**Ijalusi Emmanuel cyber security internship at code CodeAlpha**

✅ TASK 1: Basic Network Sniffer

Packet Capture and Analysis with Python

The aim of this project is to capture live network traffic, analyze packet details such as source and destination addresses, protocols, and ports, and gain a better understanding of how data flows through a network

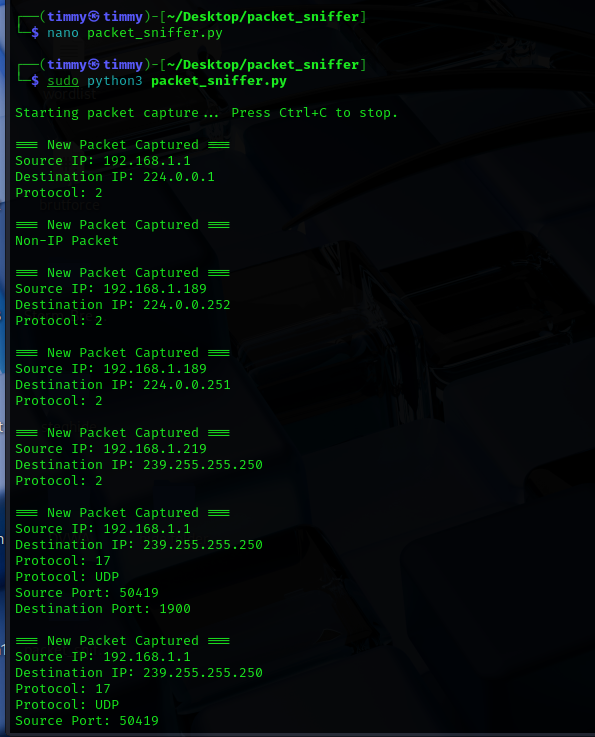
**Tools and Environment**

* **Operating System**: Kali Linux (chosen because it is designed for penetration testing and networking tasks).
* **Programming Language**: Python 3
* **Library**: scapy (used for sniffing and analyzing packets)

I created a Python script named **packet\_sniffer.py**. This program uses scapy to capture packets and analyze them in real time.

Key parts of the script:

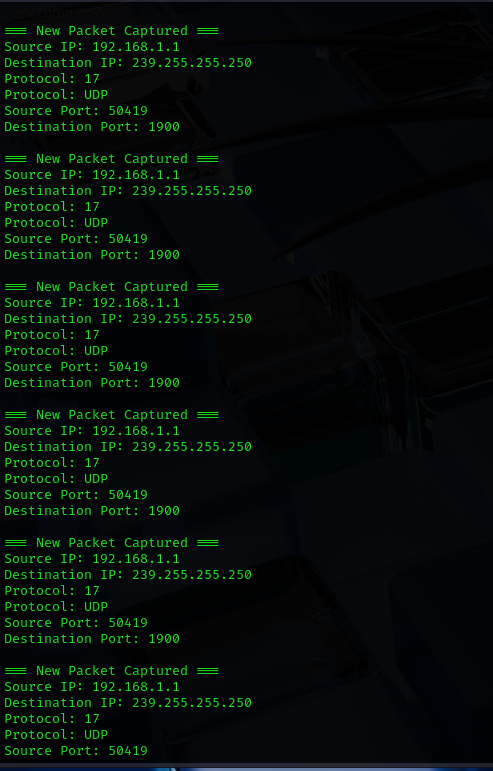
* **Packet Capture**: The sniff() function listens to the network and captures packets.
* **Analysis**: A callback function (analyze\_packet) processes each packet and prints useful details.
* **Details Shown**:
  + Source IP address
  + Destination IP address
  + Protocol (TCP, UDP, ICMP, or others)
  + Ports (source and destination) if applicable



 A device in the local network (192.168.1.5) was communicating with a server on the internet (Google server).

 The connection was using **TCP** protocol.

 Destination port **443**, which means HTTPS (secure web traffic).



This analysis helped me see how different types of traffic behave:

* **TCP packets** → mostly web browsing, SSH, or email.
* **UDP packets** → often DNS queries, streaming, or VoIP traffic.
* **ICMP packets** → pings and diagnostic traffic.

