**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**

Run a ping scan to check if the host is up.

2. **Port Scanning**

Scan the target system for open ports.

3.**Service and Version Detection**

Identify services running on open ports and detect their versions.

4.**Operating System Detection**

Detect the operating system of the target machine.

5. **Aggressive Scan**

Perform an aggressive scan to gather detailed information.

**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.

**SOLUTION:**

📚 **What You Need To Do:**

* You will use **Nmap** (a network scanning tool) to perform **different types of scans** on a **target machine** (IP address or hostname).
* For **each type of scan**, you must:
  + Take a **screenshot** of the result.
  + Write a **brief 2–3 line explanation** of what the result shows.
  + Mention **any security risks** you observe (like dangerous open ports).

🛠️ **Step-by-Step Guide:**

**1. Basic Ping Scan (Check if Host is Alive)**

🔹 **Purpose**: See if the target device is up (alive) or not.

🔹 **Command**:

bash

CopyEdit

nmap -sn <target-ip>

🔹 **Explanation**:

* -sn stands for **"ping scan"** (previously called "No port scan").
* It only checks if the host responds to ping (ICMP) or ARP (on local network).

🔹 **Example**:

bash

CopyEdit

nmap -sn 192.168.1.10

🔹 **Source**: Nmap official docs - Host Discovery

**2. Port Scanning (Find Open Ports)**

🔹 **Purpose**: Find which ports are open on the target (like port 22 for SSH, port 80 for HTTP, etc.).

🔹 **Command**:

bash

CopyEdit

nmap <target-ip>

🔹 **Explanation**:

* A simple scan that checks the **most common 1,000 TCP ports**.
* It tells you which ports are open and **listening**.

🔹 **Example**:

bash

CopyEdit

nmap 192.168.1.10

🔹 **Source**: Nmap official docs - Port Scanning Basics

**3. Service and Version Detection (What is Running on Open Ports)**

🔹 **Purpose**: Identify which service (e.g., Apache, SSH) is running and its **version**.

🔹 **Command**:

bash

CopyEdit

nmap -sV <target-ip>

🔹 **Explanation**:

* -sV asks Nmap to try to determine **service type** (like HTTP server) and **version number** (like Apache 2.4.7).
* Helps in finding vulnerabilities based on software versions.

🔹 **Example**:

bash

CopyEdit

nmap -sV 192.168.1.10

🔹 **Source**: Nmap official docs - Service Version Detection

**4. Operating System Detection (Find the OS)**

🔹 **Purpose**: Identify the **operating system** (Linux, Windows, etc.) running on the target.

🔹 **Command**:

bash

CopyEdit

nmap -O <target-ip>

🔹 **Explanation**:

* -O triggers **OS fingerprinting**.
* Based on how the host responds to TCP/IP stack behavior, Nmap guesses the OS.

🔹 **Example**:

bash

CopyEdit

nmap -O 192.168.1.10

🔹 **Source**: Nmap official docs - OS Detection

**5. Aggressive Scan (Detailed Information Gathering)**

🔹 **Purpose**: Gather a lot of info at once, including open ports, services, versions, OS, traceroute, etc.

🔹 **Command**:

bash

CopyEdit

nmap -A <target-ip>

🔹 **Explanation**:

* -A does **OS detection**, **version detection**, **script scanning**, and **traceroute**.
* Gives a **very detailed** scan result.

🔹 **Example**:

bash

CopyEdit

nmap -A 192.168.1.10

🔹 **Source**: Nmap official docs - Aggressive Scan (-A)

✍️ **What You Must Submit:**

1. **Screenshot** of output for each scan (5 screenshots total).
2. **2–3 line explanation** of what the output reveals.
3. **List of security risks** based on:
   * Open ports (e.g., Port 23 Telnet is insecure),
   * Outdated services (e.g., Apache 2.2 vulnerable),
   * OS vulnerabilities.

📋 **Quick Summary Table**

|  |  |  |
| --- | --- | --- |
| **Step** | **Command** | **Purpose** |
| 1 | nmap -sn <target-ip> | Check if the host is up |
| 2 | nmap <target-ip> | Scan for open ports |
| 3 | nmap -sV <target-ip> | Detect running services and versions |
| 4 | nmap -O <target-ip> | Detect operating system |
| 5 | nmap -A <target-ip> | Perform aggressive scan for full details |

**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.

**SOLUTION:**

📜 **Problem Statement Overview:**

You are tasked to **capture** HTTP network traffic between your computer and a web server using **Wireshark**, and **analyze**it to answer basic questions about the communication (like the server’s IP, GET request, User-Agent etc.).

🛠 **Full Step-by-Step Process:**

**Step 1: Open Wireshark and Start Capture**

🔹 **What is Wireshark?**  
Wireshark is a **packet analyzer** — it records all the network traffic going through your network interface (like WiFi or Ethernet).

🔹 **How to do it?**

1. **Open Wireshark** application on your PC.
2. You will see a list of **network interfaces** (like Wi-Fi, Ethernet, etc.).
3. Find your **active** network (where your internet is connected — usually Wi-Fi).
4. **Double click** on it to **start capturing**.

📖 Source:

* Wireshark User Guide - Starting Captures

**Step 2: Visit a Non-HTTPS Website**

🔹 **Why?**  
HTTP traffic is **unencrypted** so Wireshark can easily capture and show readable information (unlike HTTPS which is encrypted).

🔹 **How to do it?**

1. Open any web browser (Chrome, Firefox, Edge, etc.).
2. Type:  
   arduino  
   CopyEdit  
     
     
   http://example.com
3. (Make sure it’s HTTP, not HTTPS).

