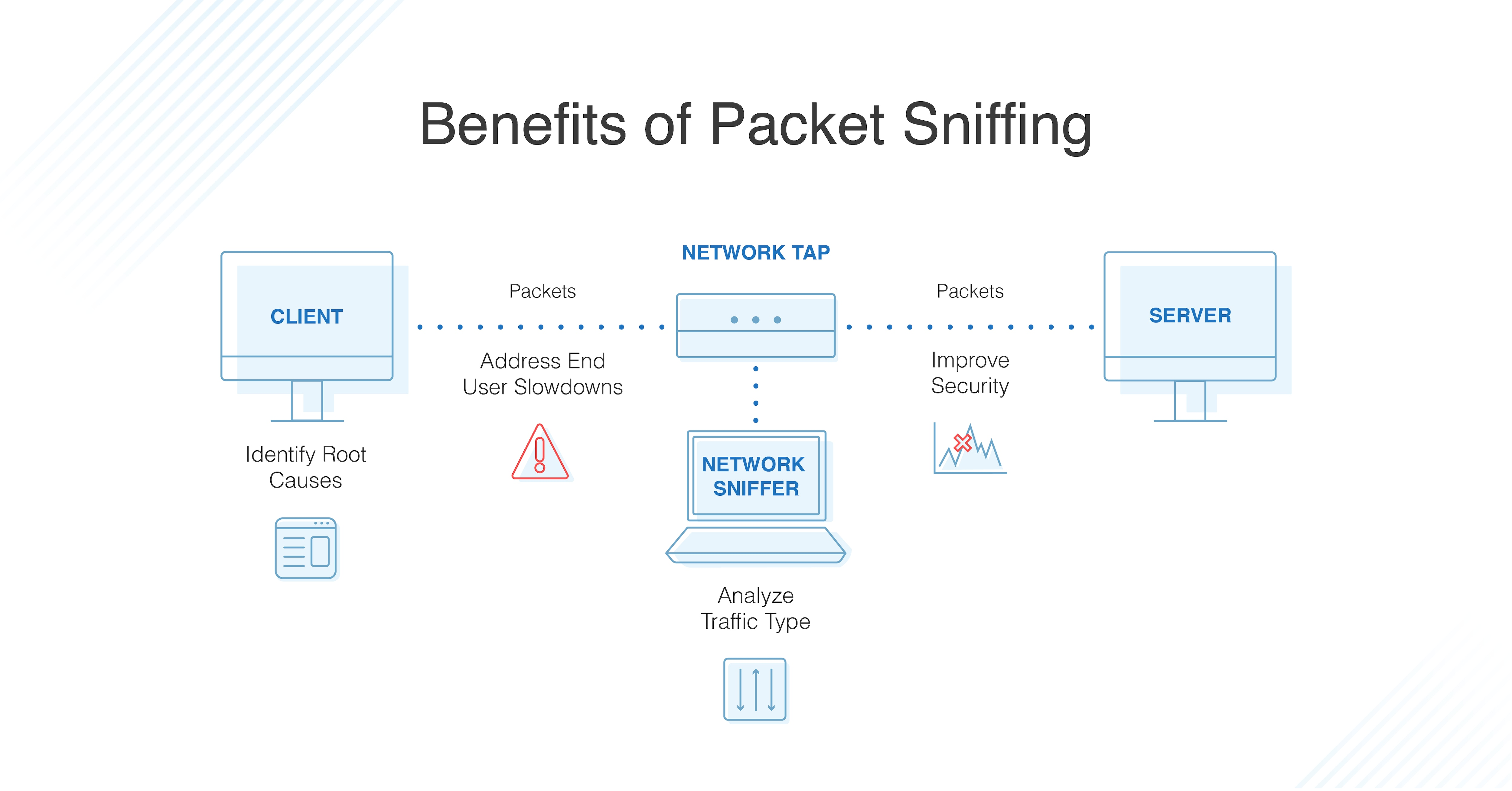
**------------------------------------------------------------------------------------ CodeAlpha**

