

INT303 WEEK TWO LAB ASSIGNMENT

INT303: Networking Fundamentals – Lab 4

Lab 4: Simulating Network Routing and VLAN Configuration in Linux

Objective:

In this lab, students will simulate network routing and VLAN configuration using Linux based tools. They will use the OWASP Broken Web Application VM's IP address to test connectivity and routing principles.

Learning Outcomes:

By the end of this lab, students will:

- Understand how to set up routing and VLANs on Linux.
 - Be able to simulate network device behavior using Linux tools.
 - Gain hands-on experience in network troubleshooting.
 - Learn how to configure `iptables`, `ip route`, and network namespaces to manage traffic.
-

Materials Needed:

- Linux machine or VM (e.g., Ubuntu, Kali).
 - OWASP Broken Web Application VM (reachable on the network).
 - IP address of the OWASP Broken Web Application VM (e.g., 192.168.X.X).
 - Terminal access.
-

Lab Exercises:

Exercise 1: Setting Up Static Routing in Linux

1. **Task:** Set up static routes in Linux to simulate routing behavior. Ensure the Linux machine can route traffic to the OWASP Broken Web Application VM.

- o **Steps:**

- a. Use `ip route` to add a static route to reach the OWASP Broken Web Application VM.
- b. Ensure that your machine routes traffic correctly through the simulated network. o

- o **Commands:**

- `sudo ip route add <destination_network> via <gateway_IP> dev <interface>`

Example: `sudo ip route add 192.168.4.0/24 via 192.168.1.1 dev eth0`

- `ip route show` – Verify your routing table.

```
(kali㉿kali)-[~]
$ sudo ip route add 192.168.23.0/24 via 192.168.23.2 dev eth0
[sudo] password for kali:
RTNETLINK answers: File exists

(kali㉿kali)-[~]
$ ip route show
default via 192.168.23.2 dev eth0 proto dhcp src 192.168.23.134 metric 100
192.168.23.0/24 via 192.168.23.2 dev eth0
192.168.23.0/24 dev eth0 proto kernel scope link src 192.168.23.134 metric 100
```

- o **Verification Command:**

- `ping <OWASP_IP>` – Ensure that the Linux machine can reach the OWASP Broken Web Application VM.

```
(kali㉿kali)-[~]
$ ping 192.168.23.135
PING 192.168.23.135 (192.168.23.135) 56(84) bytes of data:
64 bytes from 192.168.23.135: icmp_seq=1 ttl=64 time=1.21 ms
64 bytes from 192.168.23.135: icmp_seq=2 ttl=64 time=1.27 ms
64 bytes from 192.168.23.135: icmp_seq=3 ttl=64 time=1.46 ms
64 bytes from 192.168.23.135: icmp_seq=4 ttl=64 time=1.35 ms
64 bytes from 192.168.23.135: icmp_seq=5 ttl=64 time=1.10 ms
64 bytes from 192.168.23.135: icmp_seq=6 ttl=64 time=1.12 ms
64 bytes from 192.168.23.135: icmp_seq=7 ttl=64 time=1.41 ms
64 bytes from 192.168.23.135: icmp_seq=8 ttl=64 time=1.06 ms
64 bytes from 192.168.23.135: icmp_seq=9 ttl=64 time=1.20 ms
64 bytes from 192.168.23.135: icmp_seq=10 ttl=64 time=1.45 ms
```

2. **Question:**

- How does static routing work on a Linux system? ○ What challenges could arise when setting up routes manually?