📌 **Important:** If example.com redirects automatically to HTTPS, you can use another non-HTTPS test site like:

* http://neverssl.com

📖 Source:

* What is HTTP and HTTPS? (Cloudflare)

**Step 3: Stop Capture**

🔹 **How to do it?**

1. After the page loads fully, go back to Wireshark.
2. Click the **red square** (🟥) button to **stop** the capture.

**Step 4: Apply Display Filter for HTTP Packets**

🔹 **Why?**  
You want to **see only HTTP packets**, not all other network noise (like DNS, TCP handshake, etc.).

🔹 **How to do it?**

1. In the **Display Filter** bar (top of Wireshark window), type:  
   nginx  
   CopyEdit  
     
     
   http
3. Press **Enter**.

Now Wireshark will **show only HTTP packets**.

📖 Source:

* Wireshark Display Filters Guide

**Step 5: Analyze and Answer the Questions**

Now based on the HTTP packets you captured, answer the following:

🔎 **Question 1: What is the IP address of the Web Server?**

**How to find it:**

* Look at a **packet with "GET"** or "**HTTP/1.1 200 OK**" in Info column.
* Check the **Destination IP** address — this is the **server’s IP**.

🔎 **Question 2: Identify a GET Request. What Resource is Being Requested?**

**How to find it:**

* Find a packet with **Method: GET**.
* Expand the packet details in the middle pane (click the arrow next to HTTP).
* Look for **Request URI** — it tells what resource was requested (like /index.html, /, etc.)

🔎 **Question 3: What is the User-Agent String Used by the Browser?**

**How to find it:**

* In the same GET request packet, under HTTP, expand **Request Header**.
* You will see a field **User-Agent**.
* It shows which browser (Chrome, Firefox, etc.) and system (Windows, Mac, etc.) you used.

📖 Source:

* [User-Agent Header Explanation (MDN)](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent)

📷 **Deliverables You Need to Prepare:**

✅ Take **screenshots**:

* One screenshot showing an HTTP **request packet**.
* One screenshot showing an HTTP **response packet**.

✅ Prepare **answers**:

* Web server IP address.
* What resource was requested.
* What was the User-Agent string.

✅ Write a **brief summary**:

* Explaining how HTTP traffic is structured (I’ll give a simple version below 👇).

📚 **Brief Summary: How HTTP Traffic is Structured**

* HTTP (HyperText Transfer Protocol) is a **request-response protocol**.
* **Client (your browser)** sends an **HTTP Request** (like GET, POST).
* **Server** replies with an **HTTP Response** (status code, web page data, etc.).
* HTTP traffic consists of:
  + **Request Line** (e.g., GET / HTTP/1.1)
  + **Headers** (browser details, accepted formats)
  + **Body** (optional, like form data)
  + **Response Line** (e.g., HTTP/1.1 200 OK)
  + **Response Headers** (server type, content type)
  + **Response Body** (actual HTML page content)

📖 Source:

* [MDN Web Docs - HTTP Overview](https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview)

✅ **Summary of Steps to Follow:**

|  |  |
| --- | --- |
| **Step** | **Action** |
| 1 | Open Wireshark, start capture on active network |
| 2 | Open browser, visit http://example.com |
| 3 | Stop capture after page loads |
| 4 | Apply filter: http |
| 5 | Find IP, GET request resource, and User-Agent |
| 6 | Take screenshots and write brief summary and answers |

**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

**SOLUTION:**

🔵 **Objective**

You are tasked to **capture DNS queries** (when you ask for a website’s IP) and **DNS responses** (the IP address the server sends back), to understand how the Domain Name System works.

(Source for understanding DNS: Cloudflare DNS Overview)

🔵 **Steps to Perform the Lab**

**Step 1: Start Capturing on Wireshark**

🔹 **What you are doing:**  
Starting Wireshark capture to monitor all incoming and outgoing network traffic.

🔹 **How to do it:**

1. Open **Wireshark** (ensure you have it installed).
2. Select the correct **network interface** (like Wi-Fi or Ethernet, depending on what you're using).
   * You will usually see some interfaces with activity (small moving graphs).
3. Click the **"Start Capturing Packets"** button (the blue shark fin icon).

🔹 **Important Tip:**  
Make sure you start capture **before** visiting any website.

📚 Reference: Wireshark Quick Start Guide

**Step 2: Open a Browser and Visit a Few Websites**

🔹 **What you are doing:**  
Generating DNS queries by accessing websites.

🔹 **How to do it:**

1. Open **Google Chrome**, **Firefox**, or any browser.
2. Visit websites like:
   * [www.google.com](https://www.google.com/)
   * [www.wikipedia.org](https://www.wikipedia.org/)
   * [www.github.com](https://www.github.com/)
3. Wait for each website to **fully load**.

🔹 **Important Tip:**  
Don't visit too many sites (3–5 are enough) or your capture file will become too big.

📚 Reference: How Web Browsers Use DNS

**Step 3: Stop the Capture**

🔹 **What you are doing:**  
Stopping Wireshark once you’ve created enough traffic.

🔹 **How to do it:**

* Go back to **Wireshark**.
* Click the **red square "Stop Capturing Packets"** button.

🔹 **Important Tip:**  
Stopping prevents unnecessary data from getting captured.

📚 Reference: Stopping Capture in Wireshark

**Step 4: Apply a Display Filter for DNS**

🔹 **What you are doing:**  
Filtering out only **DNS packets** (ignoring all other packets like TCP, HTTP, etc.).

🔹 **How to do it:**

* In the top **Display Filter** bar of Wireshark, type:  
  nginx  
  CopyEdit  
    
    
  dns
* Then press **Enter**.

🔹 **What happens:**  
Wireshark now shows **only DNS traffic** — both **queries** and **responses**.

