CodeAlpha Internship Task 1 Report

Domain: Cyber Security

Task/Objective:

Build a network sniffer in Python that captures and analyzes network traffic. This project will help you understand how data flows on a network and how network packets are structure.

Description:

A network sniffer is a tool used to capture and analyze network traffic. By building a network sniffer in Python, you will gain insights into the structure of network packets, the protocols used, and the flow of data across a network. This project is ideal for anyone looking to enhance their knowledge of network security, protocol analysis, and packet inspection.

Key Components:

a. Basic packet capture.
b. Packet analysis.
c. Data visualization.
d. Filtering and searching.
e. Logging and reporting.

Expected Outcomes:

- A functional network sniffer that can capture and analyze network traffic.
- Enhanced understanding of network protocols and packet structures.
- Skills in using Python for network analysis and packet inspection.
- Experience in developing tools for network security and monitoring.

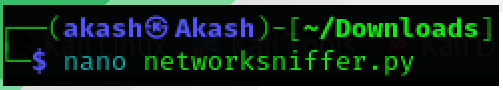
Tools and Libraries:

- **Scapy:** A powerful Python library for network packet manipulation.

Steps to complete the task:

----

1. Create networksniffer.py



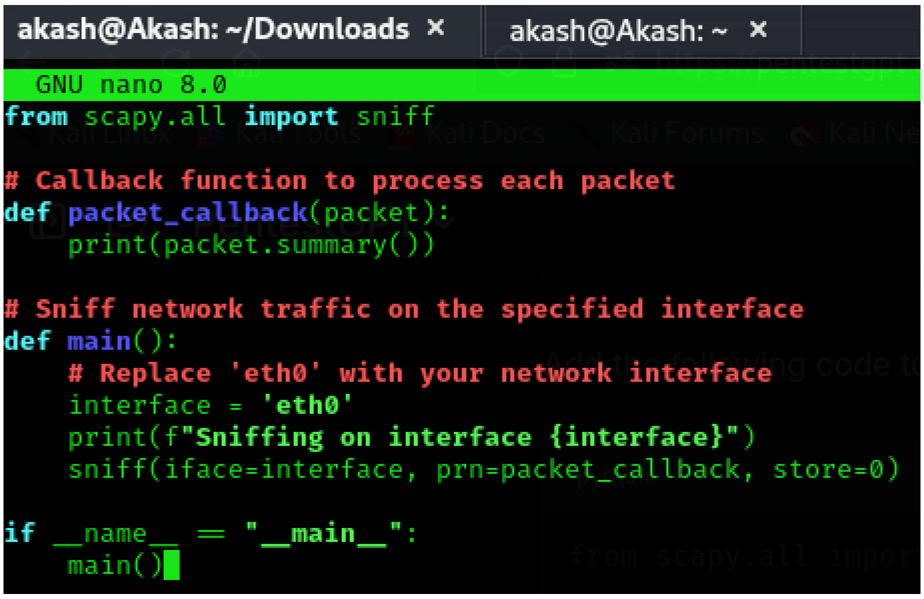
```
┌──(akash㉿Akash)-[~/Downloads]
└─$ nano networksniffer.py
```

2. I Inserted basic script of network sniffer.
   Script :-
   from scapy.all import sniff

   # Callback function to process each packet
   def packet_callback(packet):
       print(packet.summary())

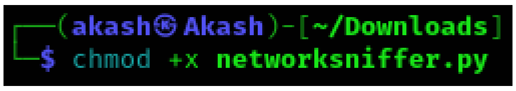
   # Sniff network traffic on the specified interface
   def main():
       # Replace 'eth0' with your network interface
       interface = 'eth0'
       print(f"Sniffing on interface {interface}")
       sniff(iface=interface, prn=packet_callback, store=0)

   if __name__ == "__main__":
       main()



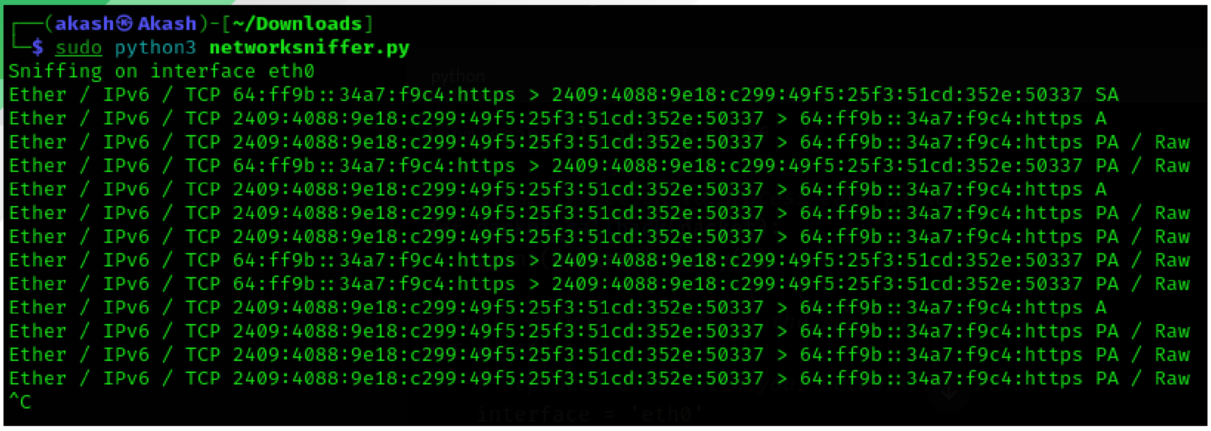
```
  GNU nano 8.0
from scapy.all import sniff

# Callback function to process each packet
def packet_callback(packet):
    print(packet.summary())

# Sniff network traffic on the specified interface
def main():
    # Replace 'eth0' with your network interface
    interface = 'eth0'
    print(f"Sniffing on interface {interface}")
    sniff(iface=interface, prn=packet_callback, store=0)

if __name__ == "__main__":
    main()
```