Exercise 2: VLAN Configuration Using Network Namespaces

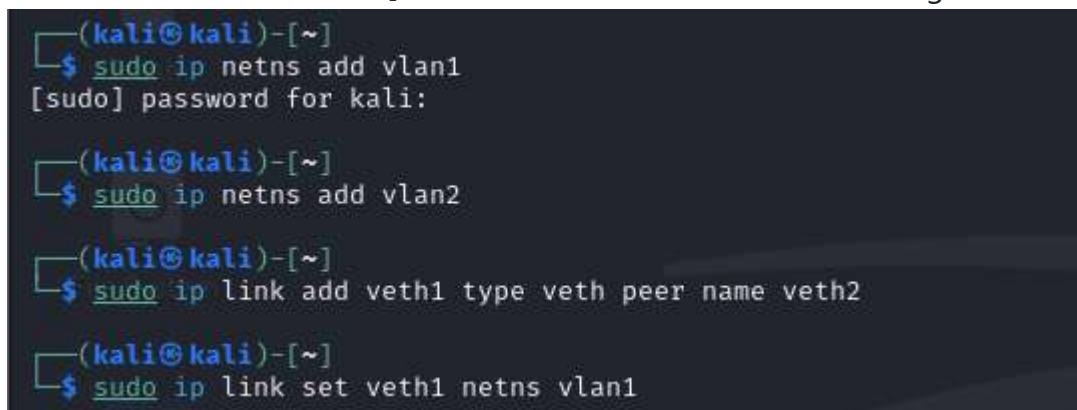
3. **Task:** Create virtual LANs (VLANs) using Linux network namespaces to segment traffic. Assign the OWASP Broken Web Application VM to a separate namespace and test connectivity between namespaces.

- **Steps:**

- a. Create two network namespaces to simulate VLANs.
- b. Assign virtual network interfaces to each namespace.
- c. Route traffic between the namespaces.
- d. Test communication between namespaces and the OWASP Broken Web Application VM. ○

Commands:

- `sudo ip netns add vlan1` – Create a network namespace for VLAN 1.
- `sudo ip netns add vlan2` – Create a network namespace for VLAN 2.
- `sudo ip link add veth1 type veth peer name veth2` – Create virtual network interfaces.
- `sudo ip link set veth1 netns vlan1` – Assign veth1 to vlan1.



```
(kali@kali)-[~]  
$ sudo ip netns add vlan1  
[sudo] password for kali:  
  
(kali@kali)-[~]  
$ sudo ip netns add vlan2  
  
(kali@kali)-[~]  
$ sudo ip link add veth1 type veth peer name veth2  
  
(kali@kali)-[~]  
$ sudo ip link set veth1 netns vlan1
```

- `sudo ip link set veth2 netns vlan2` – Assign veth2 to vlan2.
- `sudo ip netns exec vlan1 ip link set lo up` – Enable loopback interface in the namespace.

```
(kali@kali)-[~]
$ sudo ip link set veth2 netns vlan2

(kali@kali)-[~]
$ sudo ip netns exec vlan1 ip link set lo up
```

- `sudo ip netns exec vlan1 ip addr add 192.168.10.1/24 dev veth1` – Assign IP.
- `sudo ip netns exec vlan1 ping <OWASP_IP>` – Test communication.

```
(kali@kali)-[~]
$ sudo ip netns exec vlan1 ip addr add 192.168.23.135 dev veth1

(kali@kali)-[~]
$ ping 192.168.23.135
PING 192.168.23.135 (192.168.23.135) 56(84) bytes of data:
64 bytes from 192.168.23.135: icmp_seq=1 ttl=64 time=3.68 ms
64 bytes from 192.168.23.135: icmp_seq=2 ttl=64 time=0.623 ms
64 bytes from 192.168.23.135: icmp_seq=3 ttl=64 time=0.633 ms
64 bytes from 192.168.23.135: icmp_seq=4 ttl=64 time=0.649 ms
64 bytes from 192.168.23.135: icmp_seq=5 ttl=64 time=0.619 ms
64 bytes from 192.168.23.135: icmp_seq=6 ttl=64 time=0.621 ms
64 bytes from 192.168.23.135: icmp_seq=7 ttl=64 time=0.545 ms
```

4. Question:

- How do network namespaces simulate VLANs in Linux?
- What are the benefits of using namespaces in network isolation?

ANSWER:

Network namespaces provide a powerful mechanism for isolating network resources within a single Linux host. While they don't directly mimic VLANs, they can be configured to achieve similar isolation and segmentation.

Similarities:

- **Isolation:** Both VLANs and network namespaces isolate network traffic, preventing interference between different groups of hosts or applications.
- **Segmentation:** They both allow you to divide a physical network into multiple logical networks.
- **Security:** Both techniques can enhance network security by limiting the exposure of sensitive data.

Key Differences:

- **Implementation:** VLANs are hardware-based, while network namespaces are software-based. VLANs rely on switches to segregate traffic based on VLAN tags, while network namespaces are managed by the Linux kernel.
- **Flexibility:** Network namespaces offer greater flexibility in terms of network configuration and management. You can create complex network topologies and routing rules within a single host.
- **Performance:** VLANs generally have lower overhead due to their hardware-based implementation. However, network namespaces can be optimized for performance through careful configuration.