📚 Reference: Wireshark DNS Filtering Guide

**Step 5: Analyze and Answer Questions**

**a) What IP addresses are returned for the visited domains?**

🔹 **How to do it:**

* Look for **Standard query responses** in the Info column.
* Expand the packet details → find the "**Answers**" section.
* Note the **resolved IP addresses**.

🧠 Example:  
For www.google.com, you might see 142.250.182.36.

**b) Identify a DNS Query and its Corresponding Response**

🔹 **How to do it:**

* A **query** will look like:  
  css  
  CopyEdit  
    
    
  Standard query 0x3d7c A www.google.com
* The **response** will look like:  
  css  
  CopyEdit  
    
    
  Standard query response 0x3d7c A www.google.com A 142.250.182.36
* Both will have **same transaction ID** (e.g., 0x3d7c), which links the query to its answer.

**c) Was there any failed DNS resolution?**

🔹 **How to do it:**

* Look for responses indicating **"No such name"** or **"NXDOMAIN"**.
* If a query has no answer or shows an error, that means the DNS resolution **failed**.

📚 Reference for analyzing DNS responses: Wireshark DNS Errors Analysis

🔵 **Deliverables**

1. **Screenshot**
   * Capture a screenshot showing both a **DNS query** and its **response**.
2. **Table listing at least 3 domains with their resolved IPs**

|  |  |
| --- | --- |
| **Domain Name** | **Resolved IP Address** |
| [www.google.com](http://www.google.com/) | 142.250.182.36 |
| [www.wikipedia.org](http://www.wikipedia.org/) | 208.80.154.224 |
| [www.github.com](http://www.github.com/) | 20.207.73.82 |

*(Note: IP addresses might differ slightly when you perform it.)*

🔵 **Summary of Commands and Actions**

|  |  |
| --- | --- |
| **Task** | **Command/Action** |
| Start Capture | Click Blue Shark Fin |
| Visit Websites | Chrome/Firefox - Type URL |
| Stop Capture | Click Red Square |
| Filter DNS Only | Type dns in Display Filter |
| Find IP | Expand "Answers" in DNS response |

🔵 **Additional Tips:**

* If you want to **save** the capture, go to File → Save As....
* If you mistakenly capture too much, you can **restart** Wireshark and capture again.
* Always **expand** the DNS section in the packet details pane to see full info.

**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.

**SOLUTION:**

🔥 **Step-by-Step Guide**

**Step 1: Install Wireshark (if not already installed)**

👉 **Download and install Wireshark** from the official site:  
🔗 https://www.wireshark.org/download.html

**Reference:** Wireshark Official Website [Wireshark.org](https://www.wireshark.org/)

**Step 2: Start Wireshark and Begin Capture**

1. Open **Wireshark**.
2. **Select your network interface** (example: Wi-Fi if you are on a laptop).
   * Usually, the interface showing traffic numbers increasing is the active one.
3. Click on the **blue shark fin button** ("Start Capturing Packets").

✅ You are now capturing live network traffic.

**Reference:** Wireshark Documentation on Capturing Packets Wireshark User Guide

**Step 3: Open Browser and Visit a Website**

* While Wireshark is capturing traffic, open **any website** (e.g., [https://www.example.com](https://www.example.com/)).
* This will **initiate a TCP connection** — perfect for capturing the handshake.

✅ **Important:** Choose a simple website to avoid too much traffic.

**Step 4: Stop Capturing After Page Loads**

* Once the website loads fully, go back to **Wireshark**.
* Click on the **red square button** ("Stop capturing").

✅ Your capture is now paused and saved.

**Step 5: Apply Display Filter for TCP Handshakes**

You want to **filter the packets** to focus on the TCP handshake.

In the top **filter bar**, type:

ini

CopyEdit

tcp.flags.syn == 1 && tcp.flags.ack == 0

This shows **only SYN packets** (first step of handshake).

🔵 **Alternative:** You can also manually look for:

* **SYN** → SYN flag set, ACK not set.
* **SYN-ACK** → SYN and ACK both set.
* **ACK** → Only ACK flag set.

✅ This will help you **find the TCP 3-way handshake**.

**Reference:** TCP Flag Filtering in Wireshark: Wireshark Wiki - Display Filters

**Step 6: Locate a Full TCP Handshake (SYN, SYN-ACK, ACK)**

Find three packets:

1. Packet 1: **SYN** — Client → Server
2. Packet 2: **SYN-ACK** — Server → Client
3. Packet 3: **ACK** — Client → Server

**How to identify:**

* Look at **Info** column: it will say [SYN], [SYN, ACK], [ACK].
* Check **Source and Destination IP addresses** to confirm client/server.
* Look at **TCP Sequence and Acknowledgment numbers**:
  + **SYN** starts with an Initial Sequence Number (ISN).
  + **SYN-ACK** acknowledges client’s ISN and sends its own ISN.
  + **ACK** acknowledges the server’s ISN.

✅ Once you find all three packets, **take a screenshot**.

📄 **How to Answer the Questions**

➔ **1. What are the Source and Destination IP addresses and ports?**

* Right-click the packet → **"Expand Internet Protocol"** and **"Expand TCP"**.
* Note:
  + **Source IP** (client) and **Destination IP** (server).
  + **Source Port** (random client port) and **Destination Port** (usually 80 or 443 for websites).

➔ **2. How is the Sequence Number Set?**

* In the **SYN** packet:
  + Sequence Number: random initial value (called ISN).
* In the **SYN-ACK** packet:
  + Sequence Number: server's ISN.
  + Acknowledgment Number: client's ISN + 1.
* In the **ACK** packet:
  + Sequence Number: client's ISN + 1.
  + Acknowledgment Number: server's ISN + 1.

