```
Python

# Python program to capture and analyze network traffic packets using
Scapy.
# This script demonstrates basic packet sniffing, analysis of
structure/content,
# and extraction of key information like source/destination IPs,
protocols, and payloads.
# It helps in understanding how data flows through the network and the
basics of protocols
# like IP, TCP, and UDP.

# Important Notes:
# - Scapy must be installed: pip install scapy
# - This script requires root/administrator privileges to capture packets.
#   Run with sudo on Linux/macOS: sudo python3 packet_sniffer.py
#   On Windows, run as Administrator.
# - Capturing packets may require appropriate network interface
configuration.
# - For educational purposes only; ensure compliance with laws and network
policies.
# - This example captures 10 packets; adjust 'count' as needed or remove
for continuous sniffing.
# - Press Ctrl+C to stop if count=0.

from scapy.all import sniff, IP, TCP, UDP  # Import necessary classes from
Scapy

def analyze_packet(pkt):
    """
    Callback function to analyze each captured packet.
    - Understands packet structure: Layers like Ethernet/IP/TCP/UDP/Raw.
    - Extracts and displays source/destination IPs, protocol, ports (if
applicable), and payload.
    - Explains data flow: Packets travel from source to destination via IP
addresses,
      using protocols like TCP (reliable, connection-oriented) or UDP
(fast, connectionless).
    """
    # Check if the packet has an IP layer (focus on IPv4 traffic)
    if IP in pkt:
        src_ip = pkt[IP].src  # Source IP address
        dst_ip = pkt[IP].dst  # Destination IP address
        proto = pkt[IP].proto  # Protocol number (e.g., 6 for TCP, 17 for
UDP)

        # Determine protocol name for readability
        proto_name = "TCP" if proto == 6 else "UDP" if proto == 17 else
f"Other ({proto})"

        # Extract ports if TCP or UDP
        src_port = None
        dst_port = None
        if TCP in pkt:
            src_port = pkt[TCP].sport
            dst_port = pkt[TCP].dport
        elif UDP in pkt:
            src_port = pkt[UDP].sport
            dst_port = pkt[UDP].dport

        # Extract payload (raw data after transport layer)
        payload = bytes(pkt.payload) if pkt.payload else b""
        payload_len = len(payload)

        # Display useful information
        print(f"\n--- Packet Captured ---")
        print(f"Source: {src_ip}:{src_port}" if src_port else f"Source:
{src_ip}")
        print(f"Destination: {dst_ip}:{dst_port}" if dst_port else
f"Destination: {dst_ip}")
        print(f"Protocol: {proto_name}")
        print(f"Payload Length: {payload_len} bytes")
        if payload_len > 0:
            # Display payload as hex for brevity (first 50 bytes)
            print(f"Payload (hex): {payload[:50].hex()}")

        # Educational insights
        print("\nStructure Insights:")
        print("- IP Layer: Handles routing between source and
destination.")
        print("- Transport Layer (e.g., TCP/UDP): Manages data delivery to
applications via ports.")
        print("- Payload: Actual data being transmitted (e.g., HTTP
request, file chunk).")
```

```
        print("- Data Flow: Packet originates at source IP, traverses network routers, arrives at destination IP.")

        # Full packet summary for deeper analysis
        print("\nPacket Summary:")
        print(pkt.summary())
    else:
        print("\n--- Non-IP Packet Captured ---")
        print(pkt.summary())

# Start sniffing packets
# Parameters:
# - iface: Specify interface if needed (e.g., "eth0"); default is any.
# - prn: Callback function for each packet.
# - count: Number of packets to capture (0 for infinite).
# - filter: BPF filter to capture specific traffic (e.g., "tcp" for TCP only).
print("Starting packet capture... Press Ctrl+C to stop.")
packets = sniff(prn=analyze_packet, count=10, filter="ip")  # Example: Capture 10 IP packets

# After capture, you can further analyze the list of packets if needed
print(f"\nCaptured {len(packets)} packets.")
```