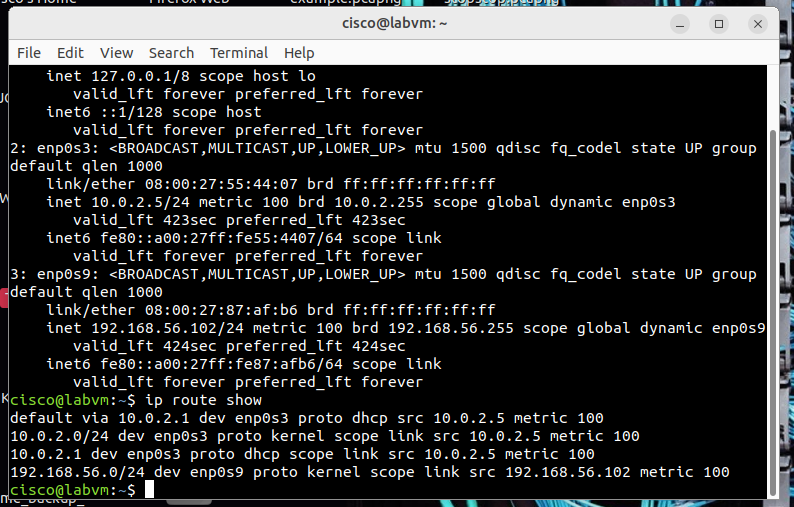
NAME: Adeyemi Akande

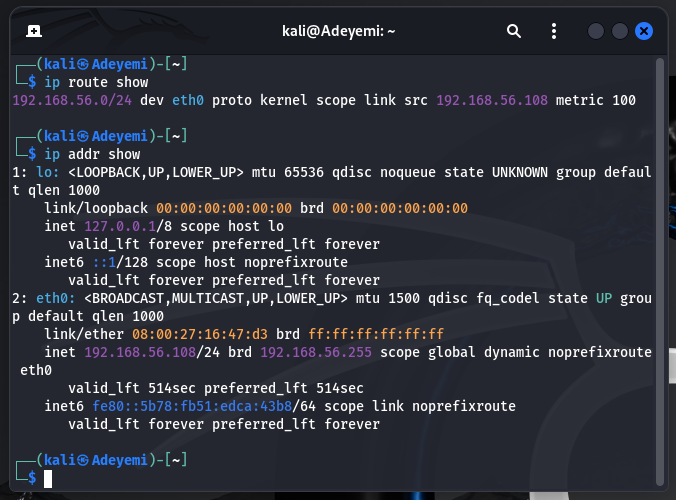
REG NO: 2024/INT/3800

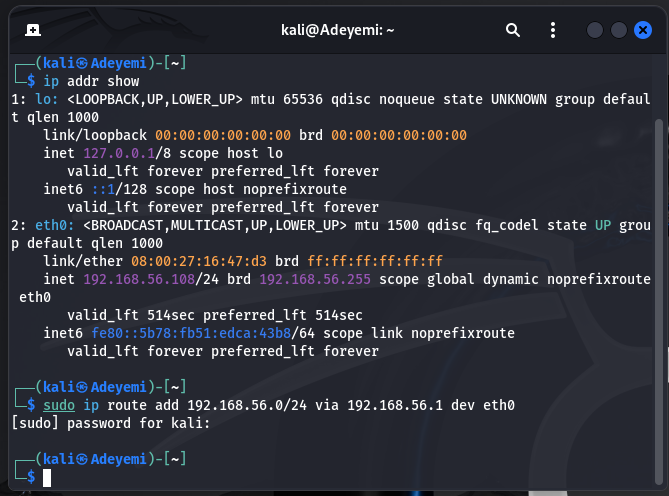
**INT303: Networking Fundamentals – Lab 4**

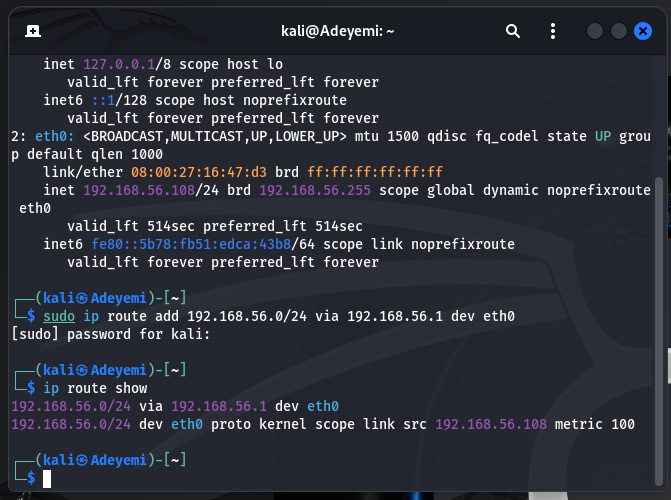
**Lab 4: Simulating Network Routing and VLAN Configuration in Linux**

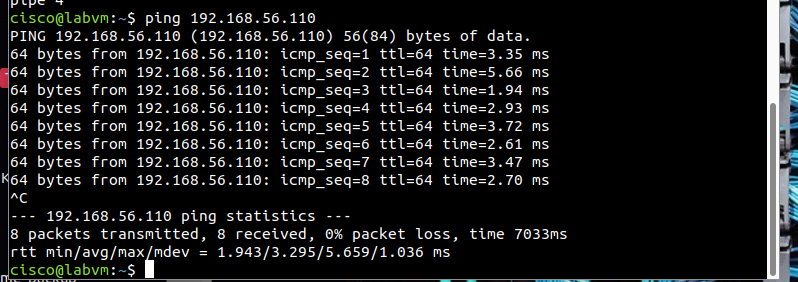
**Exercise 1: Setting Up Static Routing in Linux**

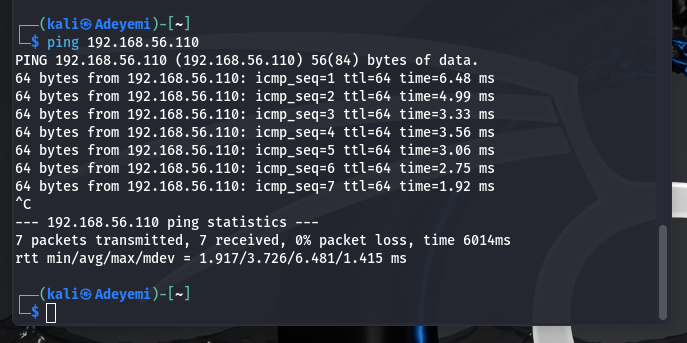












**How Static Routing Works on a Linux System**

Static routing in Linux involves manually configuring the routing table, which dictates how packets are forwarded to their destination. Here's an overview of the process:

1. **Routing Table**:
   * Linux maintains a routing table that maps destination networks to specific gateways (or interfaces).
   * Use ip route show to view the current table.
2. **Packet Forwarding**:
   * When a packet is sent, the kernel checks the routing table to determine the best route.
   * The route specifies:
     + **Destination network/subnet**: Where the packet is going.
     + **Gateway**: The next hop (router) for the packet.
     + **Network interface**: The interface to use for forwarding the packet.
3. **Static Route Commands**:
   * Use the ip route add command to specify static routes manually.
   * Static routes persist only for the current session unless saved in configuration files.
4. **IP Forwarding (If Acting as a Router)**:
   * To route packets between interfaces, IP forwarding must be enabled (net.ipv4.ip\_forward).

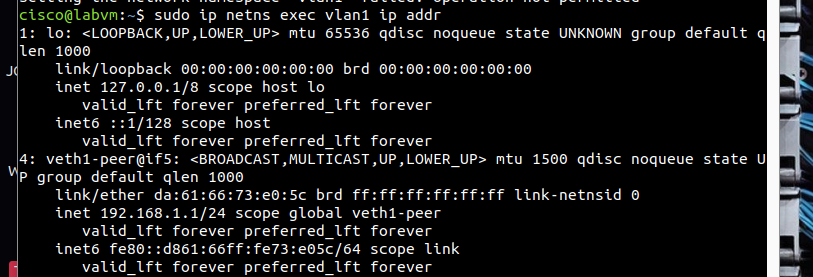
**Challenges When Setting Up Static Routes Manually**

1. **Human Error**:
   * **Incorrect Configuration**: Typing the wrong destination, gateway, or interface can lead to misrouted or dropped packets.
   * **Overlapping Routes**: Adding conflicting routes can cause unpredictable behavior.
2. **Network Changes**:
   * Static routes don't adapt to network topology changes, such as a gateway going offline or a new subnet being added.
3. **Route Persistence**:
   * Routes added manually are not persistent across reboots unless explicitly saved, requiring additional configuration.
4. **Scalability**:
   * Managing multiple routes across large networks is cumbersome and error-prone. Dynamic routing protocols (e.g., OSPF, BGP) are better suited for such environments.
5. **Debugging Complexity**:
   * If packets are not reaching their destination, identifying whether the issue is with the route, gateway, or another part of the network can be challenging.
6. **Gateway or Interface Dependency**:
   * Static routes rely on specific gateways or interfaces. If these go down or change IPs, the route breaks unless updated manually.
7. **Overhead for Redundancy**:
   * Static routing doesn't support automatic failover. If a gateway becomes unavailable, traffic won't be rerouted unless a new static route is configured.
8. **Administrative Burden**:
   * Each system in the network must be individually configured with appropriate routes, which is time-consuming.

