CodeAlpha Internship

Task 1: Basic Network Sniffer - Project Report

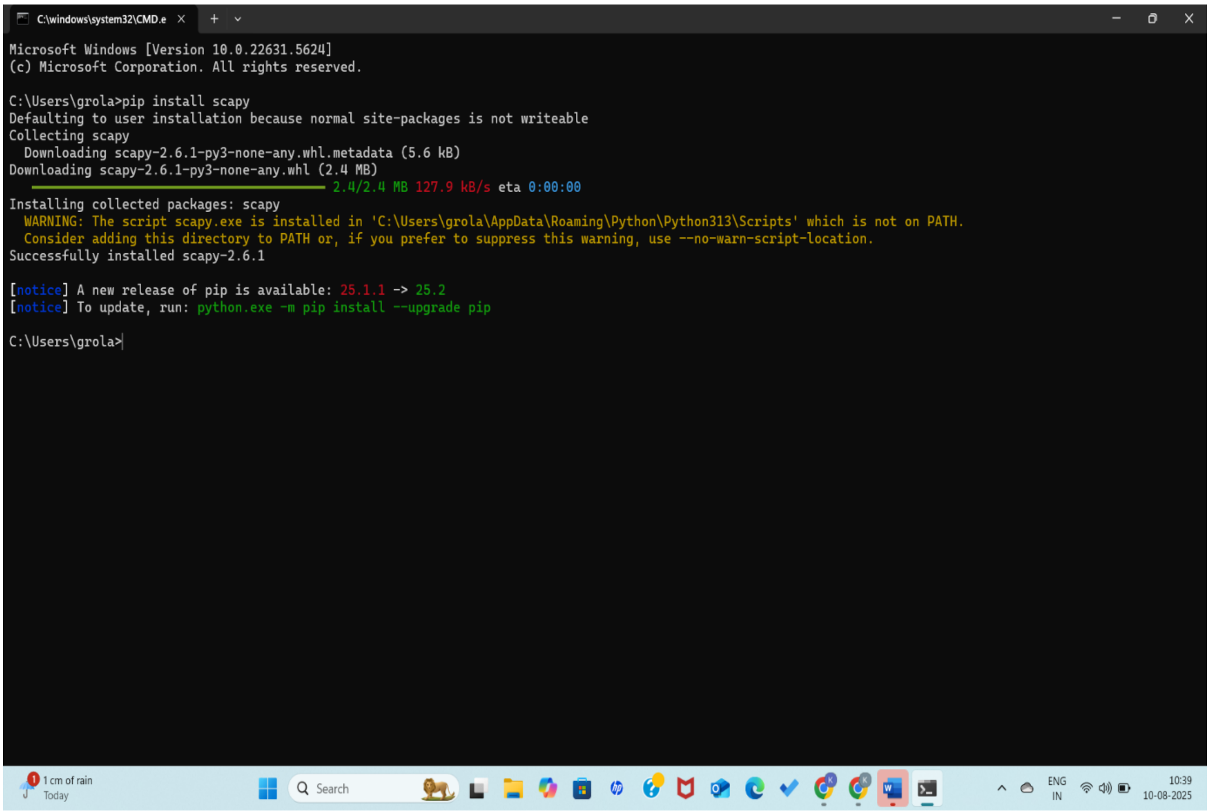
Overview: This project is part of the CodeAlpha Cyber Security Internship. The Basic Network Sniffer is a Python-based tool that captures and analyzes network packets in real-time using the Scapy library. It helps understand network protocols, data flows, and packet structures, providing insights into TCP, UDP, and ICMP traffic.

