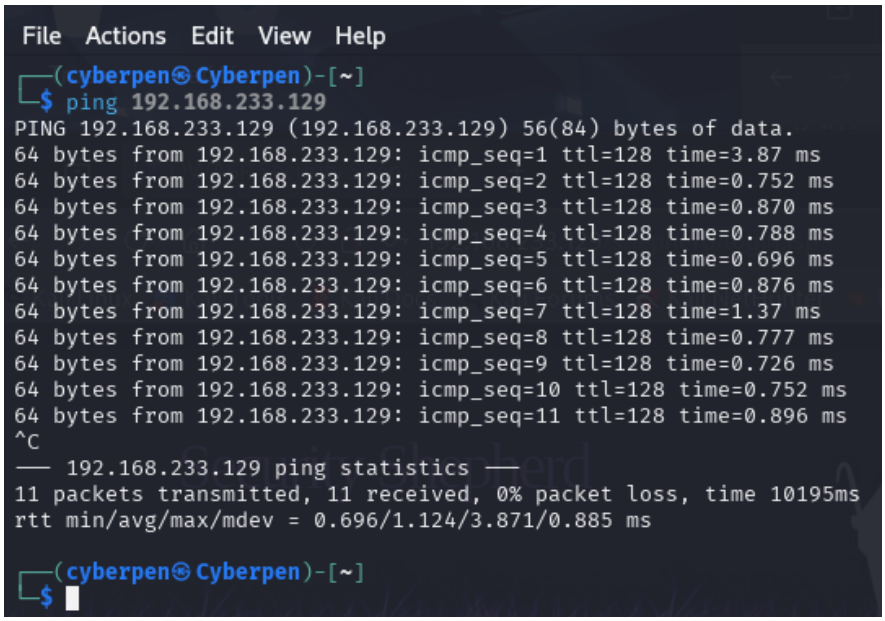
Exercise 6: ICMP and Ping Inspection (Network Layer)

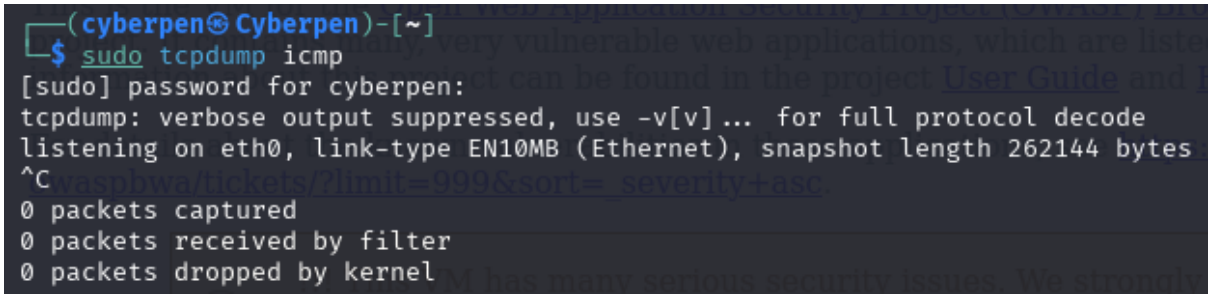
11. Task: Send a series of ping commands to the OWASP Broken Web Application

VM and capture the ICMP packets.

Command: ping <OWASP_IP> and use Wireshark or tcpdump to capture ICMP packets with the filter icmp.

ping 192.168.233.129





12. Question:

What are the key fields in an ICMP packet (e.g., Type, Code, Checksum)?

ICMP messages are typically encapsulated within IP packets. The ICMP header itself is relatively simple and includes the following key fields:

1. Type: An 8-bit field that specifies the type of ICMP message. Common types include:
   - 0: Echo Reply (response to a ping)
   - 3: Destination Unreachable
   - 4: Source Quench (deprecated for congestion control)
   - 5: Redirect
   - 8: Echo Request (a ping)
   - 9: Router Advertisement
   - 10: Router Solicitation
   - 11: Time Exceeded (e.g., TTL expired)
   - 12: Parameter Problem
   - 13: Timestamp Request
   - 14: Timestamp Reply
   - 15, 16: Information Request/Reply (deprecated)