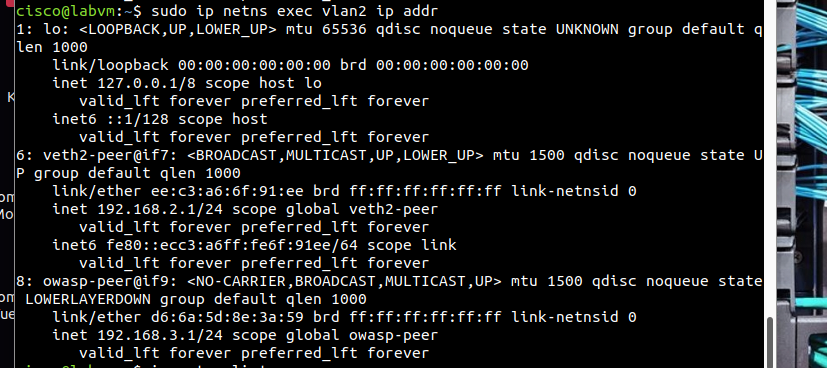
**Exercise 2: VLAN Configuration Using Network Namespaces**

**Vlan1 and vlan2 created and verified with: ip netns list in the image below**

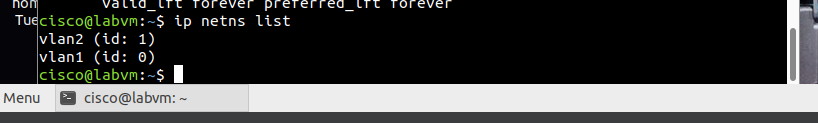
**Checking vlan1 config**

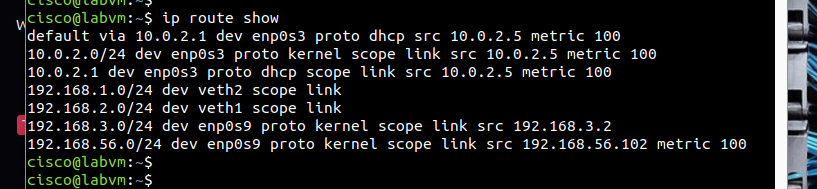


Checking vlan2 config

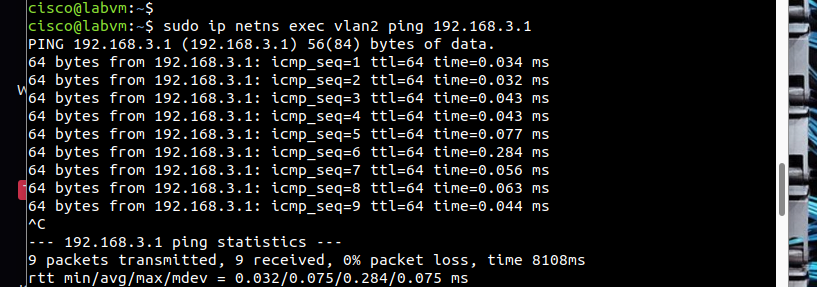


Vlan(s) creation confirmed





Pinging OWASP in the screenshot below:



### ****How Network Namespaces Simulate VLANs in Linux****

Network namespaces provide isolated environments for networking resources, effectively simulating the behavior of VLANs. Here's how they achieve this:

#### ****1. Network Isolation****

* Each namespace has its own instance of the network stack, including:
  + Interfaces
  + Routing tables
  + Firewall rules
  + ARP tables
* Packets sent within a namespace remain isolated unless explicitly routed to another namespace, akin to VLAN isolation.

#### ****2. Virtual Ethernet Interfaces****

* Virtual Ethernet (veth) pairs connect namespaces, behaving like a physical cable connecting two switches or devices in a VLAN.
* Each end of the veth pair exists in a different namespace, enabling controlled communication between namespaces.

#### ****3. IP Addressing and Subnets****

* Within a namespace, IP addresses and subnets can be assigned just like in a VLAN.
* For example:
  + Namespace vlan1: 192.168.1.0/24
  + Namespace vlan2: 192.168.2.0/24
* This segmentation mirrors the behavior of VLANs, where devices in different VLANs cannot communicate without routing.

#### ****4. Bridge Configuration for Connectivity****

* A Linux bridge can connect multiple namespaces, simulating a physical VLAN-aware switch.
* For example:
  + Namespaces connected to the same bridge are part of the same broadcast domain.
  + Traffic between namespaces on the bridge respects VLAN-like tagging or isolation.

#### ****5. Routing Between Namespaces****

* Using routing or NAT, traffic between namespaces can be managed, similar to a router connecting VLANs.

### ****Benefits of Using Network Namespaces for Network Isolation****

#### ****1. Strong Isolation****

* Each namespace has its own isolated networking stack, ensuring processes in one namespace cannot interfere with another.
* This makes it ideal for scenarios like:
  + Multi-tenant environments
  + Testing and development

#### ****2. Lightweight and Efficient****

* Unlike traditional VLANs requiring physical switches or hypervisors, namespaces are entirely software-based.
* They are lightweight, requiring minimal resources compared to virtual machines.

#### ****3. Flexible and Programmable****

* Namespaces can be dynamically created, configured, and removed using simple tools like ip netns.
* They integrate seamlessly with tools like Docker, Kubernetes, and Open vSwitch.

#### ****4. Granular Control****

* Administrators can precisely control traffic, routing, and policies within each namespace.
* Network namespaces can be combined with features like:
  + **iptables** for firewalling
  + **tc** for traffic shaping
  + **policy-based routing**

#### ****5. Simulate Complex Network Topologies****

* Namespaces can mimic VLANs, subnets, routers, and firewalls, making them invaluable for:
  + Network emulation
  + Training and certification labs
  + Testing network configurations and software

#### ****6. Cost-Effective****

* No need for additional hardware or specialized VLAN-aware switches. Everything runs on a standard Linux system.

#### ****7. Integration with Modern Networking Tools****

* Namespaces are compatible with advanced technologies like:
  + **Container networking**: Docker and Kubernetes use namespaces for pod-level isolation.
  + **SDN (Software-Defined Networking)**: Tools like Open vSwitch leverage namespaces for network virtualization.

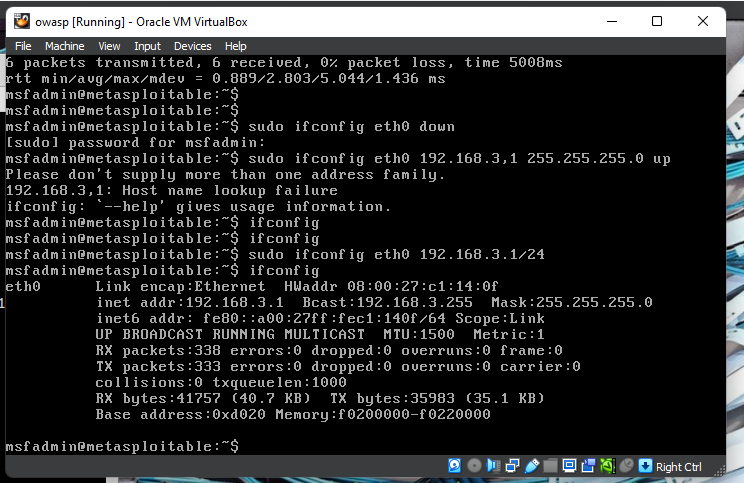
### ****Use Cases****

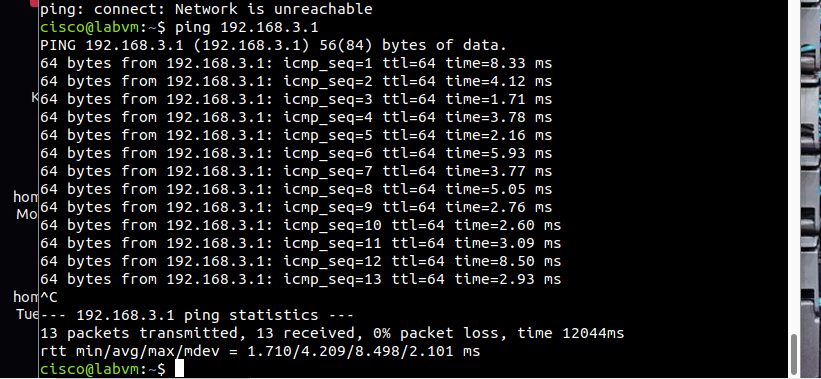
* **Testing**: Build isolated environments to test configurations without affecting production networks.
* **Multi-Tenant Solutions**: Create separate namespaces for each tenant in a shared environment.
* **Microservices**: Provide isolated network environments for individual containers.