By capturing real packets, I gained insight into:

* Who is communicating with who on the network.
* What protocols and ports are being used.
* How data flows between local devices and external servers.

This project demonstrated how Python can be used as a lightweight packet sniffer with the help of Scapy. It showed the structure of packets in real time, which deepened my understanding of protocols like TCP, UDP, and ICMP.

**How Data Flows Through a Network and the Basics of Protocols**

When devices communicate across a network, data does not simply travel in one big piece. Instead, it is broken down into smaller units called **packets**. Each packet carries not only the actual data (payload) but also important details like where it is coming from, where it should go, and which rules (protocols) should be followed to make sure it reaches the right destination.

### ****1. How Data Flows in a Network****

The flow of data follows a structured path:

1. **Application Layer** – This is where communication starts. For example, when we type a web address into a browser, it creates an HTTP/HTTPS request.
2. **Transport Layer** – The data is divided into segments and assigned port numbers. For example, web traffic usually uses port 80 (HTTP) or 443 (HTTPS).
3. **Network Layer** – IP addresses are added so the data knows the source (where it came from) and destination (where it should go).
4. **Data Link Layer** – The packet is converted into frames with MAC addresses so it can travel across the local network.
5. **Physical Layer** – Finally, the data is turned into electrical signals, light pulses, or radio waves (depending on whether it is wired, fiber, or wireless) and transmitted.

### ****2. Basics of Network Protocols****

Protocols are simply **rules that govern how data is transmitted, received, and understood**. Each protocol has a specific job, and they often work together. Some common ones include:

* **HTTP/HTTPS** – Used for web browsing (HTTPS adds encryption for security).
* **TCP (Transmission Control Protocol)** – Ensures reliable communication by checking that all packets arrive in the correct order.
* **UDP (User Datagram Protocol)** – A faster but less reliable method, often used for streaming, gaming, or voice calls.
* **ICMP (Internet Control Message Protocol)** – Used for error reporting and diagnostics (e.g., the ping command).
* **DNS (Domain Name System)** – Translates human-readable addresses (like www.example.com) into IP addresses that computers can understand.

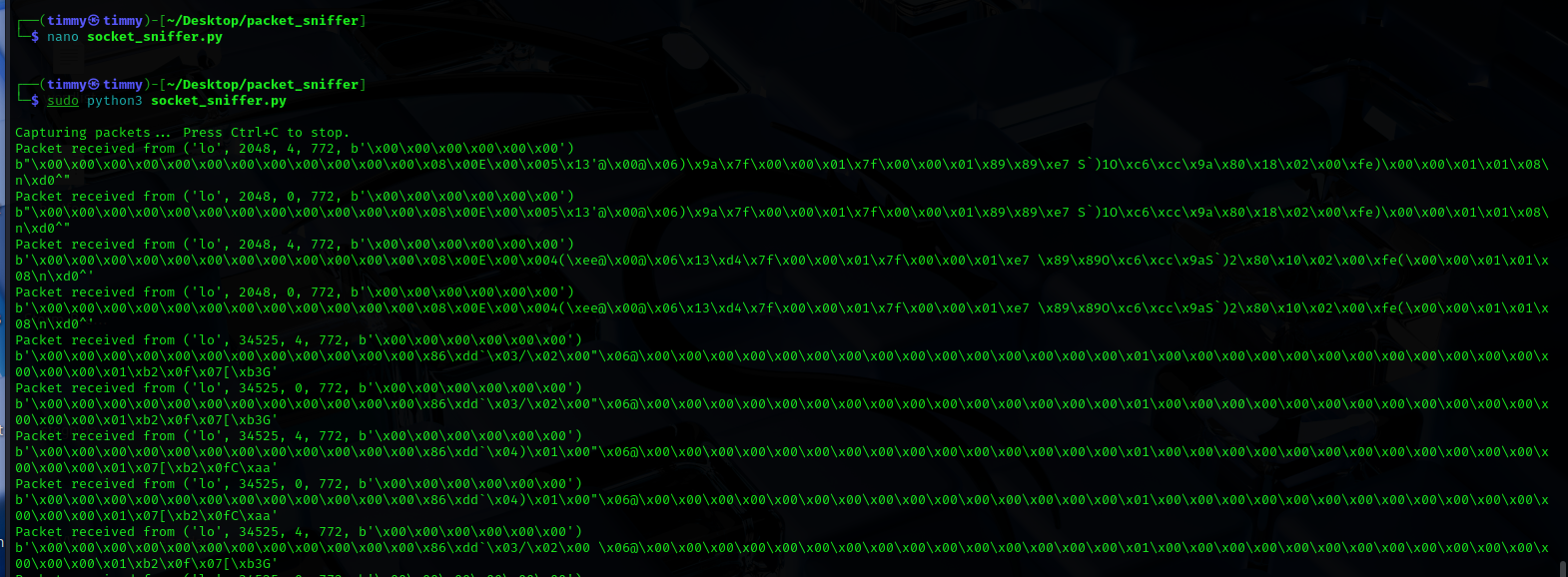
# **Using Python Libraries for Packet Capturing and Analysis**