2. Code: An 8-bit field that specifies the subtype of the ICMP message. Its meaning depends on the Type field. For example, for Type 3 (Destination Unreachable), different codes specify why the destination is unreachable (e.g., network unreachable, host unreachable, port unreachable, fragmentation needed and DF bit set).
3. Checksum: A 16-bit field used for error checking of the ICMP message itself (header and data). It's calculated over the entire ICMP message and verified by the receiver.
4. Identifier (for Echo Request/Reply): A 16-bit field used by the sender to help match Echo Replies to Echo Requests. Often, this is the process ID (PID) of the ping program.
5. Sequence Number (for Echo Request/Reply): A 16-bit field that increments for each Echo Request sent. This allows the sender to match replies with their corresponding requests and detect dropped packets. (Visible as icmp_seq= in your ping output).
6. Data (Variable): This field contains the data payload of the ICMP message. For Echo Request/Reply, it usually contains arbitrary data (often a timestamp and a padding pattern) that is echoed back by the recipient.

How does ICMP assist in diagnosing network connectivity issues?

ICMP is an invaluable tool for network diagnostics, providing critical feedback about network health and connectivity:

1. Reachability and Latency (Ping - Type 8/0):
   - The ping command (which uses ICMP Echo Request/Reply) is the most basic and common diagnostic tool. By sending an Echo Request and expecting an Echo Reply, it confirms:
     - Whether a host is reachable (0% packet loss in your ping output).
     - The round-trip time (latency) to that host (time=0.696 ms to 3.87 ms in your ping output).
     - If replies are received, it indicates that the path between the source and destination is generally functional.

2. Path Tracing (Traceroute - Type 11):
   - traceroute (or tracert on Windows) uses ICMP Time Exceeded (Type 11, Code 0) messages. It sends packets with incrementally increasing TTLs. When a router receives a packet with TTL=1 and decrements it to 0, it sends an ICMP Time Exceeded back to the source. This allows traceroute to map the path (hops) a packet takes to a destination and measure the latency to each hop. This helps identify where connectivity issues or bottlenecks occur along a path.
3. Destination Unreachable (Type 3):
   - If a router or host cannot deliver a packet to its final destination, it sends an ICMP Destination Unreachable message back to the sender. The Code field further specifies the reason (e.g., Code 0: Network Unreachable, Code 1: Host Unreachable, Code 3: Port Unreachable, Code 4: Fragmentation Needed and Don't Fragment bit set). These messages are crucial for diagnosing misconfigurations, firewall blocks, or non-existent services.
4. Redirect (Type 5):
   - A router can send an ICMP Redirect message to a host, instructing it to send packets for a specific destination via a different (better) gateway on the same network segment. This helps optimize local routing.
5. Parameter Problem (Type 12):
   - Indicates that an IP header field contains an illegal value, helping diagnose issues with malformed packets or protocol incompatibilities.

In summary, ICMP serves as the network's "error reporting" and "information query" protocol, providing essential feedback to network engineers and tools to identify, localize, and troubleshoot connectivity, routing, and reachability problems.

What is the significance of TTL in both ICMP packets and general IP packets?

TTL (Time to Live) is a critical field in the IP header, and its significance applies to all IP packets, including those carrying ICMP.