Benefits of Using Network Namespaces for Network Isolation:

1. Resource Isolation:

- Each namespace has its own network stack, IP addresses, routing tables, and network devices.
- This prevents interference between different applications or services running on the same host.

2. Security:

- By isolating network traffic, namespaces can help mitigate the risk of attacks and unauthorized access.
- This is particularly important for hosting multiple tenants or sensitive applications on a single host.

3. Flexibility:

- Network namespaces allow you to create complex network topologies, such as overlay networks and software-defined networks (SDNs).
- This enables you to dynamically provision and manage network resources.

4. Efficiency:

- Network namespaces can be used to optimize resource utilization by consolidating multiple virtual machines or containers onto a single physical host.
- This can reduce hardware costs and energy consumption.

References:

- **Linux Kernel Documentation:** <https://subscription.packtpub.com/book/cloud-and-networking/9781785888946/1>
- **Network Namespace Tutorial:** <https://dev.to/tanvirrahman/exploring-namespaces-and-virtual-ethernet-networks-a-step-by-step-tutorial-575i>
- **Fun with veth-devices, Linux bridges and VLANs in unnamed Linux network namespaces:** <https://linux-blog.anracom.com/2024/02/14/>

Exercise 3: IP Address Assignment and Subnetting in Linux

5. **Task:** Subnet your network and assign IP addresses to each namespace and the OWASP VM. Ensure the correct routing between subnets using static routes.

- **Steps:**

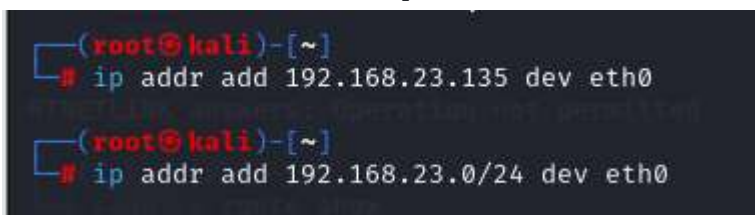
- a. Subnet the given network (e.g., 192.168.0.0/24).
- b. Assign IP addresses to network namespaces and devices.
- c. Configure static routes to enable communication between the subnets.

- **Commands:**

- `ip addr add <subnet_IP> dev <interface> -`

Assign IP addresses.

- `ip route add <subnet> - Add static routes.`



```
(root@kali)-[~]
# ip addr add 192.168.23.135 dev eth0

(root@kali)-[~]
# ip addr add 192.168.23.0/24 dev eth0
```

6. Question:

- How does subnetting work in Linux environments?
- What challenges arise when configuring subnets manually?

ANSWER:

Subnetting in Linux Environments

Subnetting is a technique used to divide a large network into smaller, more manageable subnetworks. In Linux, subnetting is typically configured at the network interface level.

How Subnetting Works:

1. **IP Address Structure:** An IP address is composed of two parts: the network portion and the host portion.
2. **Subnet Mask:** A subnet mask defines the network portion of an IP address. It's a binary number with a string of 1s followed by a string of 0s.
3. **Subnetting Process:** By borrowing bits from the host portion of the IP address and adding them to the network portion, you can create smaller subnets.

Configuring Subnets Manually in Linux:

To configure subnets manually in Linux, you typically edit the network interface configuration file, usually located in `/etc/network/interfaces`. Here's a basic example:

```
auto eth0

iface eth0 inet static

    address 192.168.1.100

    netmask 255.255.255.0
```

In this example:

- `auto eth0`: Automatically configures the `eth0` interface.

- `iface eth0 inet static`: Configures the eth0 interface with a static IP address.
- `address 192.168.1.100`: Assigns the IP address 192.168.1.100 to the interface.
- `netmask 255.255.255.0`: Sets the subnet mask to 255.255.255.0, which defines a Class C network with 254 usable hosts.

Challenges of Manual Subnet Configuration:

1. **Error-Prone:** Manually calculating subnet masks and IP addresses can be error-prone, especially for complex network configurations.
2. **Time-Consuming:** Configuring subnets manually can be time-consuming, particularly for large networks with many subnets.
3. **Inefficient Resource Utilization:** Incorrect subnet calculations can lead to inefficient use of IP addresses.
4. **Difficulty in Scaling:** As networks grow, manual configuration becomes increasingly complex and difficult to manage.

