ASSIGNMENT-1

Write a program to sniff packet sent over the local network and analyse it.

YOUTUBE VIDEO LINK

https://www.youtube.com/watch?v=cWn5PSn6QDY

from scapy.all import sniff

from scapy.layers.inet import IP, TCP, UDP, ICMP

def packet\_callback(packet):

if IP in packet:

ip\_layer = packet[IP]

protocol = ip\_layer.proto

src\_ip = ip\_layer.src

dst\_ip = ip\_layer.dst

# Determine the protocol

protocol\_name = ""

if protocol == 1:

protocol\_name = "ICMP"

elif protocol == 6:

protocol\_name = "TCP"

elif protocol == 17:

protocol\_name = "UDP"

else:

protocol\_name = "Unknown Protocol"

# Print packet details

print(f"Protocol: {protocol\_name}")

print(f"Source IP: {src\_ip}")

print(f"Destination IP: {dst\_ip}")

print("-" \* 50)

def main():

# Capture packets on the default network interface

sniff(prn=packet\_callback, filter="ip", store=0)

if \_\_name\_\_ == "\_\_main\_\_":

main()

This Python script uses **Scapy** to capture and analyze network packets in real time. Let’s break it down:

**How It Works**

1. **Imports**:
   * scapy.all: Provides network packet manipulation functionalities.
   * scapy.layers.inet: Includes the IP, TCP, UDP, and ICMP protocol definitions.
2. **Packet Processing Function (packet\_callback)**:
   * Extracts the **IP layer** from each captured packet.
   * Determines the **protocol** (ICMP, TCP, UDP, or unknown).
   * Extracts **source and destination IP addresses**.
   * Prints the packet details.
3. **Packet Sniffing (sniff)**:
   * Captures packets with the "ip" filter (i.e., only IPv4 packets).
   * Calls packet\_callback for each packet.
   * Does not store packets (store=0 to save memory).
4. **Execution (main)**:
   * Runs the packet sniffer on the default network interface.

**Enhancements & Security Considerations**

✅ **Protocol Filtering**: You can modify the filter, e.g., filter="tcp" for TCP-only packets.  
✅ **Interface Selection**: Specify iface="eth0" (Linux) or iface="Wi-Fi" (Windows) for a specific network adapter.  
✅ **Permission Requirements**:

* **Linux**: Needs sudo privileges to capture packets.

The output of this code depends on the network traffic at the time of execution. If the script is running on an active network, it will capture and display details of IPv4 packets. Here’s an example of what the output might look like:

**Example Output**

Protocol: TCP

Source IP: 192.168.1.100

Destination IP: 172.217.160.78

--------------------------------------------------

Protocol: UDP

Source IP: 192.168.1.50

Destination IP: 8.8.8.8

--------------------------------------------------

Protocol: ICMP

Source IP: 192.168.1.101

Destination IP: 192.168.1.1

--------------------------------------------------

Protocol: TCP

Source IP: 10.0.0.5

Destination IP: 93.184.216.34

--------------------------------------------------

**What This Means**

* **TCP packet**: A computer (192.168.1.100) is communicating with a remote server (172.217.160.78, possibly Google).
* **UDP packet**: A device (192.168.1.50) is sending a request to Google's DNS server (8.8.8.8).
* **ICMP packet**: A ping request is being sent from 192.168.1.101 to the router (192.168.1.1).

**Notes**

* The script continuously captures packets until manually stopped (Ctrl + C).
* If there is no network activity, the script may not display any output.
* Running the script requires administrator/root privileges (sudo in Linux).

Running this code on an **online compiler** is challenging because:

1. **Most online compilers lack root/admin privileges**, which are required to capture packets.
2. **Scapy requires direct access to the network interface**, which online environments typically restrict.
3. **Online compilers run inside a sandbox**, preventing real-time packet sniffing.

**Alternative Solutions**

**1. Run the Code on Your Local Machine**

You can run this script on your own computer with **Python** installed.

**Steps for Windows**:

1. Install **Scapy**:

pip install scapy

1. Open **Command Prompt as Administrator**.
2. Run the script:

python filename.py

**Steps for Linux/macOS**:

1. Install **Scapy**:

pip install scapy

1. Run the script with **root privileges**:

sudo python filename.py

**2. Use Google Colab (Limited Functionality)**

Google Colab does not support direct packet sniffing, but you can install Scapy and simulate packets:

!pip install scapy

from scapy.all import \*

packet = IP(dst="8.8.8.8")/ICMP()

packet.show()

This will **create a packet** but **not sniff real traffic**.

**3. Use a Cloud-Based Virtual Machine**

If you need online execution, consider using a **cloud VM** (e.g., AWS, Google Cloud, DigitalOcean).

1. **Set up an Ubuntu VM**.
2. Install Python and Scapy:

sudo apt update && sudo apt install python3-pip -y

pip install scapy

1. Run the script with sudo:

sudo python3 filename.py

**Conclusion**

* **Online compilers** won’t work due to security restrictions.
* **Use a local machine** with admin privileges.
* **A cloud VM is an alternative** if you need remote execution.

**You said:**

how to run this code on visualstudio

To run this **packet sniffing** script in **Visual Studio Code (VS Code)**, follow these steps:

**Step 1: Install Python & Scapy**

**For Windows**

1. Install **Python** (if not installed): [Download Python](https://www.python.org/downloads/)
2. Open **Command Prompt as Administrator** and install Scapy:

pip install scapy

1. Install **WinPcap or Npcap** (required for packet sniffing):
   * Download & install **Npcap**: Download Npcap
   * Ensure you check **“Install Npcap in WinPcap API-compatible mode”** during installation.

**For Linux/macOS**

1. Install Python and Scapy:

sudo apt update && sudo apt install python3-pip -y # Ubuntu/Debian

pip install scapy

1. Run the script with **sudo** to grant network access:

sudo python3 filename.py

**Step 2: Open the Script in VS Code**

1. Open **VS Code**.
2. Install the **Python extension** from the Extensions Marketplace.
3. Create a new file, e.g., **sniffer.py**.
4. Copy and paste your **Scapy script** into the file.

**Step 3: Run the Script**

1. Open **VS Code Terminal** (Ctrl + ~ on Windows/Linux, Cmd + ~ on Mac).
2. **For Windows** (Run as Administrator):

python sniffer.py

1. **For Linux/macOS** (Requires sudo):

sudo python3 sniffer.py

**Step 4: Stop the Sniffing**

The script runs indefinitely. **Press Ctrl + C** to stop it.

**Expected Output**

If packets are captured, the terminal will display:

Protocol: TCP

Source IP: 192.168.1.10

Destination IP: 142.250.180.78

--------------------------------------------------

Protocol: UDP

Source IP: 192.168.1.20

Destination IP: 8.8.8.8

--------------------------------------------------

Protocol: ICMP

Source IP: 192.168.1.5

Destination IP: 192.168.1.1

--------------------------------------------------

**Troubleshooting**

**1. "Permission Denied" Error (Linux/macOS)**

* Run with **sudo**:

sudo python3 sniffer.py

**2. "No Module Named Scapy" Error**

* Install Scapy again:

pip install scapy

**3. No Packets Captured**

* Ensure **Npcap** is installed (WinPcap won't work on modern Windows).
* Run **VS Code as Administrator** (Windows).
* Use the **correct network interface**:

sniff(iface="Wi-Fi", prn=packet\_callback, filter="ip", store=0)