Features: Captures network packets in real time- Displays Source IP, Destination IP, Protocol, and Payload- Supports TCP, UDP, and ICMP- Optional filtering by protocol- Option to save captured packets to .pcap file

Requirements:

-Python 3.10+

- Scapy library (`pip install scapy`)



```
Microsoft Windows [Version 10.0.22631.5624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\grola>pip install scapy
Defaulting to user installation because normal site-packages is not writeable
Collecting scapy
  Downloading scapy-2.6.1-py3-none-any.whl.metadata (5.6 kB)
Downloading scapy-2.6.1-py3-none-any.whl (2.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.4/2.4 MB 127.9 kB/s eta 0:00:00
Installing collected packages: scapy
  WARNING: The script scapy.exe is installed in 'C:\Users\grola\AppData\Roaming\Python\Python313\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed scapy-2.6.1

[notice] A new release of pip is available: 25.1.1 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\grola>
```

- Administrator/root privileges for packet capture

- Works on Windows, Linux, and Mac OS

Python Code:

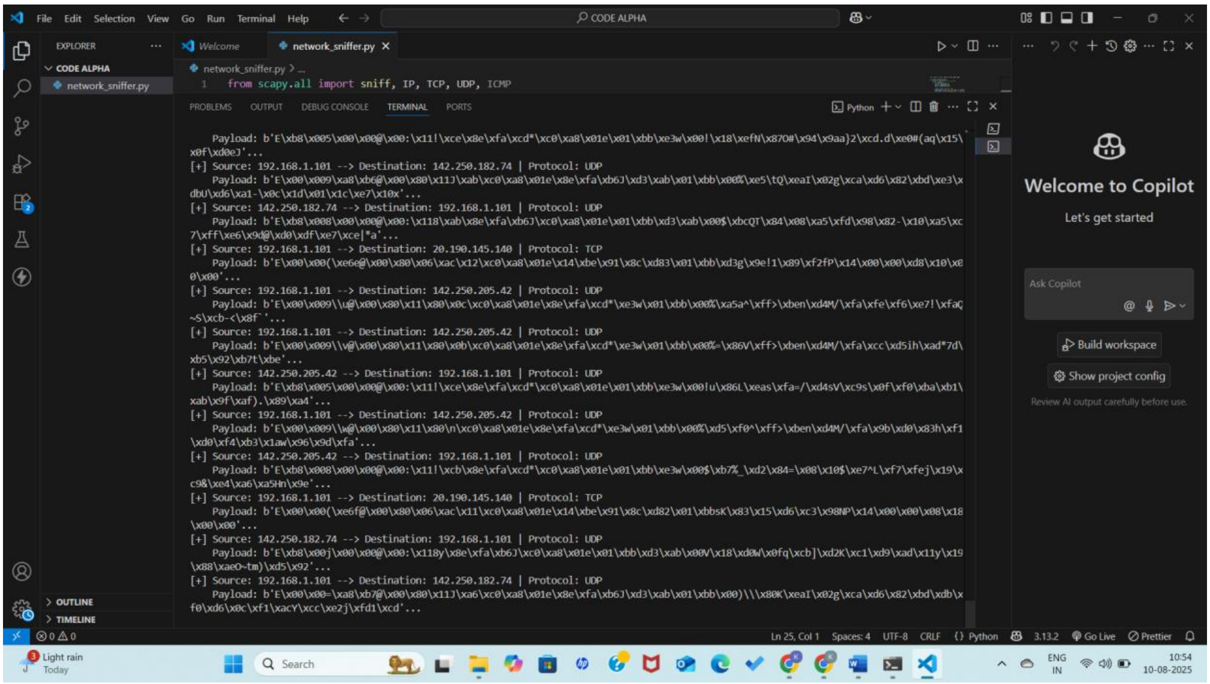
```
from scapy.all import sniff, IP, TCP, UDP, ICMP
def packet_callback(packet):
    # Check if the packet has an IP layer
    if IP in packet:
        src_ip = packet[IP].src
        dst_ip = packet[IP].dst
        proto = packet[IP].proto

        # Map protocol number to name
        protocol_map = {6: "TCP", 17: "UDP", 1: "ICMP"}
        proto_name = protocol_map.get(proto, str(proto))

        print(f"[+] Source: {src_ip} --> Destination: {dst_ip} | Protocol: {proto_name}")

        # Try to extract payload if available
        if packet.haslayer(TCP) or packet.haslayer(UDP):
            payload = bytes(packet.payload)
            if payload:
                print(f"    Payload: {payload[:50]}...")  # Show first 50 bytes

# Capture packets (adjust count or timeout as needed)
print("Starting packet capture... Press CTRL+C to stop.")
sniff(prn=packet_callback, store=False)
```