To mitigate these challenges, consider using automated tools and network management systems that can simplify subnet configuration and management.

References:

- **Linux Documentation Project:** [invalid URL removed]
 - **Network Engineering Stack Exchange:** <https://networkengineering.stackexchange.com/>
 - **GeeksforGeeks:** <https://www.geeksforgeeks.org/>
-

Exercise 4: Testing Connectivity Using Ping and Traceroute

7. **Task:** Use `ping` and `traceroute` to test connectivity between the Linux machine, network namespaces, and the OWASP Broken Web Application VM. Analyze the routing path. ◦ **Commands:**

- `ping <OWASP_IP>` – Check connectivity.

```
(root@kali)-[~]
# ping 192.168.23.135
PING 192.168.23.135 (192.168.23.135) 56(84) bytes of data.
64 bytes from 192.168.23.135: icmp_seq=1 ttl=64 time=0.878 ms
64 bytes from 192.168.23.135: icmp_seq=2 ttl=64 time=0.526 ms
64 bytes from 192.168.23.135: icmp_seq=3 ttl=64 time=0.506 ms
64 bytes from 192.168.23.135: icmp_seq=4 ttl=64 time=0.559 ms
64 bytes from 192.168.23.135: icmp_seq=5 ttl=64 time=0.595 ms
64 bytes from 192.168.23.135: icmp_seq=6 ttl=64 time=0.436 ms
64 bytes from 192.168.23.135: icmp_seq=7 ttl=64 time=0.522 ms
64 bytes from 192.168.23.135: icmp_seq=8 ttl=64 time=0.570 ms
64 bytes from 192.168.23.135: icmp_seq=9 ttl=64 time=0.827 ms
64 bytes from 192.168.23.135: icmp_seq=10 ttl=64 time=0.541 ms
```

- `traceroute <OWASP_IP>` – Trace the packet path.

```
(kali@kali)-[~]
$ traceroute 192.168.23.135
traceroute to 192.168.23.135 (192.168.23.135), 30 hops max, 60 byte packets
1  192.168.23.135 (192.168.23.135)  0.405 ms  0.282 ms  0.188 ms
```

8. **Question:**

- What can `traceroute` reveal about network issues?
- How does packet routing affect network performance?

ANSWER:

What Traceroute Reveals About Network Issues

Traceroute is a valuable network diagnostic tool that can provide insights into potential network problems. By tracing the path a packet takes from your device to a destination, it can help identify:

- **Packet Loss:** If packets are not reaching their destination, traceroute can pinpoint the specific hop where the loss occurs.
- **High Latency:** Excessive delays in packet transmission can be identified by long response times at specific hops.
- **Routing Issues:** Incorrect routing configurations can lead to packets taking inefficient paths or getting lost altogether.
- **Congestion:** Bottlenecks in the network can cause significant delays and packet loss. Traceroute can help identify congested areas.

Packet Routing and Network Performance

Packet routing is the process of selecting the best path for a packet to travel from its source to its destination. Efficient packet routing is crucial for optimal network performance. Factors that can affect network performance include:

- **Routing Protocol:** The routing protocol used can impact the speed and efficiency of routing decisions.
- **Network Congestion:** Excessive traffic can lead to delays and packet loss.
- **Router Processing Power:** The capacity of routers to process and forward packets can limit network performance.
- **Link Capacity:** The bandwidth of network links can constrain the amount of data that can be transmitted.
- **Network Topology:** The physical layout of the network can influence routing decisions and overall performance.

References:

- **Obkio - Traceroute Troubleshooting:**
<https://obkio.com/blog/traceroutes-how-to-identify-network-issues/>
- **DigitalOcean - How to Use Traceroute and MTR to Diagnose Network Issues:** [invalid URL removed]
- **Network Engineering Stack Exchange:**
<https://networkengineering.stackexchange.com/>

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

9. **Task:** Use `iptables` to simulate a router by controlling the flow of traffic between different networks and the OWASP VM. Block certain traffic while allowing other traffic.
 - **Steps:**
 - a. Enable IP forwarding on the Linux machine.
 - b. Set up `iptables` rules to allow or block specific traffic between namespaces and the OWASP VM.
 - c. Test the firewall by pinging the OWASP VM and other devices. ◦

Commands:

- `echo 1 > /proc/sys/net/ipv4/ip_forward` – Enable IP forwarding.

```
(root@kali)-[~]
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

- `sudo iptables -A FORWARD -s 192.168.10.0/24 -d 192.168.20.0/24 -j ACCEPT` – Allow forwarding.