In summary, network namespaces in Linux provide an efficient and flexible way to simulate VLANs, offering isolation and control without the need for specialized hardware. They’re particularly beneficial for development, testing, and containerized environments.

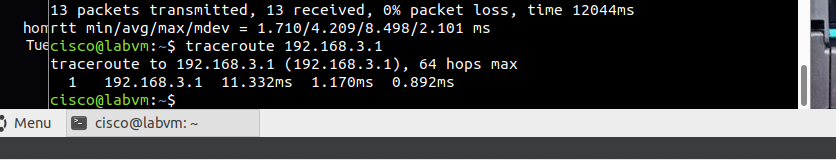
**Exercise 3: IP Address Assignment and Subnetting in Linux**

**Exercise 4: Testing Connectivity Using Ping and Traceroute**



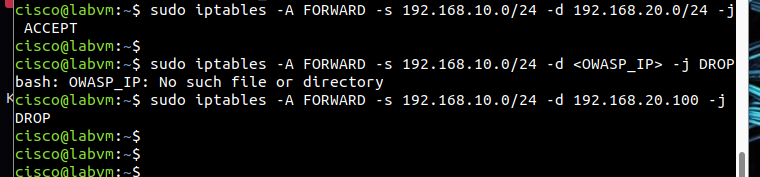


Traceroute screenshot



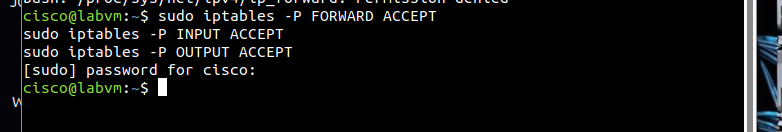
**Exercise 5: Configuring iptables for Routing and Firewall Rules**

**Initial configuration that was changed to make it go inline with my original network configuration**

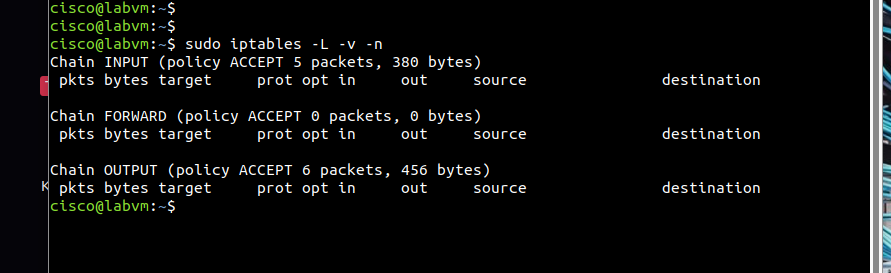


##### **Default iptables Policies**

Set default policies to allow traffic that matches the rules and drop anything else:



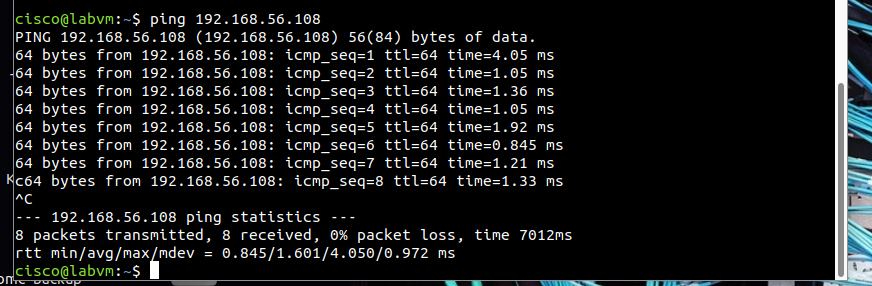
To verify the rules you’ve added:



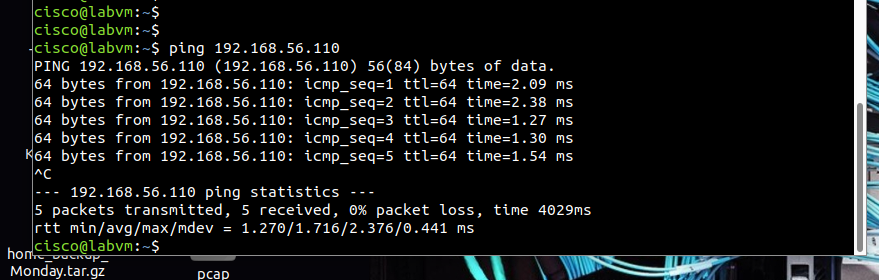
#### ****Test the Configuration****

Perform the following tests to verify the behavior of your setup:

##### **a. Ping Between Networks**



Ping the OWASP VM

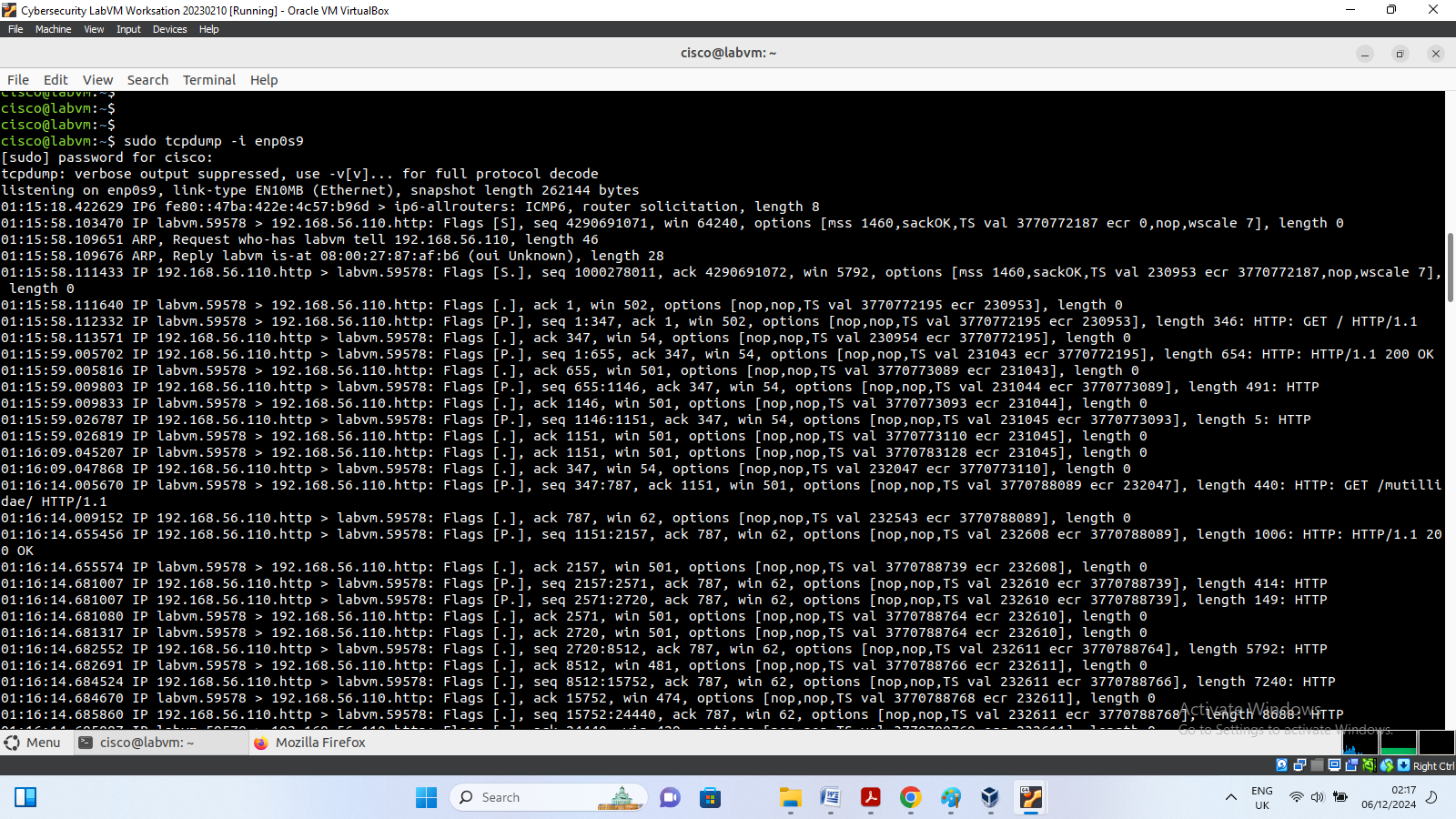


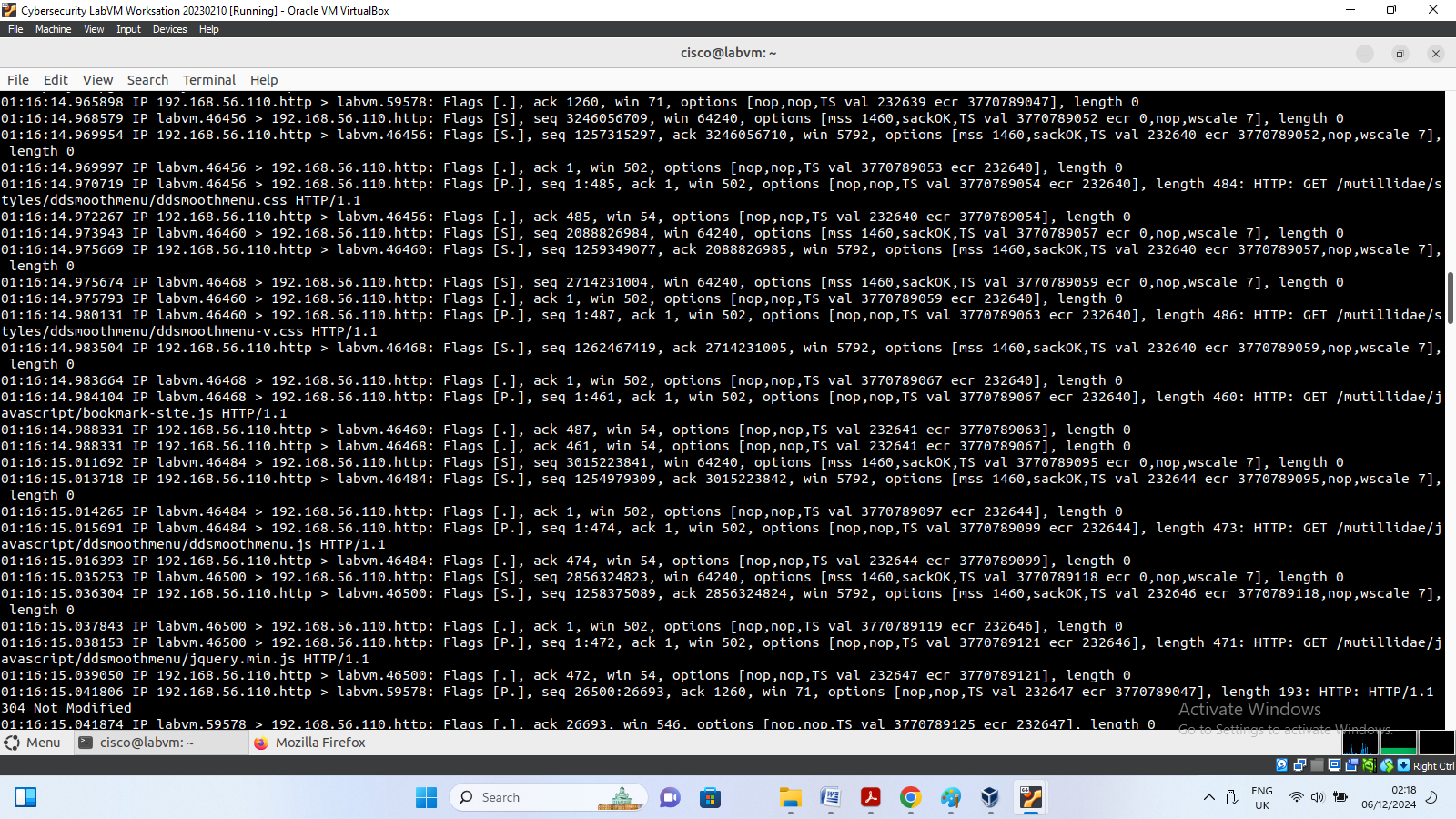
**Using iptables**:  
iptables can simulate routing by enabling IP forwarding and setting rules to control traffic flow between networks. It acts as a firewall, allowing, blocking, or redirecting traffic based on source, destination, and protocols.

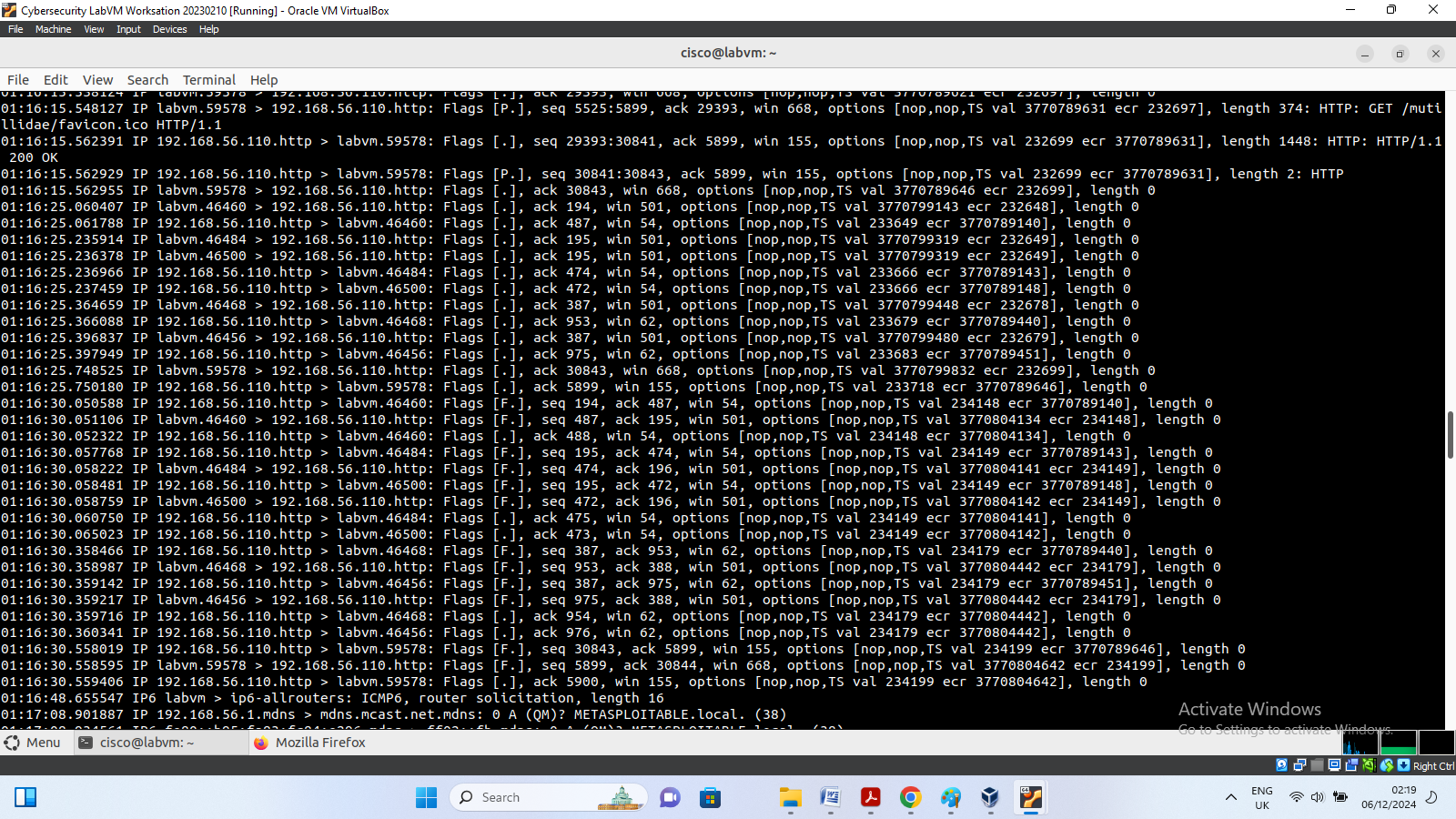
**Common mistakes**:

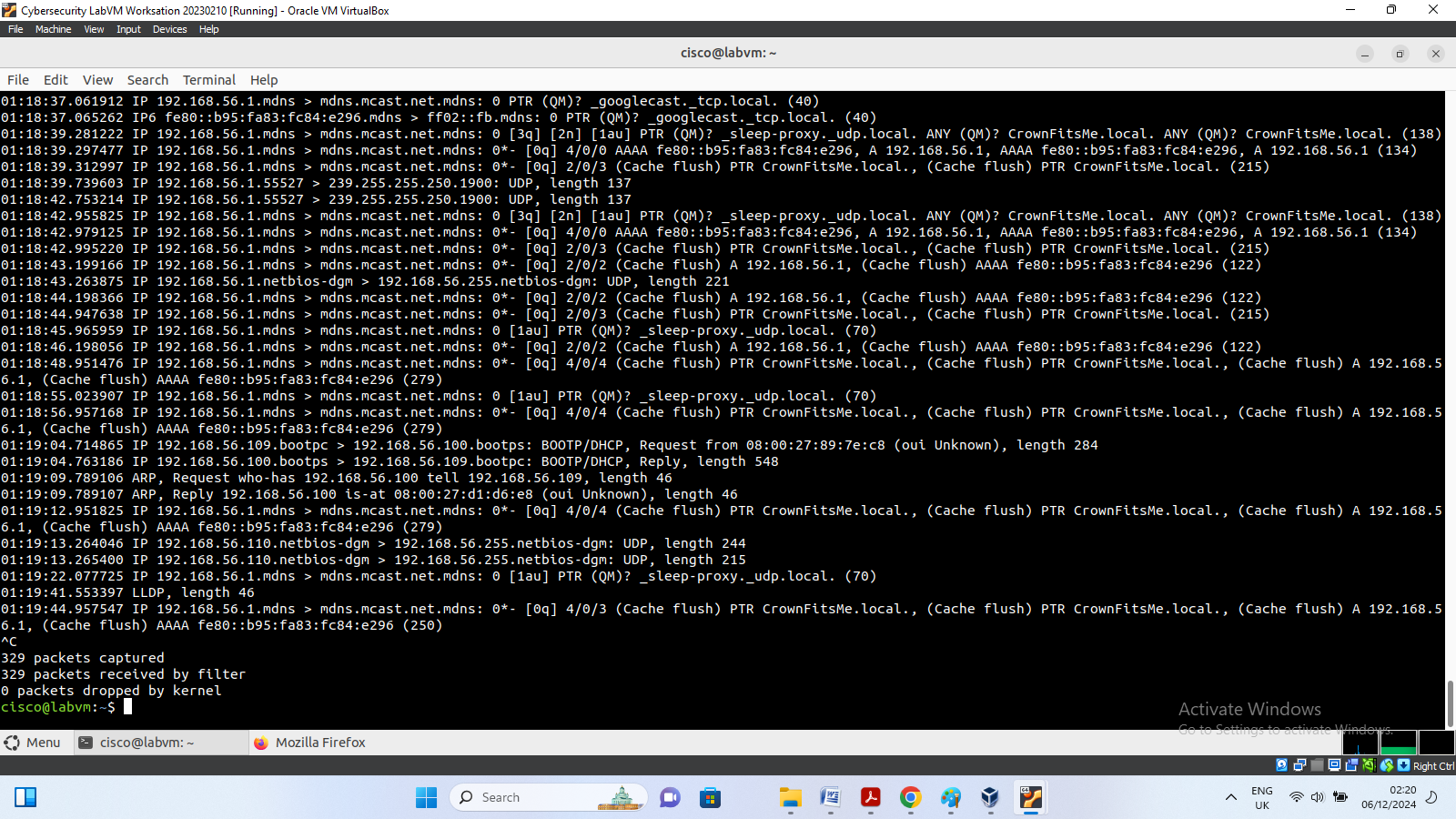
1. Forgetting to enable IP forwarding.
2. Incorrect rule order (iptables processes rules sequentially).
3. Missing default policies, leading to unintended traffic behavior.
4. Overlooking persistence (rules reset after reboot).
5. Blocking essential services (e.g., SSH).

