BASIC NETWORK SNIFFER – CODE EXPLAINATION

- from scapy.all import sniff, IP, Ether, TCP, UDP, ICMP: This imports tools from the Scapy library to capture and analyze network packets for different protocols like IP, TCP, UDP, and ICMP.
- def packet_analyzer(packet) :This function examines a network packet and displays key information such as IP addresses, protocol, and other details.
- source_ip = packet[IP].src : Extracts the source IP address from the IP layer of the packet.
  destination_ip = packet[IP].dst : Extracts the destination IP address from the IP layer of the packet.
  protocol = packet[IP].pro: Retrieves the protocol number used in the IP layer (e.g., TCP, UDP, ICMP).
  ttl =packet[IP].ttl : Gets the time-to-live value, which indicates how long the packet can remain on the network.
- packet_length = len(packet) : Determines the total length of the packet in bytes.
- print(f"Source IP: {source_ip}, Destination IP: {destination_ip}")
  print(f"Protocol: {protocol}, TTL: {ttl}, Length: {packet_length}") : This code prints the source and destination IP addresses, the protocol type, TTL (time to live), and the length of the packet.
- if Ether in packet:
  source_mac = packet[Ether].src
  destination_mac = packet[Ether].dst
  print(f"Source MAC: {source_mac},
  Destination MAC: {destination_mac}") : This checks if the packet has an Ethernet layer, and if so, it extracts and prints the source and destination MAC addresses.
- if protocol == 6 and TCP in packet: # TCP
  tcp_sport = packet[TCP].sport
  tcp_dport = packet[TCP].dport
  tcp_seq = packet[TCP].seq
  tcp_flags = packet[TCP].flags
  print(f"TCP Source Port: {tcp_sport}, Destination Port: {tcp_dport}")
  print(f"Sequence: {tcp_seq}, Flags: {tcp_flags}") :

tcp_sport: TCP Source Port
tcp_dport: TCP Destination Port
tcp_seq: TCP Sequence Number
tcp_flags: TCP Flags

- elif protocol == 17 and UDP in packet: # UDP udp_sport = packet[UDP].sport udp_dport = packet[UDP].dport print(f"UDP Source Port: {udp_sport}, Destination Port: {udp_dport}"):
  udp_sport: UDP Source Port
  udp_dport: UDP Destination Port
- elif protocol == 1 and ICMP in packet: # ICMP icmp_type = packet[ICMP].type icmp_code = packet[ICMP].code print(f"ICMP Type: {icmp_type}, Code: {icmp_code}"):
  icmp_type: ICMP Message Type
  icmp_code: ICMP Message Code
- else: print("Other or Unknown Protocol") print("-" * 50) : If the packet doesn't match any known protocol (TCP, UDP, ICMP), this indicates it as an "Other or Unknown Protocol".
- sniff(prn=packet_analyzer, store=0) if __name__ == "__main__": main() :
  This line starts a packet sniffer that processes each captured packet using the packet_analyzer function.
  Main Check: Ensures the script is being run directly (not imported as a module). It calls the smart function to start capturing packets.

This script functions as a simple network sniffer, designed to capture and inspect packets traveling through the network. It processes and reveals detailed information about different network layers, including Ethernet, IP, TCP, UDP, and ICMP. By using this tool, you can analyze the structure and content of network traffic.