1. Prevents Routing Loops:
    - This is the primary significance. In complex networks, especially the internet, routing errors can occur, leading to packets endlessly looping between routers. The TTL field acts as a "hop limit" or "packet lifespan."
    - Every time an IP packet passes through a router (a "hop"), the router decrements the TTL value by 1.
    - If a router receives a packet with a TTL of 1 and decrements it to 0, it discards the packet and sends an ICMP Time Exceeded (Type 11, Code 0) message back to the original source. This prevents packets from circulating indefinitely and consuming network resources.
2. Diagnosing Hops and Latency (Traceroute):
    - The traceroute utility (or tracert) cleverly exploits the TTL mechanism. It sends a series of packets, starting with a TTL of 1, then 2, then 3, and so on.
    - When a packet with TTL=1 reaches the first router, it decrements TTL to 0, discards it, and sends an ICMP Time Exceeded back to the source. This reveals the first hop.
    - When the packet with TTL=2 reaches the second router, it decrements TTL to 0, discards it, and sends an ICMP Time Exceeded back, revealing the second hop, and so on.
    - By doing this, traceroute can map the entire path a packet takes to a destination, identify each router (hop), and measure the round-trip time to each hop, which is incredibly useful for network path analysis and bottleneck detection.
3. Determining Packet Proximity (Ping):
    - When you ping a host, the returned ttl value in the ICMP Echo Reply can give you a rough idea of how many hops away the target host is. If your system is Linux-based, its default starting TTL is often 64. If the target is Windows, its default is often 128. If the returned TTL is close to the original TTL minus the number of known hops, it indicates a direct or near-direct connection.
    - In your ping output, ttl=128 suggests the OWASP VM might be running a Windows-like or similarly configured OS, and the packets are likely reaching it directly or with very few hops from a source that sets TTL to 128 (consistent with a virtual network directly connected to your host or via a simple virtual router).

TTL is a fundamental control mechanism at the IP layer, crucial for network stability and a powerful diagnostic indicator.

Exercise 7: Analyzing UDP Packets (Transport Layer)

13. Task: If available, generate some UDP traffic to the OWASP VM by using a simple application or UDP-based service. Capture and analyze the UDP packets.

o Command: tcpdump -i <interface> udp and host <OWASP_IP>

```
┌──(cyberpen㉿Cyberpen)-[~]
└─$ sudo  tcpdump -i eth0 udp and host 192.168.233.129
[sudo] password for cyberpen:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C
0 packets captured
0 packets received by filter
0 packets dropped by kernel

┌──(cyberpen㉿Cyberpen)-[~]
└─$
```

14. Question:

Compare UDP with TCP. What are the major differences in packet structure and behavior?

UDP vs. TCP: Packet Structure and Behavior

Feature	TCP	UDP
Connection	Connection-oriented	Connectionless
Connectionless	Reliable (ACKs, retransmission)	Unreliable
Header Size	20 bytes (min)	8 bytes
Overhead	Higher	Lower
Order	Guarantees packet order	No guarantee of order
Error Check	Yes, with retransmission	Yes, but no retransmission
Use Case	Web, Email, File Transfer	Streaming, VoIP, DNS, Games

Why does UDP not ensure reliability, and in what scenarios would you prefer UDP over TCP?

UDP does not ensure reliability because UDP does not implement:

- Handshakes (like TCP's 3-way)
- Acknowledgments (ACK)
- Sequence numbers for order
- Retransmission on failure

This makes UDP:

- Faster
- Less resource-intensive
- But unreliable for data integrity

scenarios I would prefer UDP over TCP?

- Speed is more important than reliability
- Low latency is critical
- Small, independent packets are sent

Examples:

- Streaming (video/audio)
- VoIP calls
- DNS lookups
- Online gaming
- TFTP

How does UDP manage data transmission without the need for acknowledgments or retransmissions?

UDP simply:

- Sends datagrams to the destination
- Relies on:
    - Application-layer protocols to manage loss if needed
    - Underlying network to deliver the data