✅ You can see Sequence and Acknowledgment numbers by expanding the **TCP layer** in Wireshark.

➔ **3. Is the Handshake Completed Successfully?**

* If all three packets (SYN → SYN-ACK → ACK) are exchanged correctly without retransmissions or resets (RST flags), the handshake is successful.

✅ You can confirm by seeing that after ACK, data packets (e.g., HTTP request) start flowing.

🖌️ **What to Submit**

1. **Screenshot** of the 3 handshake packets (highlight them).
2. **Labeled Diagram** showing:
   * Client → SYN → Server
   * Server → SYN-ACK → Client
   * Client → ACK → Server
3. **Short explanation** of TCP 3-Way Handshake:
   * How it establishes a reliable connection.
   * How Sequence and Acknowledgment numbers work.
   * Why it is needed before any data transfer.

🧠 **Short Explanation: TCP 3-Way Handshake**

The **TCP 3-Way Handshake** is the process to establish a reliable connection between a client and a server.

1. **SYN**: Client sends a **SYN** (synchronize) message to start a connection, choosing a random **initial sequence number (ISN)**.
2. **SYN-ACK**: Server responds with **SYN-ACK** — acknowledging client’s SYN and sending its own SYN with its ISN.
3. **ACK**: Client sends an **ACK** back to acknowledge server's SYN.

✅ After this, the TCP connection is **established**, and data transmission can begin.

**Reference:**

* Kurose, James F., and Keith W. Ross. *Computer Networking: A Top-Down Approach*, 7th Edition, Pearson. Pearson Link
* Wireshark TCP Handshake Analysis: Wireshark TCP Analysis Guide

📋 **Quick Summary Table**

|  |  |
| --- | --- |
| **Step** | **Action** |
| 1 | Install Wireshark |
| 2 | Start capture |
| 3 | Visit a website |
| 4 | Stop capture |
| 5 | Apply TCP SYN filter |
| 6 | Find SYN, SYN-ACK, ACK packets |
| 7 | Screenshot + Labeled Diagram + Answers |

**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.

**SOLUTION:**

**Problem Statement Breakdown**

**You are asked to:**

* Write a **C++ program** that can **decrypt** a text **encrypted** using the **Vigenère Cipher**.
* **Input:**
  + A ciphertext: "LXFOPVEFRNHR"
  + A keyword: "LEMON"
* **Output:**
  + The **original plaintext** (the decrypted text).

You must submit:

* The **C++ source code** with **comments**.
* The **decrypted plaintext output**.
* A **screenshot** showing the program working correctly.

**What is a Vigenère Cipher?**

The **Vigenère cipher** is a method of encrypting text by using a series of different Caesar ciphers based on the letters of a **keyword**.

* **Encryption**: Each letter of the plaintext is shifted forward by a number of positions determined by the corresponding letter of the keyword.
* **Decryption**: Each letter of the ciphertext is shifted **backward** by the number from the corresponding keyword letter.

**Example**:

* Keyword: LEMON
* L = shift by 11 positions
* E = shift by 4 positions
* M = shift by 12 positions
* O = shift by 14 positions
* N = shift by 13 positions
* and then repeat the keyword.

**Reference:**

* Vigenère Cipher explanation on Cryptii

**Steps to Solve the Problem**

**Step 1: Read Input**

* Ask the user to enter:
  + Ciphertext (string)
  + Keyword (string)

**Step 2: Repeat/Expand the Keyword**

* If the ciphertext is longer than the keyword, repeat the keyword until it matches the length of the ciphertext.

Example:

perl

CopyEdit

Ciphertext: LXFOPVEFRNHR (length = 12)

Keyword: LEMONLEMONLE (repeat "LEMON" till length 12)

**Step 3: Decrypt Each Character**

For each character:

1. Find the **shift value** from the corresponding keyword letter:
   * A=0, B=1, C=2, ..., Z=25
2. **Shift backwards** the ciphertext letter by that number.
   * If it becomes less than 'A', **wrap around** to the end ('Z').

**Formula for Decryption**:

plaintext

CopyEdit

Decrypted\_Char = (Cipher\_Char - Key\_Char + 26) % 26 + 'A'

* Cipher\_Char and Key\_Char are numerical positions (A=0, ..., Z=25).

**Step 4: Output the Decrypted Text**

* Combine the decrypted characters into the **plaintext** string and print it.

**How Each Step Works (in C++)**

**1. Accept Inputs**

You can use getline() or cin.

cpp

CopyEdit

string ciphertext, keyword;

cout << "Enter ciphertext: ";

cin >> ciphertext;

cout << "Enter keyword: ";

cin >> keyword;

**2. Expand Keyword**

Create a new keyword string matching the ciphertext length.

cpp

CopyEdit

string newKeyword;

for (int i = 0; i < ciphertext.length(); i++) {

newKeyword += keyword[i % keyword.length()];

}

* i % keyword.length() will **repeat** the keyword.

**3. Decryption Algorithm**

Loop over the ciphertext and **decrypt each character**.

cpp

CopyEdit

string plaintext = "";

for (int i = 0; i < ciphertext.length(); i++) {

char c = ((ciphertext[i] - newKeyword[i] + 26) % 26) + 'A';

plaintext += c;

}

**Explanation:**

* ciphertext[i] - newKeyword[i] finds how much to shift back.
* +26 ensures result is positive.
* %26 ensures result stays within alphabet range.
* +'A' converts number back to uppercase letter.