Usage:

1. Install dependencies: pip install scapy

2. Run the program as administrator/root: Linux/Mac: sudo python

3 network_sniffer.py Windows: python network_sniffer.py 3. Generate network activity (e.g., ping google.com or open a website) to see packet logs



```
from scapy.all import sniff, IP, TCP, UDP, ICMP
```

```
    Payload: b'E\xb8\x005\x00\x00@\x00:\x11!\xce\x8e\xfa\xcd*\xc0\xa8\x01e\x01\xbb\xe3w\x00!\x18\xefN\x870#\x94\x9aa}2\xcd.d\xe0#(aq\x15\
x0f\xd0eJ'...
[+] Source: 192.168.1.101 --> Destination: 142.250.182.74 | Protocol: UDP
    Payload: b'E\x00\x009\xa8\xb6@\x00\x80\x11J\xab\xc0\xa8\x01e\x8e\xfa\xb6J\xd3\xab\x01\xbb\x00%\xe5\tQ\xeaI\x02g\xca\xd6\x82\xbd\xe3\x
dbU\xd6\xa1-\x0c\x1d\x01\x1c\xe7\x10x'...
[+] Source: 142.250.182.74 --> Destination: 192.168.1.101 | Protocol: UDP
    Payload: b'E\xb8\x008\x00\x00@\x00:\x118\xab\x8e\xfa\xb6J\xc0\xa8\x01e\x01\xbb\xd3\xab\x00$\xbcQT\x84\x08\xa5\xfd\x98\x82-\x10\xa5\xc
7\xff\xe6\x9d@\xd0\xdf\xe7\xce|*a'...
[+] Source: 192.168.1.101 --> Destination: 20.190.145.140 | Protocol: TCP
    Payload: b'E\x00\x00(\xe6e@\x00\x80\x06\xac\x12\xc0\xa8\x01e\x14\xbe\x91\x8c\xd83\x01\xbb\xd3g\x9e!1\x89\xf2fP\x14\x00\x00\xd8\x10\x0
0\x00'...
[+] Source: 192.168.1.101 --> Destination: 142.250.205.42 | Protocol: UDP
    Payload: b'E\x00\x009\\u@\x00\x80\x11\x80\x0c\xc0\xa8\x01e\x8e\xfa\xcd*\xe3w\x01\xbb\x00%\xa5a^\xff>\xben\xd4M/\xfa\xfe\xf6\xe7!\xfaQ
~S\xcb-<\x8f`'...
[+] Source: 192.168.1.101 --> Destination: 142.250.205.42 | Protocol: UDP
    Payload: b'E\x00\x009\\v@\x00\x80\x11\x80\x0b\xc0\xa8\x01e\x8e\xfa\xcd*\xe3w\x01\xbb\x00%=\x86V\xff>\xben\xd4M/\xfa\xcc\xd5ih\xad*7d\
xb5\x92\xb7t\xbe'...
[+] Source: 142.250.205.42 --> Destination: 192.168.1.101 | Protocol: UDP
    Payload: b'E\xb8\x005\x00\x00@\x00:\x11!\xce\x8e\xfa\xcd*\xc0\xa8\x01e\x01\xbb\xe3w\x00!u\x86L\xeas\xfa=/\xd4sV\xc9s\x0f\xf0\xba\xb1\
xab\x9f\xaf).\x89\xa4'...
[+] Source: 192.168.1.101 --> Destination: 142.250.205.42 | Protocol: UDP
    Payload: b'E\x00\x009\\w@\x00\x80\x11\x80\n\xc0\xa8\x01e\x8e\xfa\xcd*\xe3w\x01\xbb\x00%\xd5\xf0^\xff>\xben\xd4M/\xfa\x9b\xd0\x83h\xf1
\xd0\xf4\xb3\x1aw\x96\x9d\xfa'...
[+] Source: 142.250.205.42 --> Destination: 192.168.1.101 | Protocol: UDP
    Payload: b'E\xb8\x008\x00\x00@\x00:\x11!\xcb\x8e\xfa\xcd*\xc0\xa8\x01e\x01\xbb\xe3w\x00$\xb7%_\xd2\x84=\x08\x10$\xe7^L\xf7\xfej\x19\x
c9&\xe4\xa6\xa5Hn\x9e'...
[+] Source: 192.168.1.101 --> Destination: 20.190.145.140 | Protocol: TCP
    Payload: b'E\x00\x00(\xe6f@\x00\x80\x06\xac\x11\xc0\xa8\x01e\x14\xbe\x91\x8c\xd82\x01\xbbsK\x83\x15\xd6\xc3\x98NP\x14\x00\x00\x08\x18
\x00\x00'...
[+] Source: 142.250.182.74 --> Destination: 192.168.1.101 | Protocol: UDP
    Payload: b'E\xb8\x00j\x00\x00@\x00:\x118y\x8e\xfa\xb6J\xc0\xa8\x01e\x01\xbb\xd3\xab\x00V\x18\xd0W\x0fq\xcb]\xd2K\xc1\xd9\xad\x11y\x19
\x88\xaeO~tm)\xd5\x92'...
[+] Source: 192.168.1.101 --> Destination: 142.250.182.74 | Protocol: UDP
    Payload: b'E\x00\x00=\xa8\xb7@\x00\x80\x11J\xa6\xc0\xa8\x01e\x8e\xfa\xb6J\xd3\xab\x01\xbb\x00)\\\x80K\xeaI\x02g\xca\xd6\x82\xbd\xdb\x
f0\xd6\x0c\xf1\xacY\xcc\xe2j\xfd1\xcd'...
```

