1. Network Scanning using Nmap

Objective:
To perform active network scanning using the Nmap tool and analyze the discovered host and service information.

Instructions:

Target Setup:
Use theIP address or hostname for scanning

Tasks:
1.**Basic Ping Scan** nmap -sn google.com
Run a ping scan to check if the host is up.
2 . **Port Scanning** nmap google.com
Scan the target system for open ports.
3 . **Service and Version Detection** nmap -sV google.com
Identify services running on open ports and detect their versions.
4 . **Operating System Detection** nmap -O google.com
Detect the operating system of the target machine.
5 . **Aggressive Scan** nmap -A scanme.nmap.org
Perform an aggressive scan to gather detailed information.

remember not to scan any public websites like google.com or any other big website like this.

Deliverables:
-A screenshot of the output for each scan.
-A brief explanation (2–3 lines) of what each scan reveals.
-Mention any security risks observed based on open ports/services.

2 : Capturing and Analyzing HTTP Traffic using Wire Shark

Objective:
To capture HTTP packets using Wireshark and analyze basic web communication between a client and server.

Tasks:

1. Open Wireshark and start capturing traffic on your active network interface.
2. Open a web browser and visit `http://example.com` or any other non-HTTPS website.
3. Stop the capture after the page loads
4. Apply a display filter to show only HTTP packets:

5. Analyze and answer the following:
   -What is the IP address of the web server?
   -Identify a **GET** request in the capture. What resource is being requested?
   -What is the **User-Agent** string used by the browser?

Deliverables:

Screenshot of HTTP request and response packets.
Answers to the questions above.
Brief summary of how HTTP traffic is structured.

3 : DNS Traffic Analysis Using Wireshark

Objective:
To capture DNS queries and responses, and understand how domain name resolution works.

Tasks:

1.Start capturing on Wireshark.
2.Open a browser and visit a few different websites (e.g., `www.google.com`, `www.wikipedia.org`).
3.Stop the capture.

4.Apply a display filter for DNS:
5.Analyze and answer:
   -What IP addresses are returned for the visited domains?
   -Identify a DNS query and its corresponding response.
   -Was there any failed DNS resolution?

Deliverables:

Screenshot showing a DNS query and response.
Table listing at least 3 domains with their resolved IPs.
A note on what happens if DNS resolution fails.

4: Analyzing TCP 3-Way Handshake

Objective:
To capture and analyze the TCP 3-way handshake process using Wireshark.

Tasks:

1. Start capturing traffic on Wireshark.

2. Open a browser and visit any website.

3. Stop capturing after the page loads.
4. Apply a display filter for TCP handshakes:
5. Locate a full TCP handshake (SYN, SYN-ACK, ACK) in the packet list.
6. Analyze and answer

   -What are the source and destination IP addresses and ports?
   -How is the sequence number set in each part of the handshake?
   -Is the handshake completed successfully?

Deliverables:

Screenshot of the TCP handshake packets.
A labeled diagram showing SYN, SYN-ACK, and ACK.
Short explanation of the TCP connection establishment process.

5 : Decrypting Vigenère Cipher (Basic)

Objective:
To implement a C++ program that decrypts text encrypted using the Vigenère cipher.

Tasks:

1. Write a C++ program that:
   -Accepts a ciphertext and a keyword.
   -Implements the Vigenère decryption algorithm.
   -Outputs the decrypted plaintext.

2. **Input:**
   Ciphertext: `LXFOPVEFRNHR`
   Keyword: `LEMON`

Expected Output:
Your program should print the original plaintext.

Deliverables:

C++ source code (with comments).
Decrypted plaintext output.
Screenshot of successful program execution.

6: Vigenère Cipher Implementation in C++

Objective:
To implement the Vigenère Cipher encryption algorithm in C++ and generate the ciphertext for the given input.

Instructions:

1. Write a C++ program to:
   -Accept a **plaintext** and a **keyword**.
   -Encrypt the plaintext using the Vigenère Cipher technique.
   -Display the resulting **ciphertext**.

2. Use the following inputs:
   Plaintext: `HELLOVIGENERE`
   Keyword: `KEY`

Expected Output:

Your program should print the **ciphertext** after encryption.

Deliverables:

C++ source code (with comments).
Decrypted plaintext output.
Screenshot of successful program execution.