**4. Print Output**

Finally:

cpp

CopyEdit

cout << "Decrypted Plaintext: " << plaintext << endl;

**Summary of Full Steps:**

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Take ciphertext and keyword as input. |
| 2 | Expand the keyword to match ciphertext length. |
| 3 | Loop through every character and decrypt using Vigenère formula. |
| 4 | Output the decrypted text. |
| 5 | Take a screenshot of successful execution. |

**Additional Important Notes**

* Assume all input is **uppercase letters** only (no spaces, no special characters).
* You should comment every part of your code (lab requirement).
* Your program must show clearly how input is taken, processed, and output is displayed.

📄 **Full C++ Code (with Comments)**

cpp

CopyEdit

#include <iostream>

using namespace std;

int main() {

// Declare variables for ciphertext and keyword

string ciphertext, keyword;

// Input ciphertext

cout << "Enter the ciphertext (in UPPERCASE, no spaces): ";

cin >> ciphertext;

// Input keyword

cout << "Enter the keyword (in UPPERCASE, no spaces): ";

cin >> keyword;

// Create a new keyword that matches the length of ciphertext

string newKeyword = "";

int keywordLength = keyword.length();

int cipherLength = ciphertext.length();

// Repeat the keyword

for (int i = 0; i < cipherLength; i++) {

newKeyword += keyword[i % keywordLength];

}

// Variable to store the decrypted plaintext

string plaintext = "";

// Decrypt each character

for (int i = 0; i < cipherLength; i++) {

// Decrypt using Vigenère formula

char decryptedChar = ((ciphertext[i] - newKeyword[i] + 26) % 26) + 'A';

plaintext += decryptedChar;

}

// Output the decrypted plaintext

cout << "\nDecrypted Plaintext: " << plaintext << endl;

return 0;

}

🛠️ **How to Run This Program**

1. Open your C++ compiler (for example, CodeBlocks, Dev C++, Visual Studio Code, etc.).
2. Create a new file, for example: vigenere\_decrypt.cpp
3. Copy the above code into the file.
4. Compile and Run.
5. Input:
   * Ciphertext: LXFOPVEFRNHR
   * Keyword: LEMON
6. **Expected Output**:  
   yaml  
   CopyEdit  
     
     
   Decrypted Plaintext: ATTACKATDAWN

✅ After that, **take a screenshot** showing the input and the correct output for submission!

**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.

**SOLUTION:**

📋 **What is to be done:**

You are required to **write a C++ program** that:

* Accepts **plaintext** (HELLOVIGENERE) and a **keyword** (KEY) from the user or code.
* **Encrypts** the plaintext using the **Vigenère Cipher** technique.
* **Displays** the encrypted **ciphertext**.
* **Decrypts** the ciphertext back into plaintext.
* Displays the **decrypted plaintext**.
* You need to **submit**:
  + The **C++ source code** (properly commented).
  + A **screenshot** of your program running successfully.
  + (Optionally) the decryption output.

🛠 **Steps to Perform:**

|  |  |  |
| --- | --- | --- |
| **Step No.** | **Task** | **Explanation** |
| 1 | Understand Vigenère Cipher | It is a method of encrypting text by applying a series of Caesar Ciphers based on the letters of a keyword. |
| 2 | Create a C++ Program | You’ll code in C++ using cin, cout, loops, and string handling. |
| 3 | Take Inputs | Input plaintext and keyword (either hardcoded or user-input). |
| 4 | Generate Repeating Keyword | If the keyword is shorter than the plaintext, repeat it to match the length. |
| 5 | Encryption Process | Encrypt by shifting letters based on the keyword. |
| 6 | Display Ciphertext | Show the encrypted message. |
| 7 | Decryption Process | Reverse the shifting to get original plaintext. |
| 8 | Display Decrypted Plaintext | Prove that decryption works correctly. |
| 9 | Submit Code and Screenshot | Comment the code and take a screenshot when running correctly. |

🧠 **Detailed Explanation of Each Step:**

**1. Understand the Vigenère Cipher**

* **Plaintext** is the normal text (e.g., HELLOVIGENERE).
* **Keyword** is a secret word (e.g., KEY).
* Each letter of the plaintext is shifted according to the corresponding letter in the keyword.
* **Example**:
  + H + K → (H shifted by K positions) → R
  + E + E → I
  + L + Y → J
  + (and so on)

🔵 **Formula for Encryption**:

Cipher[i] = (Plaintext[i] + Keyword[i]) % 26

🔵 **Formula for Decryption**:

Plaintext[i] = (Ciphertext[i] - Keyword[i] + 26) % 26

(both using 0-based position: A=0, B=1, ..., Z=25)

**2. Create C++ Program Structure**

* Include libraries: #include<iostream>, #include<string>.
* Create functions:
  + string encrypt(string plaintext, string keyword)
  + string decrypt(string ciphertext, string keyword)

**3. Take Inputs**

You will input (hardcoded for this assignment):

cpp

CopyEdit

string plaintext = "HELLOVIGENERE";

string keyword = "KEY";

**4. Generate Repeating Keyword**

Since keyword is shorter, repeat it:

Plaintext: HELLOVIGENERE  
Keyword: KEYKEYKEYKEYKE

You will use a loop to **extend the keyword** to match the plaintext length.

**5. Encryption Process**

For each character:

* Find the **position** of plaintext letter and keyword letter.
* Add their positions.
* Take modulo 26.
* Convert back to a letter.

**6. Display Ciphertext**

Print the resulting encrypted string.

**7. Decryption Process**

Reverse the process:

* Subtract the keyword position from the ciphertext letter position.
* Add 26 (to handle negatives) and take modulo 26.
* Convert back to a letter.

**8. Display Decrypted Plaintext**

To verify correctness, print the decrypted text — it should match original HELLOVIGENERE.

**9. Screenshot and Submission**

* Run the program.
* Take a clear screenshot showing input, encrypted output, and decrypted output.