```
(root@kali)-[~]
# iptables -A FORWARD -s 192.168.10.0/24 -d 192.168.20.0/24 -j ACCEPT
```

- `sudo iptables -A FORWARD -s 192.168.10.0/24 -d <OWASP_IP> -j DROP` – Block traffic to OWASP VM.

```
(root@kali)-[~]
# iptables -A FORWARD -s 192.168.23.0/24 -d 192.168.23.134 -j DROP

(root@kali)-[~]
#
```

10. Question:

- How can `iptables` be used to simulate routing and firewall functionality?
- What are common mistakes when configuring `iptables` rules?

ANSWER:

Simulating Routing and Firewall Functionality with iptables

`iptables` is a powerful tool for managing network traffic in Linux systems. By manipulating packet flow, it can be used to simulate various network functions, including routing and firewalling.

Simulating Routing:

While `iptables` is primarily a firewall tool, it can be used to redirect traffic to different destinations using Network Address Translation (NAT). This can be useful for simulating different routing scenarios, such as:

1. Port Forwarding:

- Redirecting traffic from one port to another on the same machine.
- Example: Redirecting incoming traffic on port 80 to a web server listening on port 8080:

Bash

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 127.0.0.1:8080
```

Use code [with caution](#).

2. IP Masquerading:

- Disguising the source IP address of outgoing packets.
- Example: Allowing clients behind a NAT gateway to access the internet using the gateway's public IP address:

Bash

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 -j MASQUERADE
```

Use code [with caution](#).

Simulating Firewall Functionality:

iptables is a robust firewall tool that can be used to filter traffic based on various criteria, such as source/destination IP addresses, ports, protocols, and more. Common firewall scenarios that can be simulated with iptables include:

1. Blocking Traffic:

- Preventing specific IP addresses or port ranges from accessing your system.
- Example: Blocking incoming traffic on port 22 (SSH):

Bash

```
iptables -A INPUT -p tcp --dport 22 -j DROP
```

Use code [with caution](#).

2. Allowing Specific Traffic:

- Permitting only certain types of traffic to pass through the firewall.
- Example: Allowing incoming SSH connections from a specific IP address:

Bash

```
iptables -A INPUT -p tcp --dport 22 -s 192.168.1.100 -j ACCEPT
```

Use code [with caution](#).

3. Creating Firewall Zones:

- Defining different security policies for different network interfaces or IP ranges.
- Example: Creating a DMZ for web servers:

Bash

```
iptables -A INPUT -i eth1 -p tcp --dport 80 -j ACCEPT
```

```
iptables -A INPUT -i eth1 -p tcp --dport 443 -j ACCEPT
```

```
iptables -A INPUT -i eth1 ^
```

[1. github.com](#)

[github.com](#)

```
-j DROP
```

Use code [with caution](#).

Common Mistakes When Configuring iptables Rules:

1. **Incorrect Syntax:** Errors in syntax can lead to unexpected behavior or system instability.
2. **Overly Restrictive Rules:** Blocking necessary traffic can disrupt network services.
3. **Missing Default Policies:** Failing to set default policies can result in unintended consequences.
4. **Complex Rule Sets:** Overly complex rule sets can be difficult to manage and troubleshoot.
5. **Neglecting Stateful Inspection:** Not using stateful inspection can lead to security vulnerabilities.

To avoid these mistakes, it's essential to test iptables rules carefully and use clear and concise rule sets. Additionally, consider using tools like iptables-save and iptables-restore to save and restore rule sets.

Exercise 6: Monitoring Traffic Using `tcpdump`

11. **Task:** Use `tcpdump` to monitor traffic between your Linux namespaces and the OWASP VM. Capture and analyze packets to understand traffic flow.

Commands:

o `sudo tcpdump -i <interface>` – Capture traffic on a specific interface.