**BASIC NETWORK SNIFFER**



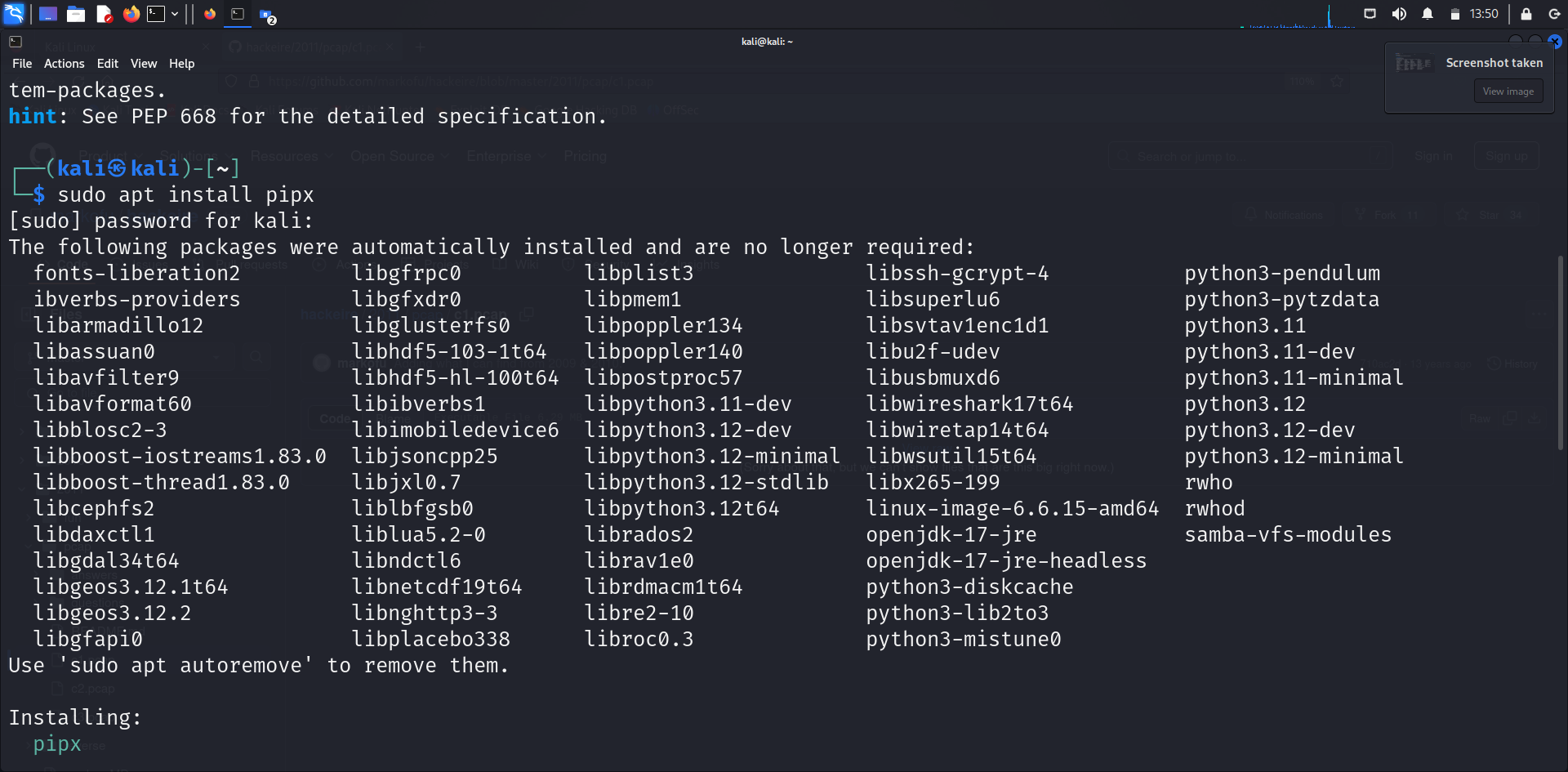
**Network Packet Sniffer**

A network packet sniffer is a tool that captures and analyzes data traveling across a network. It intercepts digital "packets" (small units of data) as they move between devices. The sniffer captures packets at the data link layer and can analyze various protocol headers to extract key information, such as source and destination IP addresses, ports, and packet payloads. Network packet sniffers are commonly used by IT teams to troubleshoot network issues and by cybersecurity professionals to detect hacking attempts and monitor for suspicious activity. Proper network security measures, such as Virtual Private Networks (VPNs) and encryption, help protect against unauthorized packet sniffing.

