**Title: Basic Network Sniffer in Python**

**Objective**: Build a Python-based network sniffer to capture and analyze network traffic, improving understanding of data flow and packet structures.

**Environment**:

* Python
* Libraries: scapy

**Implementation**:

* Code captures packets using scapy.sniff and displays packet summaries via a callback function.
* User can specify the number of packets or set a time-based limit.

**Key Features**:

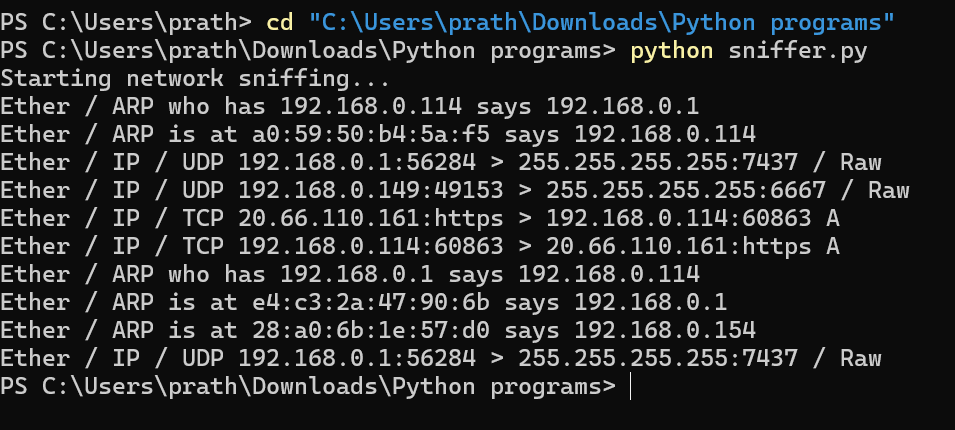
1. Captures real-time network packets.
2. Displays summary of packet details.

**Results**:

* Successfully captured and analyzed packets in a local environment.
* Demonstrated practical understanding of network traffic.

**Conclusion**: This project enhances the ability to work with network data and serves as a foundational step for understanding network security and traffic analysis.

Output:



**Explanation of Output**

1. **Packet Types**:
   * **Ether**: Indicates the packet is at the Ethernet layer.
   * **ARP**: Address Resolution Protocol, which resolves IP addresses to MAC addresses.
   * **IP**: Internet Protocol, for communication between devices.
   * **TCP**: Transmission Control Protocol, a reliable communication protocol.
   * **UDP**: User Datagram Protocol, a lightweight communication protocol.
2. **Fields in Each Packet**:
   * **ARP Example**:

bash

Ether / ARP who has 192.168.0.114 says 192.168.0.1

* + - This is an ARP request asking, "Who has the IP address 192.168.0.114?" The sender is 192.168.0.1.
    - Responses like ARP is at [MAC] says [IP] indicate a device replying with its MAC address.
  + **UDP Example**:

Ether / IP / UDP 192.168.0.1:56284 > 255.255.255.255:7437 / Raw

* + - A UDP packet sent from 192.168.0.1 (port 56284) to the broadcast address 255.255.255.255 (port 7437).
    - Raw means the payload is unprocessed by the sniffer.
  + **TCP Example**:

Ether / IP / TCP 20.66.110.161:https > 192.168.0.114:60863 A

* + - A TCP packet from 20.66.110.161 (using the HTTPS protocol) to 192.168.0.114 (on port 60863).
    - A indicates an acknowledgment (ACK) in the TCP handshake.

1. **What It Shows**:
   * **ARP Traffic**: Devices discovering each other on the local network.
   * **UDP Traffic**: Broadcast messages or lightweight communications.
   * **TCP Traffic**: Connections, likely for web browsing, file transfers, or secure communications (HTTPS).

**Analysis**

* Your sniffer is capturing both local network traffic (e.g., ARP requests) and traffic to external IPs (e.g., TCP connections to 20.66.110.161).
* You can identify:
  + **Local devices** by their IPs (192.168.x.x) and MAC addresses.
  + **Broadcast traffic** (255.255.255.255) used for discovery or announcements.

**Next Steps**

* **Dig Deeper**: Use packet.show() in your script for detailed packet content.
* **Filter Packets**:
  + To focus on specific protocols (e.g., TCP or HTTP), you can add filters:

sniff(prn=process\_packet, filter="tcp", count=10)

* **Analyze Traffic**:
  + Look up unknown IP addresses or MAC addresses to understand the devices involved.
  + Tools like [Wireshark](https://www.wireshark.org/) can provide more in-depth packet analysis.

Similar result of traffic analysis can be done using the tool Wireshark for understanding network security and traffic analysis.