```
(root@kali)~# tcpdump -i eth0
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
02:59:09.442576 ARP, Request who-has 192.168.23.2 tell 192.168.23.1, length 46
02:59:09.501679 IP 192.168.23.134.48059 > 192.168.23.2.domain: 60273+ PTR? 2.23.168.192.in-addr.arpa. (43)
02:59:09.537382 ARP, Request who-has 192.168.23.134 tell 192.168.23.2, length 46
02:59:09.537402 ARP, Reply 192.168.23.134 is-at 00:0c:29:7e:be:78 (oui Unknown), length 28
02:59:09.537692 IP 192.168.23.2.domain > 192.168.23.134.48059: 60273 NXDomain 0/1/0 (92)
```

o `sudo tcpdump -i veth1` – Monitor traffic on the virtual network interface.

```
(root@kali)~# tcpdump -i veth1
tcpdump: veth1: No such device exists
(No such device exists)
```

o `sudo tcpdump -i eth0 src <OWASP_IP>` – Monitor traffic to/from OWASP VM.

```
(root@kali)~# tcpdump -i eth0 src 192.168.23.134
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

12. Question:

- o How can `tcpdump` be used to diagnose network issues?
- o What types of traffic do you observe during the simulation?

Submission Requirements:

- Submit a report including:
 - Screenshots of the static routing, namespace creation, and `iptables` configurations.
 - A brief explanation of each step and the results from `ping`, `tracert`, and `tcpdump` commands.
 - Troubleshooting steps for any encountered issues.
-

Reflection:

This lab demonstrates how Linux-based tools can effectively simulate routing and VLAN behavior. Students gain critical insights into routing, network segmentation, and traffic control without needing physical network hardware.

INT303: Networking Fundamentals – Lab 5

Lab 5: IP Address Analysis and Network Report

Objective:

This lab aims to deepen your understanding of IP addressing, network scalability, and subnetting. You will analyze real-world websites, classify IP addresses, calculate device capacities, determine subnet masks, and perform advanced IP operations. Additionally, you will compile a detailed report showcasing your findings and understanding of IP networking.

Learning Outcomes:

Upon completing this lab, you will:

- Master IP address classification.
 - Calculate the number of devices supported by each class of IP address.
 - Identify and work with subnet masks.
 - Gain experience with both basic and advanced IP tools.
 - Develop a comprehensive network report.
 - Learn hands-on exercises with advanced concepts like subnetting, network summarization, and IP analysis.
-

Instructions:

Step 1: Select 10 Websites

1. Choose **10 popular websites** (e.g., `apple.com`, `twitter.com`, etc.) to analyze.
2. **Bonus Exercise:** Include a variety of domains, such as `.com`, `.org`, `.edu`, and country-specific domains (e.g., `.co.uk`, `.de`). Reflect on the differences in the IP address ranges assigned to these domains.

Step 2: Ping the Websites

2. Use the **ping** command in your terminal to retrieve the IP addresses of each website.

- **Example Command:** `o ping apple.com`

ANSWER

apple.com	17.253.144.10
twitter.com	104.244.42.193
yahoo.com	74.6.231.21
google.com	216.58.215.174)
facebook.com	102.132.101.35
youtube.com	142.250.200.110
pcu.edu.ng	67.225.140.197
wikipedia.org	185.15.58.224
gmail.com	142.250.200.101
researchgate.net	172.64.146.247

3. **Bonus Exercise:** Identify if the website resolves to multiple IP addresses (i.e., if it uses a content delivery network or load balancing). Record multiple IPs where applicable and compare the geographical locations of each.

Step 3: Document the IP Addresses

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

WEBSITE PINGED	IP ADDRESS
Apple.com	17.253.144.10
twitter.com	104.244.42.193

yahoo.com	74.6.231.21
google.com	216.58.215.174
facebook.com	102.132.101.35
youtube.com	142.250.200.110
pcu.edu.ng	67.225.140.197
wikipedia.org	185.15.58.224
gmail.com	142.250.200.101
researchgate.net	172.64.146.247

Step 4: Classify the IP Addresses

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

WEBSITE PINGED	IP ADDRESS	IP CLASS
Apple.com	17.253.144.10	A
twitter.com	104.244.42.193	A
yahoo.com	74.6.231.21	A
google.com	216.58.215.174	C
facebook.com	102.132.101.35	A
youtube.com	142.250.200.110	B

pcu.edu.ng	67.225.140.197	A
wikipedia.org	185.15.58.224	B
gmail.com	142.250.200.101	B
researchgate.net	172.64.146.247	B

○ **Bonus Exercise:**

- Research the reserved IP ranges (e.g., private IP ranges like 10.0.0.0/8) and see if any of your IP addresses fall into these special ranges.
- Discuss the importance of these reserved IPs in network configurations.