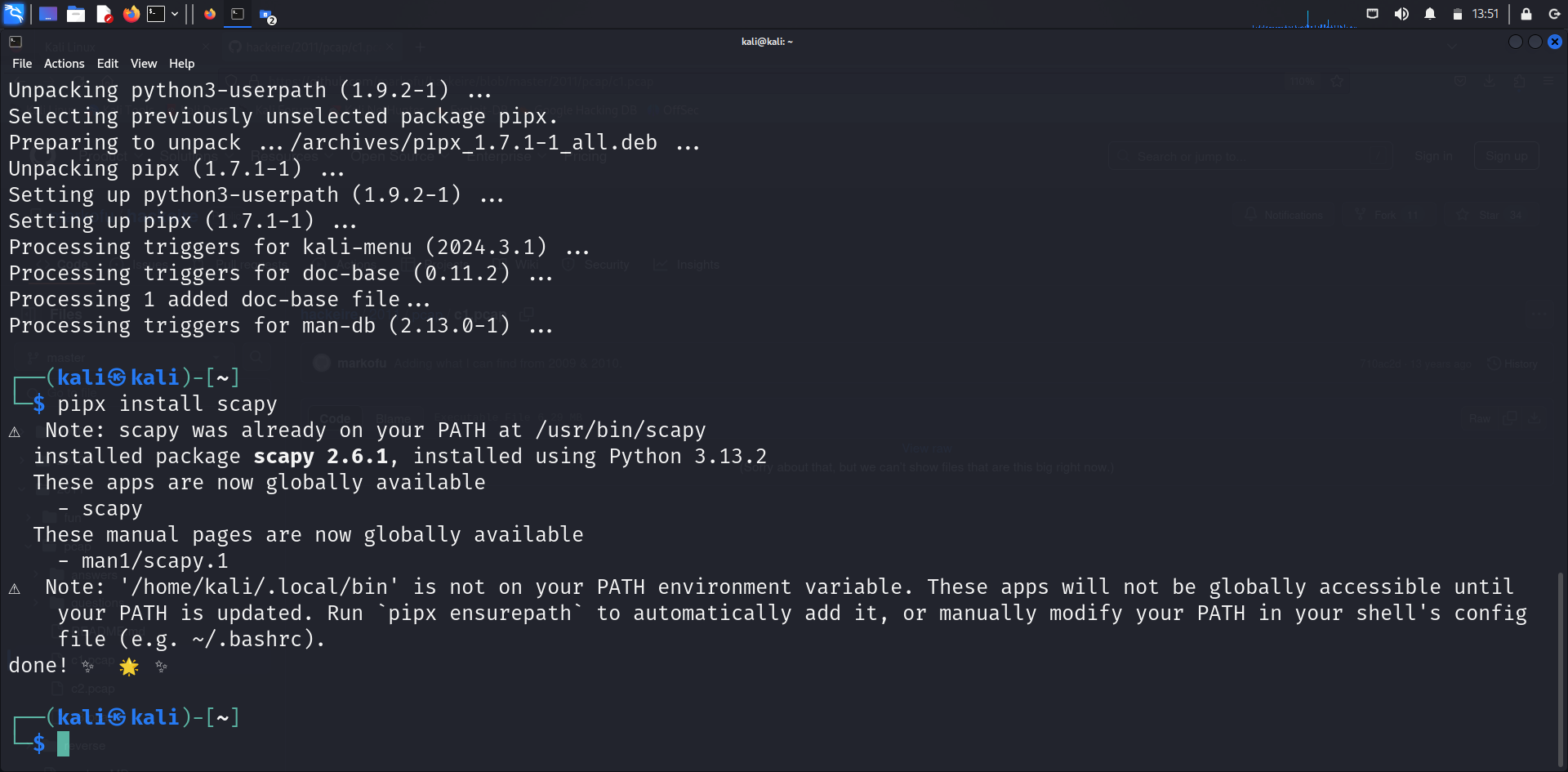
**Basic network sniffer in Python**

To build a basic network sniffer in Python, we can use the scapy library, which is a powerful tool for network packet manipulation and analysis. Scapy is a powerful interactive packet manipulation library written in Python. Scapy is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more.

