Project Report: Basic Network Sniffer

1. Introduction

This report details the successful completion of **Task 1: Basic Network Sniffer** for the **CodeAlpha Cyber Security Internship Program**. The objective of this task was to develop a Python program capable of capturing and analyzing network traffic packets, thereby providing insights into network data flow and fundamental protocol structures.

2. Project Overview (Task 1 Requirements)

The core requirements for Task 1 were to:

- Build a Python program to capture network traffic packets.
- Analyze captured packets to understand their structure and content.
- Learn how data flows through the network and the basics of protocols.
- Utilize libraries such as scapy or socket for packet capturing.
- Display useful information including source/destination IPs, protocols, and payloads.

3. Technical Implementation

The project was implemented using Python, leveraging the powerful scapy library for network packet manipulation.

3.1. Key Functionalities

The developed program (packet_analyzer.py) performs the following operations:

- **Packet Capture:** It uses scapy.all.sniff() to listen for and intercept network packets on a specified or default network interface.
- **Packet Processing (Callback Function):** A packet_callback function is defined and passed to sniff(). This function is executed for every packet captured, allowing for real-time analysis.
- **Layer-wise Analysis:** The packet_callback function intelligently inspects different layers of the OSI model present in each packet (e.g., Ethernet, IP, TCP, UDP, ICMP).
- **Information Extraction and Display:** For each packet, the program extracts and prints the following useful information:
    - **Ethernet Layer:** Source MAC Address, Destination MAC Address, EtherType.
    - **IP Layer:** Source IP Address, Destination IP Address, Protocol Number (e.g., 6 for TCP, 17 for UDP, 1 for ICMP).

- **Transport Layer (TCP/UDP/ICMP):**
  - **TCP:** Source Port, Destination Port, TCP Flags.
  - **UDP:** Source Port, Destination Port.
  - **ICMP:** ICMP Type, ICMP Code.
- **Payload:** Attempts to display the raw payload content if present in the packet (for TCP, UDP, ICMP, or other IP protocols).

3.2. Libraries Used

- **scapy**: An indispensable Python library for packet manipulation. It allows for creating, sending, sniffing, and dissecting network packets. Its high-level abstraction simplifies complex network operations.

4. Setup and Execution Guide

To run the network sniffer program, follow these steps:

4.1. Prerequisites

- **Python 3.x:** Ensure Python is installed on your system.
- **Scapy:** The Python library required for packet capture and analysis.
- **Npcap (for Windows users):** A necessary packet capture driver for Windows, replacing the older WinPcap.

4.2. Installation Steps

1. **Install Scapy:**
   pip install scapy

2. **Install Npcap (Windows Only):**
   - Download the latest stable installer from https://nmap.org/npcap/.
   - Run the installer and **ensure "Install Npcap in WinPcap API-compatible Mode" is checked** during the installation process.
   - It is highly recommended to **restart your computer** after installing Npcap to ensure the driver is fully loaded.

4.3. Running the Program

1. **Save the code:** Save the provided Python code (from the previous interaction) as packet_analyzer.py.
2. **Open Terminal/Command Prompt with Administrator/Root Privileges:**
   - **On Linux/macOS:** Open your terminal and run:
     sudo python packet_analyzer.py

     (You will be prompted for your password.)

- **On Windows:** Right-click on the "Command Prompt" or "PowerShell" shortcut in your Start Menu and select "Run as administrator." Navigate to the directory where you saved packet_analyzer.py and run:
  python packet_analyzer.py

3. **Observe Traffic:** The program will begin capturing packets. Generate some network activity (e.g., open a web browser, visit a website, ping an IP address) to see the output. To stop the sniffer, press Ctrl+C.

5. Project Code

Submit the code

6. Learning Outcomes

Through the development and execution of this project, significant learning was achieved in the following areas:

- **Network Fundamentals:** Gained a practical understanding of how data is encapsulated and transmitted across a network.
- **Packet Structure:** Learned to identify and dissect different layers within a network packet (Ethernet, IP, TCP/UDP/ICMP).
- **Protocol Basics:** Reinforced knowledge of common network protocols and their roles in communication.
- **Scapy Proficiency:** Developed hands-on experience with scapy, a crucial tool for network security professionals for tasks like sniffing, crafting, and analyzing packets.
- **Troubleshooting:** Encountered and resolved common issues related to permissions and driver installations in network programming environments.

7. Conclusion

The "Basic Network Sniffer" project successfully meets all the requirements of Task 1 for the CodeAlpha Cyber Security Internship. It provides a functional tool for basic network traffic analysis and serves as a strong foundation for understanding more advanced cybersecurity concepts. The experience gained in building and troubleshooting this application has been invaluable in solidifying practical cybersecurity skills.