**Exercise 6: Monitoring Traffic Using tcpdump**

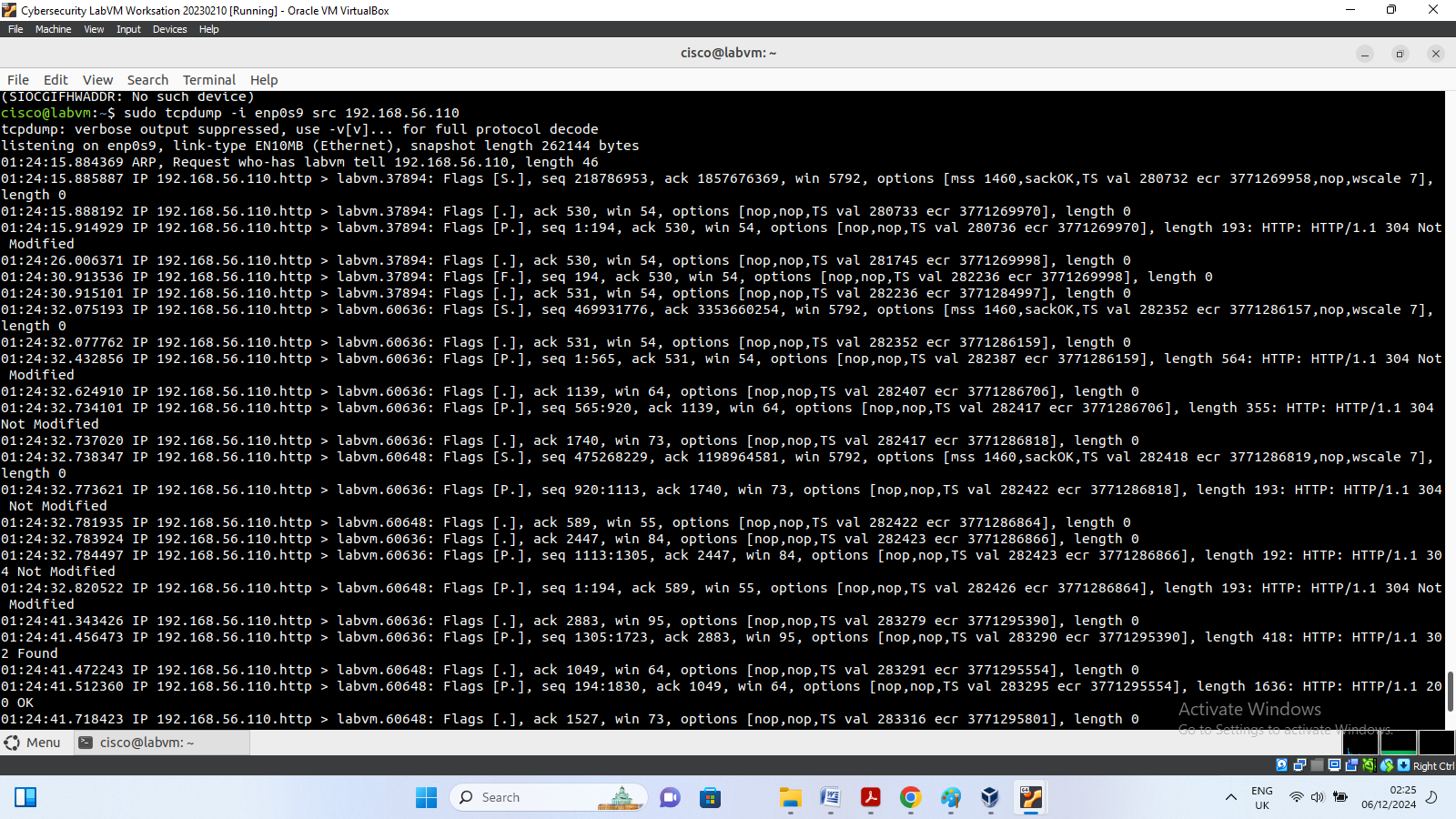


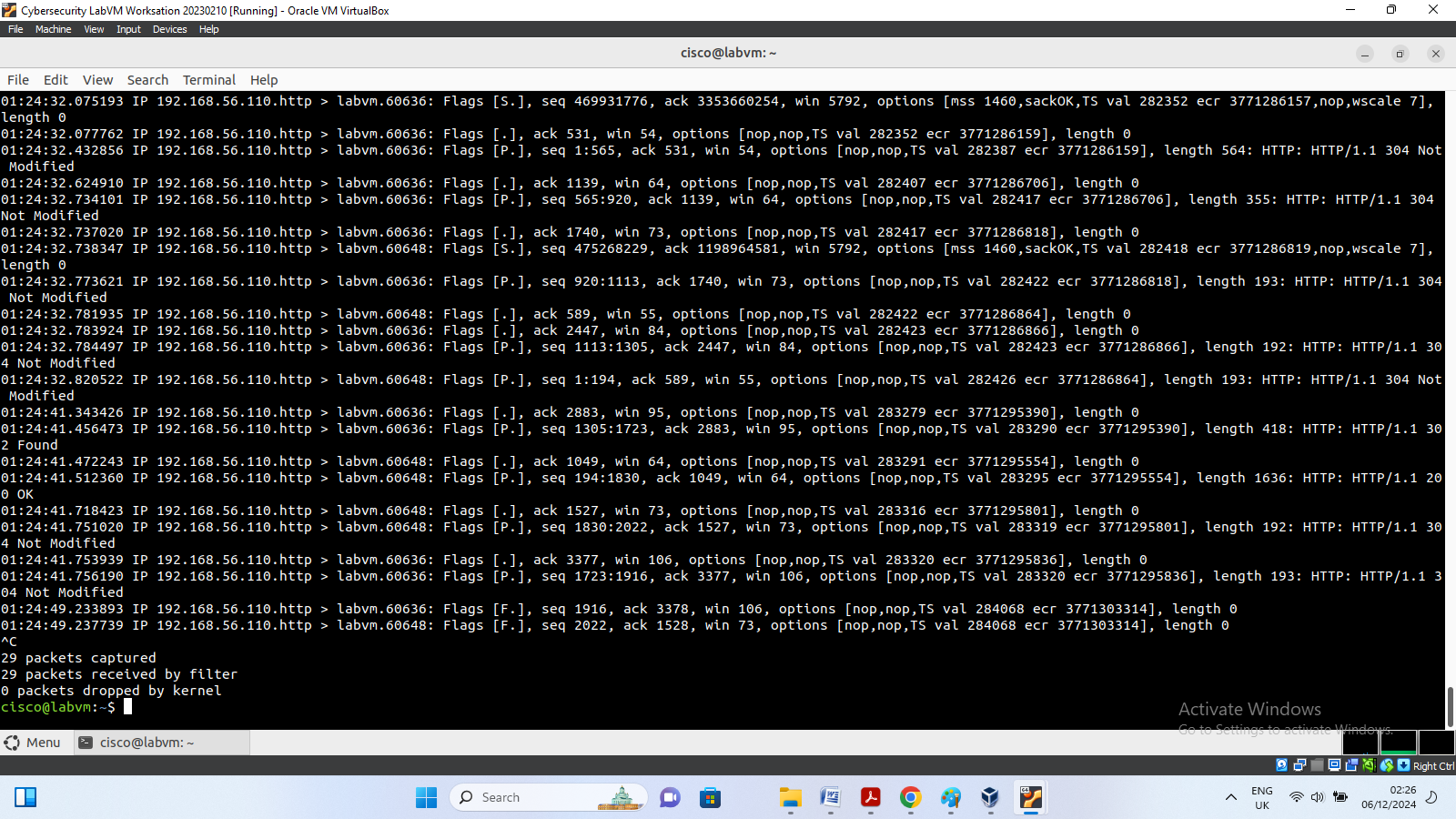






Data from OWASP





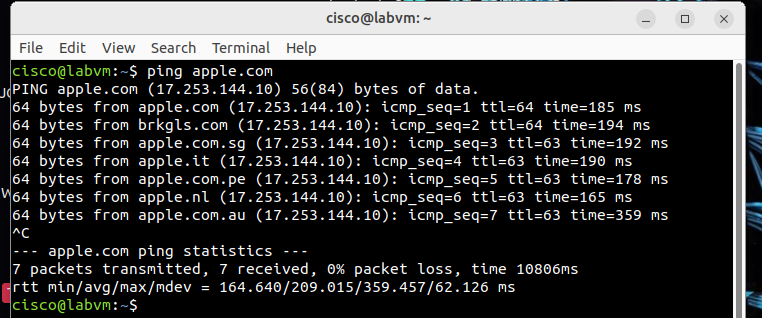
**INT303: Networking Fundamentals – Lab 5**

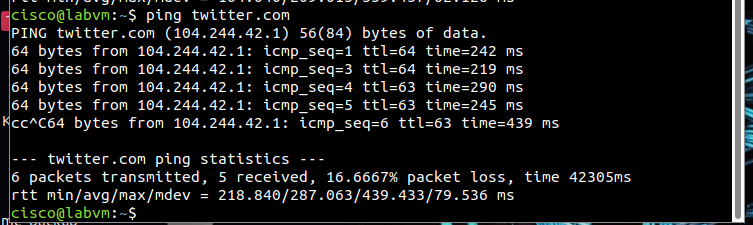
**Please note that professional report required for submission is attached immediately after the lab test screen shots and explanation**

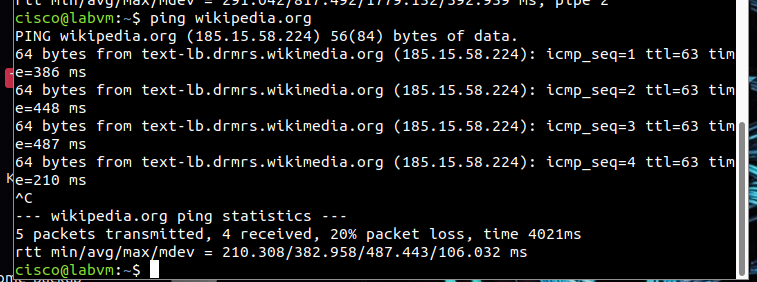
**Step 1: Choose 10 Popular Websites**

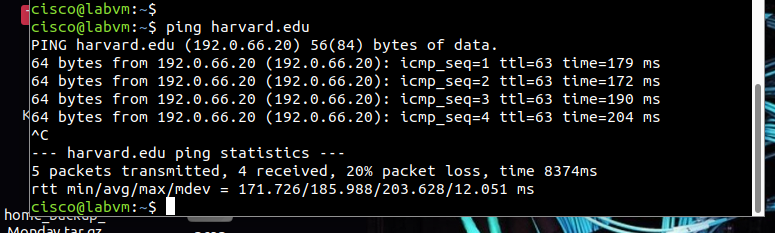
Include a mix of domain types for variety:

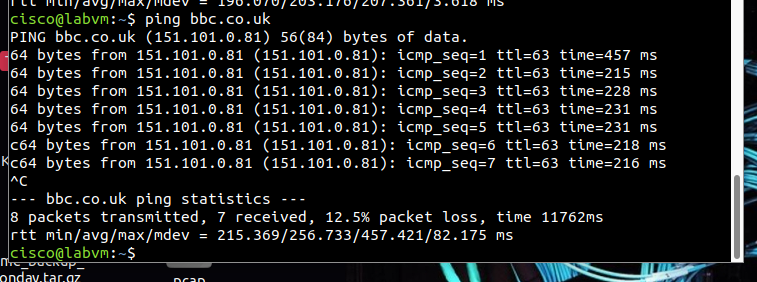
1. **apple.com** (.com - Technology)
2. **twitter.com** (.com - Social Media)
3. **wikipedia.org** (.org - Nonprofit)
4. **harvard.edu** (.edu - Educational Institution)
5. **bbc.co.uk** (.co.uk - Media, UK)
6. **gov.uk** (.gov.uk - Government, UK)
7. **cnn.com** (.com - News Media)
8. **un.org** (.org - United Nations)
9. **spiegel.de** (.de - German News Media)
10. **google.com** (.com - Technology)