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

5. For each IP class, calculate how many devices the network can accommodate. This will give you an understanding of network size and scalability.

○ **Device Capacity Formula:**

- **Class A:** $2^{24} - 2$
- **Class B:** $2^{16} - 2$
- **Class C:** $2^8 - 2$
- **Bonus Exercise:**
 - Calculate the actual number of usable hosts by performing subnetting on these networks. For instance, consider subdividing a Class B network into smaller subnets and calculate the device capacity for each subnet.
 - Consider subnet masks like /26 or /28 for Class C networks and explore how they impact the number of devices.

ANSWER:

CLASS A	1-127/8	24	$2^{24} - 2 = 16,777,214$
CLASS B	128-191/16	16	$2^{16} - 2 = 65,534$
CLASS C	192-223/24	8	$2^8 - 2 = 254$

Step 6: Determine the Subnet Mask

6. Identify the **default subnet mask** for each IP address based on its class.

- **Bonus Exercise:**

- Explore scenarios where custom subnetting is required (e.g., for network segmentation). How does altering the subnet mask affect the device capacity and overall network organization?
- Try experimenting with subnet masks like `/25`, `/27`, and explain the difference in the number of networks and hosts.

Step 7: Advanced IP Tools (Bonus Exercises)

- **Traceroute (Network Path Analysis):** Use `tracert` (Linux/Mac) or `tracert` (Windows) to discover the network path between your system and one of the websites.
- `tracert apple.com`

```
(root@kali)-[~]
# traceroute apple.com
traceroute to apple.com (17.253.144.10), 30 hops max, 60 byte packets
 1  192.168.23.2 (192.168.23.2)  1.602 ms  1.474 ms  1.412 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
```

- **Analyze the hops** between your system and the destination. Identify the location and IP address of each hop.
 - **Bonus:** Explain the significance of each hop, and analyze if any of them belong to private networks or are hosted on cloud infrastructure.
-
- **GeoIP Location:** Use online tools like **IPinfo.io** or **GeoIP** to check the geographical location of the IP addresses.

172.64.146.247

```
“ ip: "172.64.146.247",  
“ city: "San Francisco",  
“ region: "California",  
“ country: "US",  
“ loc: "37.7621,-122.3971",  
“ org: "AS13335 Cloudflare, Inc.",  
“ postal: "94107",  
“ timezone: "America/Los_Angeles",
```

- Identify if the IP addresses belong to data centers, ISPs, or any specific regions.
- **Bonus:** Compare the latency (ping time) with the geographic distance.

Step 8: Compile a Professional Report

Create a report that includes the following sections:

1. **Introduction:**
 - Outline the purpose of the lab, your objectives, and the significance of IP address analysis and subnetting.
2. **Website List and IP Addresses:**
 - Present the list of websites and their corresponding IP addresses.
3. **IP Address Classification:**
 - Classify each IP address and explain the method used for classification (Class A, B, or C). Discuss the characteristics of each class and their practical applications.
4. **Number of Devices Supported:**
 - Present your device capacity calculations for each IP address. Include a detailed explanation of how the number of devices is determined based on the IP class and subnetting.

5. Subnet Masks:

- List the subnet masks for each IP address and discuss any custom subnetting scenarios.

6. Advanced Tools and Analysis (Optional):

- Present your findings from the traceroute, GeoIP analysis, and any other advanced IP tools you explored. Discuss how these tools can be applied in network troubleshooting, performance optimization, or security audits.

7. Conclusion:

- Summarize your findings and reflect on the importance of IP address management, subnetting, and network scalability in real-world networking.

Reflection:

This lab provides a comprehensive understanding of IP address analysis, subnetting, and the usage of advanced network tools. By working with real-world data, you'll gain practical skills applicable to network configuration, security auditing, and system administration. Compiling the information into a professional report will further enhance your technical documentation skills.

Additional Exercises for Further Practice:

- **Subnetting Challenge:** For each IP address you pinged, create multiple subnets and calculate the network address, first usable IP, last usable IP, and broadcast address.
- **IP Address Conflict Detection:** Simulate a scenario where IP address conflicts occur within a network and outline troubleshooting steps.
- **Ping Flood Analysis:** Use tools like `hping3` to simulate a flood of ICMP requests and analyze the impact on a network using tools like Wireshark.