7: Securing Network Devices Using Cisco Packet Tracer

Objective:
To configure basic security features on Cisco routers and switches to protect network devices from unauthorized access.

Network Topology:

Design a simple topology in **Cisco Packet Tracer** with:

1 Router (`R1`)
1 Switch (`S1`)
1 PC (`PC1`) connected to the switch

Tasks:

1. Set Console and VTY Passwords on the Router

Set a **console password** as `cisco123`
Set a **VTY password** as `telnet123`
Configure **login** to require a password for both console and VTY access.

2. Enable a Password for Privileged EXEC Mode

Set an **enable secret password**: `secureadmin`

3. Configure Banner Message of the Day (MOTD)

Set the banner as:
`"Unauthorized access is prohibited!"`

4. Secure Access to the Switch

Repeat steps 1 to 3 on the switch `S1` with:
- Console password: `switch123`
-VTY password: `remotesw`

-Enable secret: `adminsw`

Deliverables:

Packet Tracer (.pkt) file with the configured topology.
Screenshots of:

-Console login prompt.
-VTY login prompt via Telnet.
-`show running-config` output showing encrypted passwords.
-A short explanation of why each security step is important.

8: Basic Network Administration and Troubleshooting using Windows Command Line Utilities

Objective:
To use basic Windows command-line tools to analyze, monitor, and troubleshoot network connectivity and configuration issues.

Tasks:

1. View Network Configuration : Note down and report the following:

IP Address ipconfig /all
Subnet Mask
Default Gateway
MAC Address (Physical Address)

2. Test Connectivity ping <your_gateway_IP>

Ping your default gateway
Ping a public DNS server ping google.com
Ping a website:

Record the response times and identify if there's packet loss.

3. Trace the Route to a Remote Host tracert google.com

Observe and write down how many hops it takes to reach the destination.

4. Check DNS Resolution nslookup google.com

Record the IP address(es) returned.

5. View Active Connections and Listening Ports Using `netstat` netstat -an

Answer the following:
How many **TCP connections** are in the **ESTABLISHED** state?
netstat -an | find /c "ESTABLISHED"

Are there any services listening on port **80** or **443**?

netstat -an | find ":80"
netstat -an | find ":443"

Deliverables:

Screenshots of each command output.
Answers to the questions in each task.
A short conclusion on what the tests revealed about your system's network connectivity.

9 : Network Traffic Capture and Analysis Using TCPDump

Objective:
To use the **TCPDump** tool to capture and analyze network traffic for understanding packet-level communication on a network.

Tasks:
1.Capture All Network Traffic (Basic Capture) sudo tcpdump -i eth0 -c 100

Run **TCPDump** to capture all traffic on your active network interface.
Let the capture run for 10 seconds, then stop it
Observe and record the types of packets that appear (e.g., ARP, IP, TCP, UDP).

tcpdump -r http_traffic.pcap -A

2.Capture Only HTTP Traffic sudo tcpdump -i etho tcp port 80 -w http_traffic.pcap
from this all the captured packets will be stored in http_traffic.pcap

Capture HTTP traffic to and from a web server by filtering traffic on port 80.
Visit any website (e.g., `http://example.com`) while the capture is running.
top the capture and observe the HTTP packets (GET, POST, etc.) and their contents.
so then run this command wireshark http_traffic.pacp which will open this file and will help you analysis the packets. in filter put http to get desired result.

3. Capture and Analyze TCP Handshake

Run **TCPDump** to capture only **TCP traffic**. sudo tcpdump -i eth0 tcp -w tcp_handshake.pacap
Initiate a TCP connection by opening a website (e.g., `https://www.google.com`).
Stop the capture and analyze the three-way TCP handshake:
SYN: Sent by the client. wireshark tcp_handshake.pcap
SYN-ACK: Sent by the server.
tcpdump -r tcp_handshake.pcap
ACK: Sent by the client to confirm the connection.

Deliverables:

1. **Screenshots** of:
   TCPDump output for each of the tasks.
   Analysis of key packets (TCP handshake, HTTP traffic, DNS queries).
   The contents of a specific packet in the capture (e.g., HTTP GET request, TCP SYN packet).

2. **Analysis Report:**
   Answers to the following questions:
   What type of traffic was most common in your capture?
   What were the source and destination IP addresses in the TCP handshake?
   What domain name was resolved through DNS queries?