TASK 1

Basic Network Sniffer

Installation

First, you'll need to install `scapy`:

```
pip install scapy
```

Code

Here's a basic implementation of a network sniffer using `scapy`:

```
from scapy.all import *

def packet_callback(packet):
    if IP in packet:
        src_ip = packet[IP].src
        dst_ip = packet[IP].dst
        protocol = packet[IP].proto

        if protocol == 6 and TCP in packet:  # TCP
            src_port = packet[TCP].sport
            dst_port = packet[TCP].dport
            print(f'TCP Packet: {src_ip}:{src_port} -->
{dst_ip}:{dst_port}')

        elif protocol == 17 and UDP in packet:  # UDP
            src_port = packet[UDP].sport
            dst_port = packet[UDP].dport
            print(f'UDP Packet: {src_ip}:{src_port} -->
{dst_ip}:{dst_port}')

        elif protocol == 1 and ICMP in packet:  # ICMP
            icmp_type = packet[ICMP].type
            icmp_code = packet[ICMP].code
            print(f'ICMP Packet: {src_ip} --> {dst_ip} Type: {icmp_type}
Code: {icmp_code}')

def start_sniffing():
```

```
    sniff(prn=packet_callback, store=False)

if __name__ == '__main__':
    start_sniffing()
```

Explanation

1. **Imports**:
   - `from scapy.all import *`: Importing all functions and classes from `scapy`, which is used for constructing and dissecting network packets.
2. **packet_callback(packet)**:
   - This function is called for each packet captured by `sniff()`.
   - It checks if the packet is an IP packet (`IP in packet`).
   - Depending on the protocol (TCP, UDP, ICMP), it extracts and prints relevant information such as source IP, destination IP, source port, destination port, ICMP type, and ICMP code.
3. **start_sniffing()**:
   - This function starts capturing packets using `sniff()` from `scapy`.
   - `prn=packet_callback` specifies the callback function to be called for each captured packet.
   - `store=False` ensures that captured packets are not stored in memory to avoid running out of memory when capturing for long periods.
4. **Main section**:
   - `start_sniffing()` is called to begin capturing and analyzing network traffic.

Usage

- Run the script with root/administrator privileges to capture network traffic:

```
sudo python sniffer.py
```

Notes

- This is a basic example. Scapy allows for extensive packet manipulation and analysis, including crafting and sending packets.
- Ensure to run this script responsibly and in compliance with applicable laws and regulations.
- For more advanced features or specific protocol handling, you may need to extend the script further.