📄 **Here’s a Full C++ Code Example (with comments):**

cpp

CopyEdit

#include<iostream>

#include<string>

using namespace std;

// Function to generate the key in a cyclic manner until it's the same length as the plaintext

string generateKey(string plaintext, string keyword) {

int pLen = plaintext.size();

int kLen = keyword.size();

string key = keyword;

for (int i = 0; i < pLen - kLen; i++) {

key.push\_back(keyword[i % kLen]);

}

return key;

}

// Function to encrypt the plaintext

string encrypt(string plaintext, string key) {

string cipherText;

for (int i = 0; i < plaintext.length(); i++) {

char x = ((plaintext[i] + key[i]) % 26) + 'A';

cipherText.push\_back(x);

}

return cipherText;

}

// Function to decrypt the ciphertext

string decrypt(string cipherText, string key) {

string origText;

for (int i = 0; i < cipherText.length(); i++) {

char x = ((cipherText[i] - key[i] + 26) % 26) + 'A';

origText.push\_back(x);

}

return origText;

}

int main() {

string plaintext = "HELLOVIGENERE";

string keyword = "KEY";

string key = generateKey(plaintext, keyword);

string cipherText = encrypt(plaintext, key);

cout << "Encrypted Ciphertext: " << cipherText << endl;

string decryptedText = decrypt(cipherText, key);

cout << "Decrypted Plaintext: " << decryptedText << endl;

return 0;

}

📸 **Example Output:**

pgsql

CopyEdit

Encrypted Ciphertext: RIJVSAMYGMVHZR

Decrypted Plaintext: HELLOVIGENERE

**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.

**SOLUTION:**

📄 **Explanation of Problem Statement**

You are asked to **secure a basic network** (a router, a switch, and a PC) using **Cisco Packet Tracer** by doing the following:

|  |  |  |
| --- | --- | --- |
| **Task** | **Device(s)** | **Action** |
| 1. Set console & VTY passwords | Router (R1) & Switch (S1) | Password for local and remote (Telnet) access |
| 2. Set a password for privileged EXEC mode | Router (R1) & Switch (S1) | Set an "enable secret" password (used to enter privileged mode #) |
| 3. Configure a Banner MOTD (Message of the Day) | Router (R1) & Switch (S1) | Display a warning to unauthorized users |
| 4. Save and capture screenshots | Whole network | Take screenshots showing the login prompts and configuration |

Deliverables:

* **.pkt file** (your network file)
* **Screenshots**:
  + Console login prompt.
  + VTY login via Telnet.
  + show running-config showing encrypted passwords.
* **Short Explanation**: Why each step is important.

🔵 **Console password** — protects physical access (direct connection via console cable). 🔵 **VTY password** — protects remote access (Telnet sessions). 🔵 **Enable secret** — protects high-level access to configuration (privileged mode). 🔵 **Banner MOTD** — legal warning to unauthorized users.

🖥️ **Environment Setup (Kali Linux)**

Since you're using **Kali Linux**, here's what you need:

1. **Install Cisco Packet Tracer** on Kali Linux  
   (Packet Tracer is a Cisco simulation tool, it’s not pre-installed.)  
   **How to Install Cisco Packet Tracer:**
   * **Step 1:** Go to Cisco's official Netacad page and download Packet Tracer:  
     🔗 Cisco Networking Academy Login  
     (You need to create a free account if you don't have one.)
   * **Step 2:** Install dependencies:  
     bash  
     CopyEdit  
       
       
     sudo apt update
   * sudo apt install libqt5webkit5 libqt5multimedia5 libqt5multimediawidgets5 libdouble-conversion3 libssl1.1
   * **Step 3:** Install Packet Tracer: (Assuming you downloaded a .deb file)  
     bash  
     CopyEdit  
       
       
     sudo dpkg -i CiscoPacketTracer\_8.x.x\_amd64.deb
   * sudo apt --fix-broken install
   * **Step 4:** Launch Packet Tracer:  
     bash  
     CopyEdit  
       
       
     packettracer
2. 📚 **Reference**:
   * Official Packet Tracer Download - Cisco NetAcad
   * Installing Packet Tracer on Linux - Cisco Community Guide

⚡ **Steps to Perform in Packet Tracer**

**1. Create the Topology:**

* **Drag & drop**:
  + 1 Router → "Generic Router" (e.g., 2911).
  + 1 Switch → "2960 Switch".
  + 1 PC.
* **Connect** with cables:
  + PC to Switch (use Copper Straight-Through Cable).
  + Switch to Router (use Copper Straight-Through Cable).

⚙️ **Configuration Commands (on Router and Switch)**

➔ **On Router R1:**

1. **Access the CLI** (Console tab).
2. **Enter configuration mode**:  
   bash  
   CopyEdit  
     
     
   enable
3. configure terminal
5. **Set Console Password**:  
   bash  
   CopyEdit  
     
     
   line console 0
6. password cisco123
7. login
8. exit
10. **Set VTY (Telnet) Password**:  
    bash  
    CopyEdit  
      
      
    line vty 0 4
11. password telnet123
12. login
13. exit
15. **Set Enable Secret Password**:  
    bash  
    CopyEdit  
      
      
    enable secret secureadmin
17. **Set Banner MOTD**:  
    bash  
    CopyEdit  
      
      
    banner motd #Unauthorized access is prohibited!#
19. **Save Configuration**:  
    bash  
    CopyEdit  
      
      
    end
20. write memory

➔ **On Switch S1 (same steps, but different passwords):**

1. **Access CLI**.
2. **Enter configuration mode**:  
   bash  
   CopyEdit  
     
     
   enable
3. configure terminal
5. **Set Console Password**:  
   bash  
   CopyEdit  
     