To understand how data moves across a network, I used Python libraries such as **Scapy** and **Socket**. These libraries allow us to capture raw network packets, analyze their contents, and display useful details such as source/destination IP addresses, protocols, and even parts of the payload.

**Scapy**: A powerful packet manipulation tool that makes it easy to sniff, dissect, and analyze network traffic with just a few lines of code.

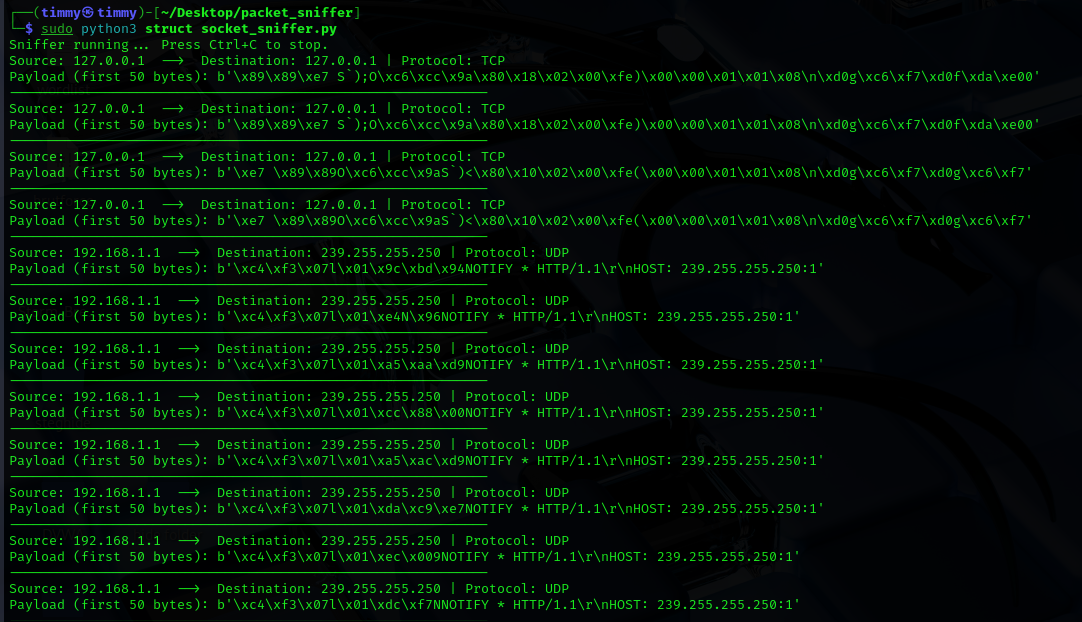


**Socket**: A lower-level Python library that gives direct access to raw network packets, allowing us to build our own packet sniffer from scratch.



* **Scapy** → Easy to capture and analyze packets in readable form.
* **Socket** → Gives raw packet data, good for understanding packet structure at a low level.

**source/destination IPs, protocols and payloads.**



## How it works:

1. **Ethernet frame** → Extracts MAC headers (not printed, but needed to parse further).
2. **IPv4 header** → Extracts Source IP, Destination IP, and Protocol type.
3. **Protocol detection** → Maps protocol numbers (1=ICMP, 6=TCP, 17=UDP).
4. **Payload** → Shows first 50 bytes of packet payload for analysis.