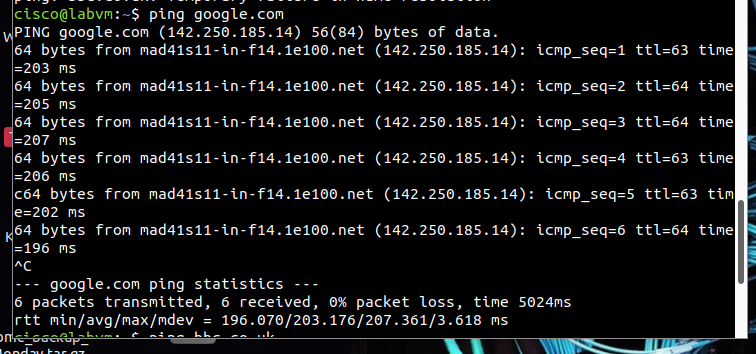


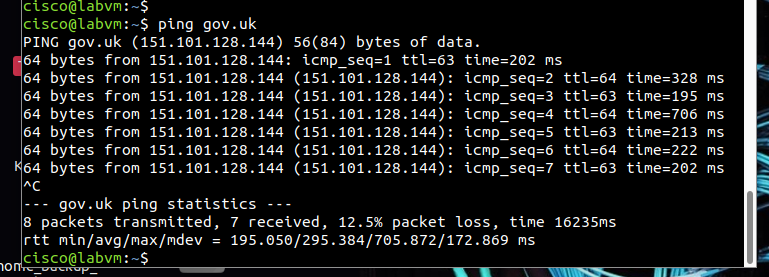


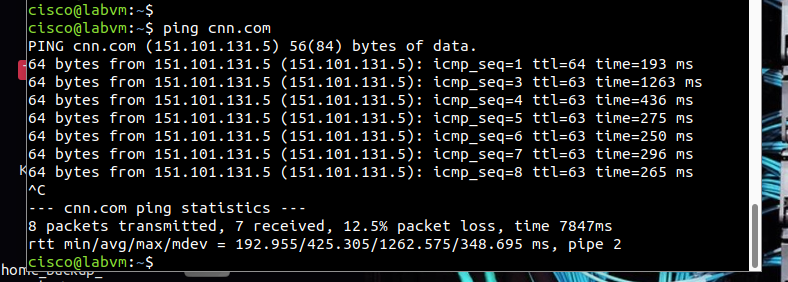


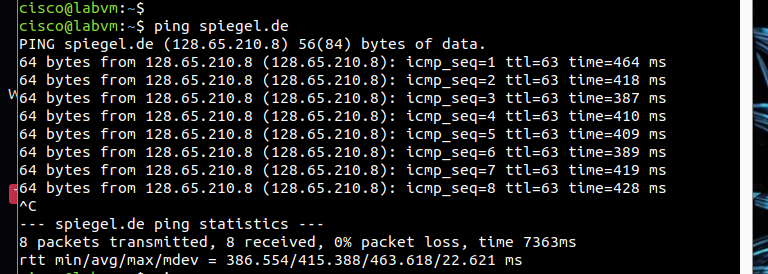


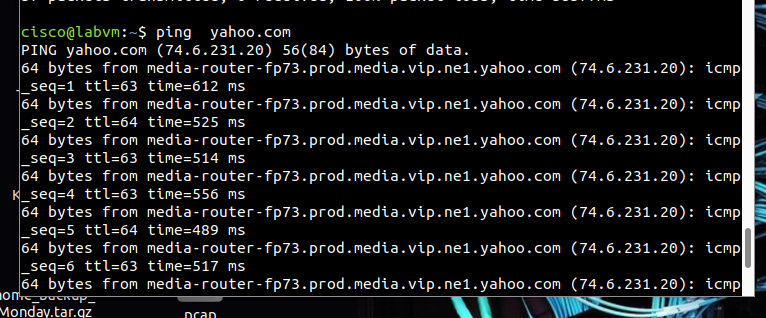












**Step 3: Document the IP Addresses**

3. Create a table to record the IP addresses for each website you ping.

|  |  |  |
| --- | --- | --- |
| Serial number | Website | Ip Address |
| 1 | Apple.com | 17.253.144.10 |
| 2 | Twitter.com | 104.244.42.1 |
| 3 | Wikipedia.org | 185.15.58.224 |
| 4 | Harvard.edu | 192.0.66.20 |
| 5 | Google.com | 142.250.185.14 |
| 6 | Bbc.co.uk | 151.101.0.81 |
| 7 | Gov.uk | 151.101.128.144 |
| 8 | Cnn.com | 151.101.131.5 |
| 9 | Spiegel.de | 128.65.210.8 |
| 10 | Sfczonkwa.com.ng | 108.166.183.137 |

**Step 4: Classify the IP Addresses**

1. Determine the **class** (A, B, or C) of each IP address based on its first octet

Let's classify each IP address based on its first octet:

1. **17.253.144.10**
   * **First Octet:** 17
   * **Class:** A (1-126)
2. **104.244.42.1**
   * **First Octet:** 104
   * **Class:** A (1-126)
3. **185.15.58.224**
   * **First Octet:** 185
   * **Class:** B (128-191)
4. **192.0.66.20**
   * **First Octet:** 192
   * **Class:** C (192-223)
5. **142.250.185.14**
   * **First Octet:** 142
   * **Class:** B (128-191)
6. **151.101.0.81**
   * **First Octet:** 151
   * **Class:** B (128-191)
7. **151.101.128.144**
   * **First Octet:** 151
   * **Class:** B (128-191)
8. **151.101.131.5**
   * **First Octet:** 151
   * **Class:** B (128-191)
9. **128.65.210.8**
   * **First Octet:** 128
   * **Class:** B (128-191)
10. **108.166.183.137**
    * **First Octet:** 108
    * **Class:** A (1-126)



### ****Common Reserved IP Ranges****

Reserved IPs are defined by the Internet Assigned Numbers Authority (IANA) in RFC 1918 and other RFCs for private and special-use purposes. These ranges are not routable on the public Internet and are primarily used in internal networks.

Here are the key ranges:

#### ****Private IP Ranges (RFC 1918)****:

1. **10.0.0.0/8**
   * Range: 10.0.0.0 – 10.255.255.255
   * Total IPs: ~16.7 million
   * Usage: Large private networks (e.g., enterprises or ISPs).
2. **172.16.0.0/12**
   * Range: 172.16.0.0 – 172.31.255.255
   * Total IPs: ~1 million
   * Usage: Medium-sized private networks.
3. **192.168.0.0/16**
   * Range: 192.168.0.0 – 192.168.255.255
   * Total IPs: ~65,536
   * Usage: Small/home networks.

#### ****Other Special Reserved IP Ranges****:

1. **Loopback (127.0.0.0/8)**
   * Range: 127.0.0.1 – 127.255.255.255
   * Usage: Testing and internal host communication (e.g., localhost).
2. **Link-Local (169.254.0.0/16)**
   * Usage: Automatic private IP addressing when no DHCP is available.
3. **Multicast (224.0.0.0/4)**
   * Range: 224.0.0.0 – 239.255.255.255
   * Usage: Multicast traffic (e.g., video streams).
4. **Documentation (192.0.2.0/24, 198.51.100.0/24, 203.0.113.0/24)**
   * Usage: Example addresses for documentation.
5. **Carrier-Grade NAT (100.64.0.0/10)**
   * Range: 100.64.0.0 – 100.127.255.255
   * Usage: NAT in ISP-level networks.
6. **Broadcast Address (255.255.255.255)**
   * Usage: Network-wide broadcasts.

### ****Importance of Reserved IPs in Network Configurations****

1. **Network Segmentation and Scalability**
   * Reserved IPs, especially private ranges, enable organizations to create isolated networks. Without them, IPv4 exhaustion would prevent the scalability of modern networks.
2. **Security**
   * Private IPs prevent external access by default since they are not routable over the public Internet. This makes them ideal for internal systems.
3. **Cost-Effectiveness**
   * Organizations can reuse these IP ranges internally without acquiring public IPs for every device.
4. **IP Conservation**
   * Private IPs help mitigate the exhaustion of IPv4 space. Technologies like NAT allow private networks to share a single public IP for external communication.
5. **Multicast and Routing Functions**
   * Reserved ranges for multicast and loopback ensure proper routing and application behavior. For example, multicast addresses are used for real-time applications like video conferencing.
6. **Testing and Documentation**
   * Reserved ranges like 192.0.2.0/24 help avoid conflicts in testing and training environments.

### ****How to Identify Reserved IPs in Your Network****

To check if any IPs in your network fall into reserved ranges:

1. Identify all subnets in use.
2. Compare them against reserved ranges (e.g., using tools like IP calculators or scripting with Python).
3. Cross-reference against DHCP and static IP configurations.

**Step 5: Calculate the Number of Devices Each IP Can Support**

### Device Capacity for Each IP Class

1. **Class A:**
   * Formula: 224−22^{24} - 2
   * Calculation: 16,777,216−2=16,777,21416,777,216 - 2 = 16,777,214 devices
2. **Class B:**
   * Formula: 216−22^{16} - 2
   * Calculation: 65,536−2=65,53465,536 - 2 = 65,534 devices
3. **Class C:**
   * Formula: 28−22^{8} - 2
   * Calculation: 256−2=254256 - 2 = 254 devices

### Bonus Exercise: Subnetting

#### Class B Network Subnetting

Let's subdivide a Class B network into smaller subnets.