     
   line console 0
6. password switch123
7. login
8. exit
10. **Set VTY Password**:  
    bash  
    CopyEdit  
      
      
    line vty 0 4
11. password remotesw
12. login
13. exit
15. **Set Enable Secret**:  
    bash  
    CopyEdit  
      
      
    enable secret adminsw
17. **Set Banner MOTD**:  
    bash  
    CopyEdit  
      
      
    banner motd #Unauthorized access is prohibited!#
19. **Save Configuration**:  
    bash  
    CopyEdit  
      
      
    end
20. write memory

📸 **Required Screenshots**

1. **Console Login Prompt**  
   When you reboot the router or switch, and connect via console, it should ask for **Console Password**.
2. **VTY Login via Telnet**  
   Telnet from PC1:  
   bash  
   CopyEdit  
     
     
   telnet <router/switch IP address>
3. It should ask for the **VTY password**.
4. **show running-config** Output  
   Run on Router and Switch:  
   bash  
   CopyEdit  
     
     
   show running-config
5. Take a screenshot where you see **encrypted passwords**.

✍️ **Short Explanation (for your report)**

Here’s a short sample explanation you can expand:

|  |  |
| --- | --- |
| **Security Step** | **Importance** |
| Console Password | Prevents unauthorized local access to device settings. |
| VTY Password | Secures remote access (via Telnet) to the devices. |
| Enable Secret | Protects the highest privilege mode where full device control is possible. |
| Banner MOTD | Legal warning to prevent unauthorized usage and inform users of monitoring policies. |

📚 **Sources / References**

* Cisco Official Guide: Password Protection and Security Best Practices
* Cisco Learning: Configure Password Protection on Cisco Devices
* Packet Tracer Installation on Linux: Cisco Packet Tracer Installation - Community Guide

✅ **In Summary**

1. Install Packet Tracer.
2. Build the network (Router - Switch - PC).
3. Configure passwords, enable secret, banner.
4. Test console and Telnet access.
5. Take screenshots.
6. Save your .pkt file and prepare a short explanation.

**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

Subnet Mask

Default Gateway

MAC Address (Physical Address)

**2. Test Connectivity**

Ping your default gateway

Ping a public DNS server

Ping a website:

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

**3. Trace the Route to a Remote Host**

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

**4. Check DNS Resolution**

Record the IP address(es) returned.

**5. View Active Connections and Listening Ports Using** netstat

Answer the following:

How many **TCP connections** are in the **ESTABLISHED** state?

Are there any services listening on port **80** or **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.

**SOLUTION:**

🖥️ **Problem Statement Explained (Step-by-Step)**

🔹 **Objective:**

You are learning to **analyze**, **monitor**, and **troubleshoot** network connectivity issues using **Windows Command Line Tools** (CMD).

🛠️ **Steps to Perform Each Task**

**1. View Network Configuration**

**Goal:** Find out your computer’s network settings:

* IP Address
* Subnet Mask
* Default Gateway
* MAC Address (Physical Address)

**Command to Use:**

bash

CopyEdit

ipconfig /all

**How to Do It:**

* Press Windows + R, type cmd, and hit Enter to open the **Command Prompt**.
* Type ipconfig /all and press Enter.
* Look under your active network adapter (e.g., **Ethernet adapter** or **Wi-Fi**).
* **Note down** the following:
  + **IPv4 Address** = IP Address
  + **Subnet Mask** = Subnet Mask
  + **Default Gateway** = Router IP
  + **Physical Address** = MAC Address

✅ **Take a screenshot** of the output.

**2. Test Connectivity**

**Goal:** Check if your computer can "talk" to your network and the internet.

**Commands to Use:**

bash

CopyEdit

ping <target>

Where <target> is your:

* Default gateway IP
* Public DNS server (e.g., 8.8.8.8 = Google's DNS)
* Website (e.g., [www.google.com](http://www.google.com/))

**How to Do It:**

* **Ping Default Gateway**:  
  bash  
  CopyEdit  
    
    
  ping <your-default-gateway-ip>
* **Ping Public DNS Server**:  
  bash  
  CopyEdit  
    
    
  ping 8.8.8.8
* **Ping a Website**:  
  bash  
  CopyEdit  
    
    
  ping www.google.com

**What to Observe:**

* **Response Times** (in ms)
* **Packet Loss** (% loss)

✅ **Record response times** and **check if there is packet loss**.  
✅ **Take a screenshot** of each ping.

**3. Trace the Route to a Remote Host**

**Goal:** See the path your packets take across the internet to reach a remote server.

**Command to Use:**

bash

CopyEdit

tracert www.google.com

**How to Do It:**

* In CMD, type:  
  bash  
  CopyEdit  
    
    
  tracert www.google.com
* Press Enter.

**What to Observe:**

* Each **hop** represents a device/router the packet goes through.
* **Count the number of hops** till it reaches the destination.

✅ **Take a screenshot** of the tracert output.

**4. Check DNS Resolution**

**Goal:** Find out which IP addresses correspond to a website.

**Command to Use:**

bash

CopyEdit

nslookup www.google.com

**How to Do It:**

* In CMD, type:  
  bash  
  CopyEdit  
    
    
  nslookup www.google.com
* Press Enter.

**What to Record:**

* The **IP address** returned by the DNS server for the website.

✅ **Take a screenshot** of the nslookup output.

**5. View Active Connections and Listening Ports**

**Goal:** Check which network connections are active and which ports your computer is listening to.

**Command to Use:**

bash

CopyEdit

netstat -an

**How to Do It:**

* In CMD, type:  
  bash  
  CopyEdit  
    
    
  netstat -an
* Press Enter.