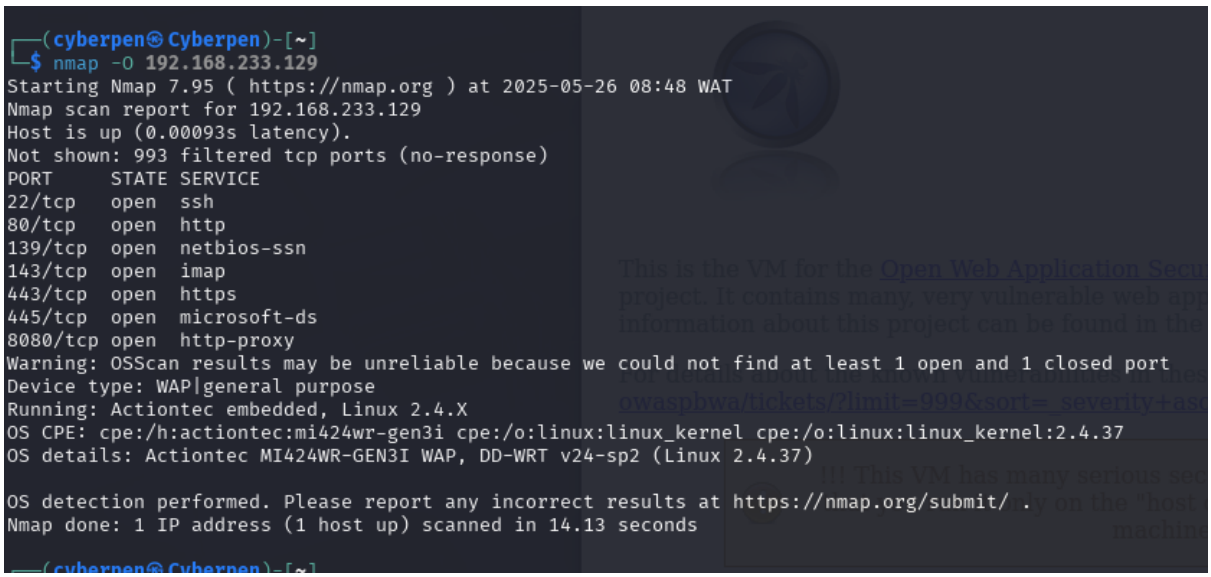
There is no built-in mechanism for:

- Confirmation of delivery
- Retransmission
- Flow control

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

15. Task: Use nmap to perform OS detection on the OWASP Broken Web Application VM, analyzing how nmap identifies the operating system based on packet behavior.

Command: nmap -O 192.168.233.129



17. Question:

How does nmap detect the OS based on the captured packets?

Nmap performs TCP/IP stack fingerprinting, which means:

- It sends a series of crafted packets to the target.
- Analyzes the responses (or lack thereof).
- Compares these to its internal OS database of known fingerprints.

These fingerprints are based on:

- TCP/UDP responses to specific flags.
- ICMP replies.
- TCP ISN (Initial Sequence Number) behavior.
- TCP options like SACK, Timestamps, etc.

What packet characteristics (TTL, window size, etc.) help in identifying the OS?

Key characteristics include:

Characteristic	Description
TTL (Time To Live)	Different OSes set different default TTL values (e.g., Linux often uses 64, Windows 128)
Window Size	OS-specific default TCP window sizes help distinguish systems
DF (Don't Fragment) bit	OS-specific use of the DF bit in IP headers
TCP Options	Order and presence of options like MSS, Window Scale, Timestamps
IP ID Sequence	How an OS increments its IP IDs (random, incremental, etc.)
Response to invalid flags	OS reactions to unusual flag combos (e.g., NULL, FIN, Xmas scans)

Explain why OS detection can be an important step in network analysis and vulnerability assessment.

Reason	Explanation
Targeted Exploits	Knowing the OS helps choose compatible exploits (e.g., Windows vs Linux)
Security Posture	OS version may reveal unpatched vulnerabilities or outdated software
Network Mapping	Helps understand roles (e.g., router, server, client) in the network
Risk Assessment	OS-specific risks can be evaluated (e.g., SMB vulnerabilities on Windows)
Service Behavior	OS affects default service configurations and open ports

Exercise 9: Analyzing ARP Traffic (Link Layer)

18. Task: Use arp-scan or tcpdump to capture ARP traffic in your local network, focusing on communication with the OWASP VM.

Command: arp-scan -I eth0 192.168.233.129

18. Question:

What information can you gather from ARP packets (e.g., MAC addresses)?

Information gather from ARP are as follows:

- IP ↔ MAC mapping: e.g. 192.168.233.129 is at 00:0c:29:8f:ca:00
- Packet type: ARP Request vs. ARP Reply
- Sender IP / Sender MAC and Target IP / Target MAC fields
- OUI/vendor lookup (if you have the vendor file)

How does the ARP protocol function at the link layer?

How ARP protocol function at the link layer is as follows:

1. Broadcast Request
   - A host needing a MAC for an IP sends a frame to ff:ff:ff:ff:ff:ff.
2. Unicast Reply
   - The owning host responds with its MAC in a direct (unicast) ARP reply.
3. ARP Cache Update
   - Both parties store the mapping in their ARP tables for future use.