Testing & Validation:-

Verified packet capture with ping and browser traffic- Observed real-time IP, protocol, and payload data- Ensured correct mapping of protocol numbers to names- Validated script on both Linux and Windows

Conclusion:

The Basic Network Sniffer provides a hands-on introduction to network packet analysis. It serves as a foundation for more advanced network monitoring tools such as intrusion detection systems. This project enhances practical understanding of networking and cybersecurity concepts

Task 4: Intrusion Detection System (IDS) using Snort on Windows

This document provides a step-by-step guide for installing and running Snort as an Intrusion Detection System (IDS) on Windows. It includes the installation, configuration, and testing of a simple ICMP Ping Detection rule. The setup assumes Snort is installed on the E: drive.

Step 1:

Install Snort 1. Download Snort 2.9.20 WIN64 from snort.org.

2. Extract it to E:\Snort.

3. Add E:\Snort\bin to your PATH environment variable.

Step 2:

Create Local Rule - Create or open E:\Snort\rules\local.rules

- Add the following rule:

alert icmp any any -> $HOME_NET any (msg:"ICMP Ping Detected"; sid:1000001; rev:1;)

Step 3:

Configure snort.conf - Open E:\Snort\etc\snort.conf - Update dynamic module paths to point to E:\Snort\lib\... - Ensure $RULE_PATH/local.rules is included. - Set HOME_NET to your IP range or 'any'.

Step 4:

Find Network Interface - Run: snort -W - Note the index number for your active Wi-Fi/Ethernet adapter.