**What to Observe:**

* Find **TCP connections** that are **ESTABLISHED** (you can count them manually).
* Look for any service **LISTENING** on:
  + **Port 80** (HTTP)
  + **Port 443** (HTTPS)

✅ **Take a screenshot** of the netstat output.

📋 **Deliverables**

You must submit:

* **Screenshots** of all command outputs.
* **Answers** to each task:
  + IP, Subnet Mask, Gateway, MAC Address.
  + Ping results: response times and packet loss.
  + Number of hops in traceroute.
  + IP addresses found with nslookup.
  + TCP connections and services on port 80/443.
* **Conclusion:**
  + Summarize if your network is working fine or if there were any issues like high ping, packet loss, DNS problems, etc.

**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)**

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).

**2.Capture Only HTTP Traffic**

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.

**3. Capture and Analyze TCP Handshake**

Run **TCPDump** to capture only **TCP traffic**.

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.

**SYN-ACK**: Sent by the server.

**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?

**SOLUTION:**

🔥 **Step-by-Step Execution:**

**Step 1: Install TCPDump**

* If not installed yet, install TCPDump:

**On Ubuntu/Linux:**

bash

CopyEdit

sudo apt update

sudo apt install tcpdump

**On MacOS:**

bash

CopyEdit

brew install tcpdump

**On Windows:**

* You can use TCPDump via **WSL (Windows Subsystem for Linux)**, or install **WinDump** (Windows version).
* WinDump Download

**Step 2: Find Your Active Network Interface**

Before capturing, you must know which network interface to monitor (e.g., eth0, wlan0, etc.)

Use this command:

bash

CopyEdit

tcpdump -D

It will list all network interfaces.  
Pick the active one — usually it will say "up" or be the one connected to WiFi or Ethernet.

Example output:

arduino

CopyEdit

1. eth0 (Ethernet)

2. wlan0 (WiFi)

Suppose your active interface is wlan0.

🚀 **Now Perform the Tasks:**

**Task 1: Capture All Network Traffic (Basic Capture)**

**What you must do:**

* Capture **everything** on the network.
* Let it run for **10 seconds**.
* See what kinds of packets appear (e.g., ARP, IP, TCP, UDP).

**Commands:**

bash

CopyEdit

sudo tcpdump -i wlan0

* -i wlan0 → listen on wlan0 interface.

**Now:**

* Let it capture.
* After about 10 seconds, **press Ctrl+C** to stop.

**What you'll see:**

* A bunch of lines like:

nginx

CopyEdit

IP 192.168.1.5.52462 > 8.8.8.8.53: UDP, length 29

ARP, Request who-has 192.168.1.1 tell 192.168.1.5

* You can **identify** packet types: **ARP**, **IP**, **UDP**, **TCP**.

**Task 2: Capture Only HTTP Traffic**

**What you must do:**

* Capture **only HTTP traffic** (which uses port **80**).
* Open a **website** (example: http://example.com) during capture.
* Analyze **HTTP packets** (like GET, POST).

**Commands:**

bash

CopyEdit

sudo tcpdump -i wlan0 port 80 -A

**Explanation:**

* port 80 → only capture traffic on port 80 (HTTP).
* -A → print packets in **ASCII** so you can read text (like HTTP GET/POST easily).

**Now:**

1. Run the command.
2. Open a browser → visit http://example.com.
3. Let traffic generate.
4. After 10-15 seconds, **Ctrl+C** to stop.

**What you'll see:**

* **GET / HTTP/1.1** requests
* **Host: example.com** headers
* **Response data**

Sample output:

makefile

CopyEdit

GET / HTTP/1.1

Host: example.com

User-Agent: Mozilla/5.0 ...

**Task 3: Capture and Analyze TCP Handshake**

**What you must do:**

* Capture **only TCP traffic**.
* Open a website (like https://google.com — HTTPS uses TCP).
* Find **SYN**, **SYN-ACK**, and **ACK** packets (TCP three-way handshake).

**Commands:**

bash

CopyEdit

sudo tcpdump -i wlan0 tcp

**Now:**

1. Run the above command.
2. Quickly open https://www.google.com or any site.
3. After about 10 seconds, **Ctrl+C** to stop.

**What to observe:**

* You should find 3 handshake steps:
  + SYN — Sent by your machine.
  + SYN-ACK — Server responds.
  + ACK — Your machine confirms.

Example:

ruby

CopyEdit

192.168.1.5.54321 > 172.217.194.138.443: Flags [S], seq 12345678

172.217.194.138.443 > 192.168.1.5.54321: Flags [S.], seq 98765432, ack 12345679

192.168.1.5.54321 > 172.217.194.138.443: Flags [.], ack 98765433

Flags Meaning:

* [S] → SYN
* [S.] → SYN-ACK
* [.] → ACK

📸 **Deliverables (Screenshots & Report)**

You need to submit:

|  |  |
| --- | --- |
| **Deliverable** | **What to Screenshot** |
| TCPDump outputs | Terminal outputs of each task |
| Analysis of key packets | Screenshot SYN, SYN-ACK, ACK packets |
| Specific packet contents | Screenshot HTTP GET request / DNS query |

**Write an Analysis Report:**

Answer these questions:

* What type of traffic was most common? (e.g., TCP? ARP?)
* What were the Source & Destination IPs for TCP handshake?
* What domain name was resolved through DNS queries? (for example, google.com will show in DNS packets)
* Explain each step in your own words (e.g., SYN packet is for requesting connection, etc.)

✅ **Quick Summary of Commands**

|  |  |
| --- | --- |
| **Task** | **Command** |
| Basic Capture | sudo tcpdump -i wlan0 |
| HTTP Traffic Only | sudo tcpdump -i wlan0 port 80 -A |
| TCP Handshake Only | sudo tcpdump -i wlan0 tcp |