All ARP traffic stays within the local broadcast domain; routers do not forward it.

What role does ARP play in the communication between your system and the OWASP VM?

The role played by ARP in the communication between my system and the OWASP VM are listed below:

- MAC discovery: Lets your Kali build proper Ethernet frames addressed to the OWASP VM's NIC.
- Enables IP communication: Without resolving the MAC, IP-layer packets can't be delivered on the LAN.
- Performance: Caching ARP entries reduces broadcast traffic on subsequent communications.

With this mapping in place, every ping, TCP, or UDP packet you send to the OWASP VM rides inside an Ethernet frame addressed to 00:0c:29:8f:ca:00.

Exercise 10: Troubleshooting Network Connectivity Using TCP/IP Knowledge

19. Task: Simulate a common network issue (e.g., incorrect subnet mask, gateway misconfiguration) and troubleshoot the issue using your understanding of the TCP/IP stack.

Simulation: Incorrect Subnet Mask on One Device

Device	IP Address	Subnet Mask	Default Gatew
PC1	192.168.1.10	255.255.255.0	192.168.1.1
PC2	192.168.1.20	255.255.0.0	192.168.1.1

- **Effect:**
  PC2's subnet mask is incorrectly set to 255.255.0.0 instead of 255.255.255.0. This causes PC1 and PC2 to perceive their networks differently.
- **Result:**
  PC1 and PC2 think they are on different networks and cannot communicate properly. For example, ping from PC1 to PC2 will fail or time out.

Troubleshooting Steps:

1. **Diagnose the Problem:**
   - From PC1, run ping 192.168.1.20 → No response.
   - Run tracert 192.168.1.20 (Windows) or traceroute 192.168.1.20 (Linux) → Unexpected routing or failure.
   - Check IP configurations on both PCs using ipconfig (Windows) or ifconfig/ip addr (Linux).
   - Notice the mismatch in subnet masks.

2. **Use Packet Inspection:**
    - Capture traffic with Wireshark on PC1 during the ping attempt.
    - Observe ARP requests: PC1 sends ARP requests for PC2's IP but doesn't receive replies because PC2 believes PC1 is outside its subnet.
    - ICMP echo requests fail at the network layer.

3. Resolution:
    - Correct the subnet mask on PC2 to 255.255.255.0.
    - After correction, ping from PC1 to PC2 succeeds.
    - ARP requests and replies work properly, enabling Layer 2 communication.

20. Question:

What is the issue you simulated, and how did it affect network communication?

- **Issue:** Incorrect subnet mask configured on PC2 (255.255.0.0 instead of 255.255.255.0).
- **Effect:** Devices incorrectly interpreted which IP addresses are local or remote, causing failed communication as PC1 and PC2 considered each other on different networks, blocking direct communication.

How did you diagnose and resolve the issue using packet inspection and knowledge of the TCP/IP layers?

- Diagnosis started with ping and traceroute tests failing.
- Checked IP configurations and found subnet mask mismatch.
- Packet inspection with Wireshark showed ARP requests sent by PC1 but no ARP replies from PC2 due to network mismatch.
- Understanding of the TCP/IP layers helped identify that Layer 3 (Network layer) subnetting errors were preventing Layer 2 (Data Link) communication.
- Resolved by correcting the subnet mask on PC2 to match PC1, restoring proper IP addressing and allowing successful ARP resolution and packet delivery.

What tools and techniques would you recommend for real-world network troubleshooting?

- Tools:
    - Ping to test basic connectivity.
    - Traceroute / Tracert to determine the path packets take.
    - Wireshark for packet capture and analysis.
    - ipconfig / ifconfig / ip addr to check and configure IP settings.
    - ARP tables inspection (arp -a) to verify MAC address resolution.
    - Network simulators like Cisco Packet Tracer for lab simulations.

- **Techniques:**
  - Start from physical connectivity checks.
  - Verify IP addressing and subnet masks.
  - Use ping and traceroute to isolate connectivity issues.
  - Capture packets to analyze traffic flow and detect where failures occur.
  - Use knowledge of TCP/IP layers to narrow down problem layers (Physical, Data Link, Network, Transport).
  - Document configuration and changes for future reference.