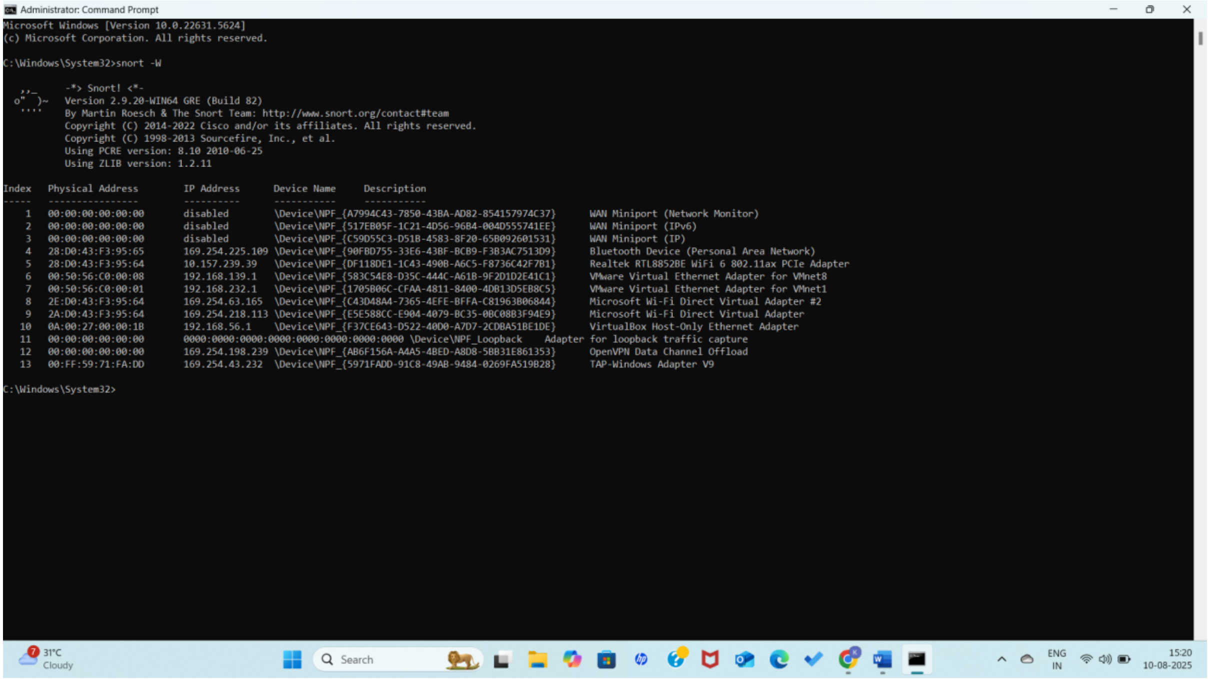
Step 5:

Run Snort in IDS Mode Example (if interface number is 5): snort -i 5 -A console -c E:\Snort\etc\snort.conf