* **Original Class B Network:**
  + Total devices: 65,53465,534
* **Subnetting Example 1: /24 Subnet Mask (Subnetting a Class B into Class C sized subnets)**
  + New Subnet Mask: 255.255.255.0 (/24)
  + Subnets created: 28=2562^{8} = 256 subnets
  + Devices per subnet: 28−2=2542^{8} - 2 = 254 devices
* **Subnetting Example 2: /26 Subnet Mask**
  + New Subnet Mask: 255.255.255.192 (/26)
  + Subnets created: 210=1,0242^{10} = 1,024 subnets
  + Devices per subnet: 26−2=622^{6} - 2 = 62 devices

#### Class C Network Subnetting

Let's consider subnet masks like /26 and /28 for Class C networks.

* **Original Class C Network:**
  + Total devices: 254254
* **Subnetting Example 1: /26 Subnet Mask**
  + New Subnet Mask: 255.255.255.192 (/26)
  + Subnets created: 22=42^{2} = 4 subnets
  + Devices per subnet: 26−2=622^{6} - 2 = 62 devices
* **Subnetting Example 2: /28 Subnet Mask**
  + New Subnet Mask: 255.255.255.240 (/28)
  + Subnets created: 24=162^{4} = 16 subnets
  + Devices per subnet: 24−2=142^{4} - 2 = 14 devices

By performing subnetting, we can break down larger networks into smaller, more manageable subnets, each with its own device capacity. This enhances network organization and scalability.

**Step 6: Determine the Subnet Mask**

### Default Subnet Masks for Each IP Address Class

1. **Class A:**
   * Default Subnet Mask: 255.0.0.0
2. **Class B:**
   * Default Subnet Mask: 255.255.0.0
3. **Class C:**
   * Default Subnet Mask: 255.255.255.0

### Bonus Exercise: Custom Subnetting and Its Impact

#### Scenario: Custom Subnetting for Network Segmentation

Custom subnetting is often required for network segmentation, which helps in improving network performance, security, and management. By altering the subnet mask, we can create smaller subnets within a larger network, each with its own set of IP addresses.

For example, in an organization, different departments (e.g., HR, Sales, IT) might need separate subnets to isolate their traffic and enhance security.

#### Impact on Device Capacity and Network Organization

1. **Subnet Mask /25 (255.255.255.128):**
   * **Class C Network:**
     + Creates 2 subnets.
     + Each subnet has 128 addresses (2^7) minus 2 for network and broadcast addresses.
     + Usable hosts per subnet: 126
2. **Subnet Mask /27 (255.255.255.224):**
   * **Class C Network:**
     + Creates 8 subnets.
     + Each subnet has 32 addresses (2^5) minus 2 for network and broadcast addresses.
     + Usable hosts per subnet: 30

#### Example Calculation

**Original Class C Network (Default Mask /24):**

* Total usable addresses: 254

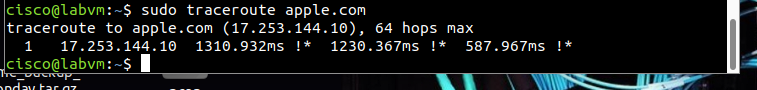
**With Subnet Mask /25:**

* Number of Subnets: 2
* Usable Hosts per Subnet: 126

**With Subnet Mask /27:**

* Number of Subnets: 8
* Usable Hosts per Subnet: 30

**Step 7: Advanced IP Tools (Bonus Exercises)**



1. **First (and Only) Hop (17.253.144.10):**
   * The IP address 17.253.144.10 is directly associated with apple.com.
   * **Response Times:** The response times for this hop are 1310.932ms, 1230.367ms, and 587.967ms. These times indicate how long it took for the packets to travel to and from this address.
   * *Annotations (!*):\* The !\* symbol indicates an administrative filter preventing the traceroute from progressing. This usually means that further probing is blocked or the destination is configured not to respond to further traceroute queries.

**Understanding the Output**

* **Single Hop:** The traceroute shows only one hop, which directly reaches the destination IP address 17.253.144.10.
* **Response Times:** The relatively high response times (especially the first two) suggest either network latency or load on the destination server.
* *Annotations (!*):\* The !\* annotation signifies that this hop is filtered or blocked, commonly by a firewall or other network security mechanism.

This traceroute scan reveals that the packet reaches apple.com (17.253.144.10) in a single hop with notable response times. The !\* annotation suggests that further probing is blocked, likely due to security measures at the destination. This indicates the presence of administrative controls, ensuring that network probing tools like traceroute cannot gather more detailed routing information.

**Report of IP Address Analysis and Subnetting**

**1. Introduction**

The purpose of this lab is to analyze and classify IP addresses, calculate their device capacities, and explore subnetting techniques for improved network organization. IP address analysis and subnetting are critical for managing network scalability, security, and efficiency. This report examines a set of websites, their corresponding IP addresses, and how advanced tools like traceroute and subnetting strategies can optimize real-world network configurations.

**2. Website List and IP Addresses**

| **Serial Number** | **Website** | **IP Address** |
| --- | --- | --- |
| 1 | apple.com | 17.253.144.10 |
| 2 | twitter.com | 104.244.42.1 |
| 3 | wikipedia.org | 185.15.58.224 |
| 4 | harvard.edu | 192.0.66.20 |
| 5 | google.com | 142.250.185.14 |
| 6 | bbc.co.uk | 151.101.0.81 |
| 7 | gov.uk | 151.101.128.144 |
| 8 | cnn.com | 151.101.131.5 |
| 9 | spiegel.de | 128.65.210.8 |
| 10 | sfczonkwa.com.ng | 108.166.183.137 |

**3. IP Address Classification**

The IP addresses were classified based on their first octet, using the IPv4 class system:

| **IP Address** | **First Octet** | **Class** | **Range** |
| --- | --- | --- | --- |
| 17.253.144.10 | 17 | A | 1-126 |
| 104.244.42.1 | 104 | A | 1-126 |
| 185.15.58.224 | 185 | B | 128-191 |
| 192.0.66.20 | 192 | C | 192-223 |
| 142.250.185.14 | 142 | B | 128-191 |
| 151.101.0.81 | 151 | B | 128-191 |
| 151.101.128.144 | 151 | B | 128-191 |
| 151.101.131.5 | 151 | B | 128-191 |
| 128.65.210.8 | 128 | B | 128-191 |
| 108.166.183.137 | 108 | A | 1-126 |

**Characteristics of Classes:**

* **Class A**: Large networks (e.g., enterprises); supports 224−22^{24} - 2224−2 devices.
* **Class B**: Medium-sized networks; supports 216−22^{16} - 2216−2 devices.
* **Class C**: Small networks; supports 28−22^8 - 228−2 devices.

**4. Number of Devices Supported**

| **IP Address** | **Class** | **Device Capacity (Usable Hosts)** |
| --- | --- | --- |
| 17.253.144.10 | A | 16,777,214 |
| 104.244.42.1 | A | 16,777,214 |
| 185.15.58.224 | B | 65,534 |
| 192.0.66.20 | C | 254 |
| 142.250.185.14 | B | 65,534 |
| 151.101.0.81 | B | 65,534 |
| 151.101.128.144 | B | 65,534 |
| 151.101.131.5 | B | 65,534 |
| 128.65.210.8 | B | 65,534 |
| 108.166.183.137 | A | 16,777,214 |

**5. Subnet Masks**

Default subnet masks and possible custom subnetting scenarios:

| **IP Address** | **Class** | **Default Subnet Mask** | **Custom Subnetting** | **Usable Hosts** |
| --- | --- | --- | --- | --- |
| 17.253.144.10 | A | 255.0.0.0 | /24 (Class C size subnet) | 254 |
| 104.244.42.1 | A | 255.0.0.0 | /22 | 1,022 |
| 185.15.58.224 | B | 255.255.0.0 | /26 | 62 |
| 192.0.66.20 | C | 255.255.255.0 | /27 | 30 |

### **6. Advanced Tools and Analysis**

#### **Traceroute Analysis**

* **Target**: apple.com (17.253.144.10)
* **Response Times**:
  + 1310.932ms, 1230.367ms, 587.967ms
* **Annotations (!\*)**: Indicates administrative filters, likely firewalls, blocking further probing.

#### **Significance of Tools**:

* **Traceroute**: Identifies network hops and latency, useful for troubleshooting.
* **GeoIP Analysis**: Maps IP addresses to physical locations, aiding in network security and performance monitoring.

### **7. Conclusion**

This analysis highlights the importance of IP address management and subnetting in modern networking. Classifying IP addresses, calculating device capacities, and utilizing custom subnet masks ensure optimal network performance, scalability, and security. Tools like traceroute and GeoIP analysis further enhance troubleshooting and efficiency in real-world scenario