**1. Installing Scapy**

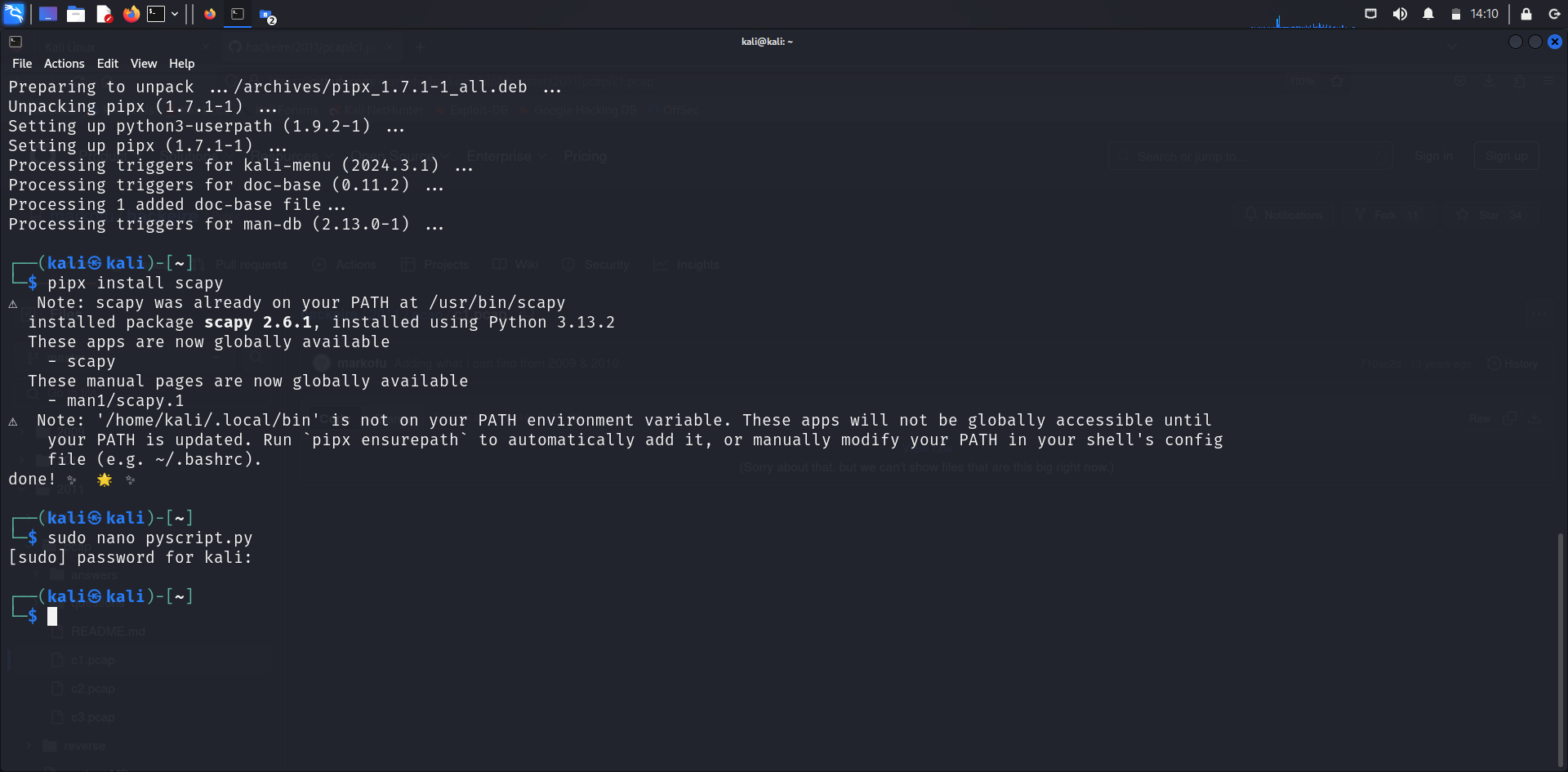


| pipx install scapy



**2. Writing Python script**

| sudo nano pyscript.py





This Python script utilizes the powerful scapy library to create a simple yet effective network sniffer capable of analyzing live traffic. When a packet is captured, it is passed to the packet\_handler() function, which examines its structure and extracts relevant details. The handler first checks whether the packet contains an IP layer, and if so, it prints the source and destination IP addresses, giving an overview of where the traffic is coming from and going to.

If the packet is identified as a TCP packet, the script further extracts the source and destination TCP port numbers, providing insight into the services or applications in use. In the case of UDP packets, it performs a similar task by retrieving and printing the UDP port information. This makes the tool versatile for monitoring both TCP and UDP traffic.

The sniffing process is initialized through the start\_sniffing() function, which uses Scapy's built-in sniff() method. This method captures packets in real time and applies the custom handler function to each, allowing on-the-fly analysis without storing packet data in memory. When the script is executed, it immediately begins monitoring the network, serving as a foundational tool for traffic inspection, debugging, or even security research.

**3. Script execution**

| sudo python3 pyscript.py