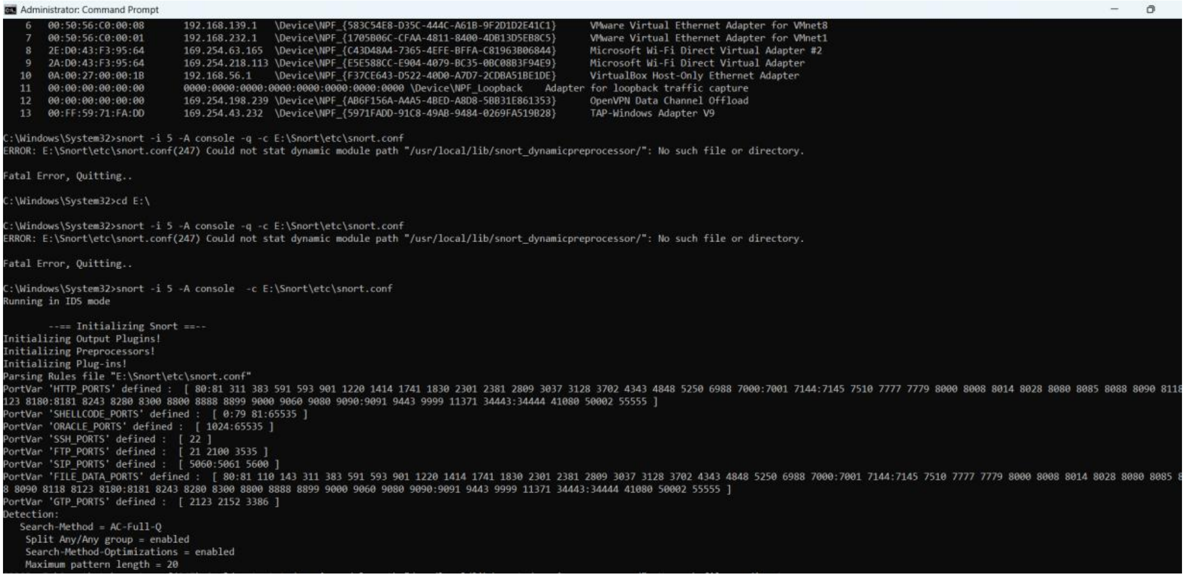
Step 6:

Test IDS - From another device on the same network: ping - Expected alert: [**] [1:1000001:1] ICMP Ping Detected [**]

Network Interfaces Output:



```
Microsoft Windows [Version 10.0.22631.5624]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>snort -W

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.20-WIN64 GRE (Build 82)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.11

Index   Physical Address        IP Address      Device Name     Description
-----   -----------------       ----------      -----------     -----------
    1   00:00:00:00:00:00       disabled        \Device\NPF_{A7994C43-7850-43BA-AD82-854157974C37}      WAN Miniport (Network Monitor)
    2   00:00:00:00:00:00       disabled        \Device\NPF_{517EB05F-1C21-4D56-96B4-004D555741EE}      WAN Miniport (IPv6)
    3   00:00:00:00:00:00       disabled        \Device\NPF_{C59D55C3-D51B-4583-8F20-65B092601531}      WAN Miniport (IP)
    4   28:D0:43:F3:95:65       169.254.225.109 \Device\NPF_{90FBD755-33E6-43BF-BCB9-F3B3AC7513D9}      Bluetooth Device (Personal Area Network)
    5   28:D0:43:F3:95:64       10.157.239.39   \Device\NPF_{DF118DE1-1C43-490B-A6C5-F8736C42F7B1}      Realtek RTL8852BE WiFi 6 802.11ax PCIe Adapter
    6   00:50:56:C0:00:08       192.168.139.1   \Device\NPF_{583C54E8-D35C-444C-A61B-9F2D1D2E41C1}      VMware Virtual Ethernet Adapter for VMnet8
    7   00:50:56:C0:00:01       192.168.232.1   \Device\NPF_{1705B06C-CFAA-4811-8400-4DB13D5EB8C5}      VMware Virtual Ethernet Adapter for VMnet1
    8   2E:D0:43:F3:95:64       169.254.63.165  \Device\NPF_{C43D48A4-7365-4EFE-BFFA-C81963B06844}      Microsoft Wi-Fi Direct Virtual Adapter #2
    9   2A:D0:43:F3:95:64       169.254.218.113 \Device\NPF_{E5E588CC-E904-4079-BC35-0BC08B3F94E9}      Microsoft Wi-Fi Direct Virtual Adapter
   10   0A:00:27:00:00:1B       192.168.56.1    \Device\NPF_{F37CE643-D522-40D0-A7D7-2CDBA51BE1DE}      VirtualBox Host-Only Ethernet Adapter
   11   00:00:00:00:00:00       0000:0000:0000:0000:0000:0000:0000:0000 \Device\NPF_Loopback    Adapter for loopback traffic capture
   12   00:00:00:00:00:00       169.254.198.239 \Device\NPF_{AB6F156A-A4A5-4BED-A8D8-5BB31E861353}      OpenVPN Data Channel Offload
   13   00:FF:59:71:FA:DD       169.254.43.232  \Device\NPF_{5971FADD-91C8-49AB-9484-0269FA519B28}      TAP-Windows Adapter V9

C:\Windows\System32>
```



```
    6   00:50:56:C0:00:08       192.168.139.1   \Device\NPF_{583C54E8-D35C-444C-A61B-9F2D1D2E41C1}      VMware Virtual Ethernet Adapter for VMnet8
    7   00:50:56:C0:00:01       192.168.232.1   \Device\NPF_{1705B06C-CFAA-4811-8400-4DB13D5EB8C5}      VMware Virtual Ethernet Adapter for VMnet1
    8   2E:D0:43:F3:95:64       169.254.63.165  \Device\NPF_{C43D48A4-7365-4EFE-BFFA-C81963B06844}      Microsoft Wi-Fi Direct Virtual Adapter #2
    9   2A:D0:43:F3:95:64       169.254.218.113 \Device\NPF_{E5E588CC-E904-4079-BC35-0BC08B3F94E9}      Microsoft Wi-Fi Direct Virtual Adapter
   10   0A:00:27:00:00:1B       192.168.56.1    \Device\NPF_{F37CE643-D522-40D0-A7D7-2CDBA51BE1DE}      VirtualBox Host-Only Ethernet Adapter
   11   00:00:00:00:00:00       0000:0000:0000:0000:0000:0000:0000:0000 \Device\NPF_Loopback    Adapter for loopback traffic capture
   12   00:00:00:00:00:00       169.254.198.239 \Device\NPF_{AB6F156A-A4A5-4BED-A8D8-5BB31E861353}      OpenVPN Data Channel Offload
   13   00:FF:59:71:FA:DD       169.254.43.232  \Device\NPF_{5971FADD-91C8-49AB-9484-0269FA519B28}      TAP-Windows Adapter V9

C:\Windows\System32>snort -i 5 -A console -q -c E:\Snort\etc\snort.conf
ERROR: E:\Snort\etc\snort.conf(247) Could not stat dynamic module path "/usr/local/lib/snort_dynamicpreprocessor/": No such file or directory.

Fatal Error, Quitting..

C:\Windows\System32>cd E:\

C:\Windows\System32>snort -i 5 -A console -q -c E:\Snort\etc\snort.conf
ERROR: E:\Snort\etc\snort.conf(247) Could not stat dynamic module path "/usr/local/lib/snort_dynamicpreprocessor/": No such file or directory.

Fatal Error, Quitting..

C:\Windows\System32>snort -i 5 -A console  -c E:\Snort\etc\snort.conf
Running in IDS mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "E:\Snort\etc\snort.conf"
PortVar 'HTTP_PORTS' defined :  [ 80:81 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 7144:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118
123 8180:8181 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444 41080 50002 55555 ]
PortVar 'SHELLCODE_PORTS' defined :  [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined :  [ 1024:65535 ]
PortVar 'SSH_PORTS' defined :  [ 22 ]
PortVar 'FTP_PORTS' defined :  [ 21 2100 3535 ]
PortVar 'SIP_PORTS' defined :  [ 5060:5061 5600 ]
PortVar 'FILE_DATA_PORTS' defined :  [ 80:81 110 143 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 7144:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8
8 8090 8118 8123 8180:8181 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444 41080 50002 55555 ]
PortVar 'GTP_PORTS' defined :  [ 2123 2152 3386 ]
Detection:
   Search-Method = AC-Full-Q
    Split Any/Any group = enabled
    Search-Method-Optimizations = enabled
    Maximum pattern length = 20
```

LOVAKUMARI GORLA

ISTS WOMEN’S ENGINEERING COLLEGE,RAJANAGARAM

Kumarigorla09@gmail.com