3. I saved the file and give executable permissions.



```
┌──(akash㉿Akash)-[~/Downloads]
└─$ chmod +x networksniffer.py
```

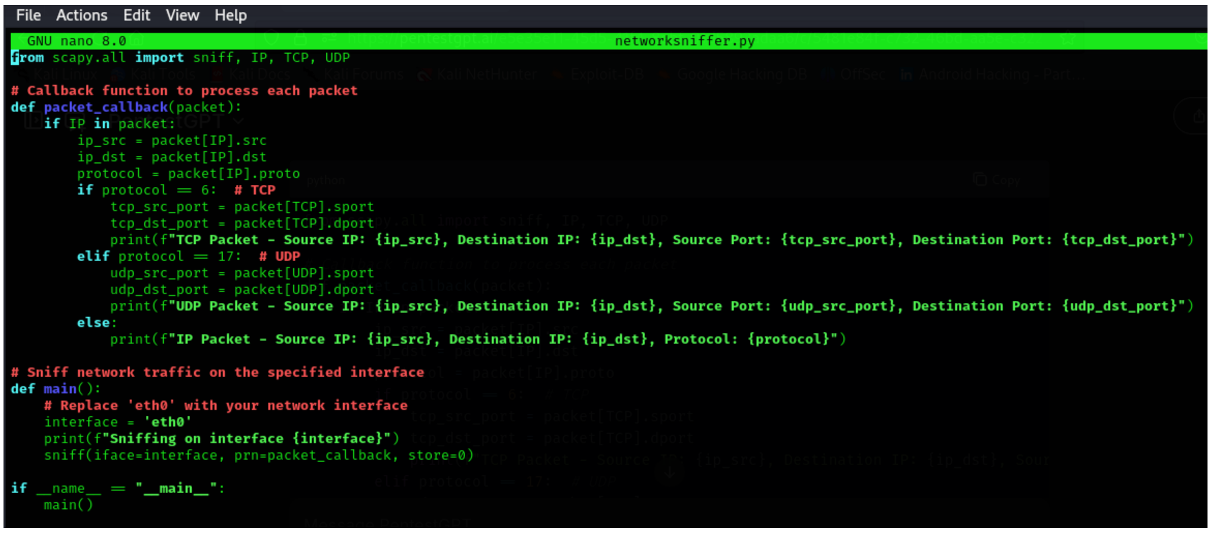
4. I run the sniffer.



```
┌──(akash㉿Akash)-[~/Downloads]
└─$ sudo python3 networksniffer.py
Sniffing on interface eth0
Ether / IPv6 / TCP 64:ff9b::34a7:f9c4:https > 2409:4088:9e18:c299:49f5:25f3:51cd:352e:50337 SA
Ether / IPv6 / TCP 2409:4088:9e18:c299:49f5:25f3:51cd:352e:50337 > 64:ff9b::34a7:f9c4:https A
Ether / IPv6 / TCP 2409:4088:9e18:c299:49f5:25f3:51cd:352e:50337 > 64:ff9b::34a7:f9c4:https PA / Raw
Ether / IPv6 / TCP 64:ff9b::34a7:f9c4:https > 2409:4088:9e18:c299:49f5:25f3:51cd:352e:50337 PA / Raw
Ether / IPv6 / TCP 2409:4088:9e18:c299:49f5:25f3:51cd:352e:50337 > 64:ff9b::34a7:f9c4:https A
Ether / IPv6 / TCP 2409:4088:9e18:c299:49f5:25f3:51cd:352e:50337 > 64:ff9b::34a7:f9c4:https PA / Raw
Ether / IPv6 / TCP 2409:4088:9e18:c299:49f5:25f3:51cd:352e:50337 > 64:ff9b::34a7:f9c4:https PA / Raw
Ether / IPv6 / TCP 64:ff9b::34a7:f9c4:https > 2409:4088:9e18:c299:49f5:25f3:51cd:352e:50337 PA / Raw
Ether / IPv6 / TCP 64:ff9b::34a7:f9c4:https > 2409:4088:9e18:c299:49f5:25f3:51cd:352e:50337 PA / Raw
Ether / IPv6 / TCP 2409:4088:9e18:c299:49f5:25f3:51cd:352e:50337 > 64:ff9b::34a7:f9c4:https A
Ether / IPv6 / TCP 2409:4088:9e18:c299:49f5:25f3:51cd:352e:50337 > 64:ff9b::34a7:f9c4:https PA / Raw
Ether / IPv6 / TCP 2409:4088:9e18:c299:49f5:25f3:51cd:352e:50337 > 64:ff9b::34a7:f9c4:https PA / Raw
Ether / IPv6 / TCP 2409:4088:9e18:c299:49f5:25f3:51cd:352e:50337 > 64:ff9b::34a7:f9c4:https PA / Raw
^C
```

5. To analyze packets I further modified the sniffer.
   Update the script to include more detailed packet analysis:

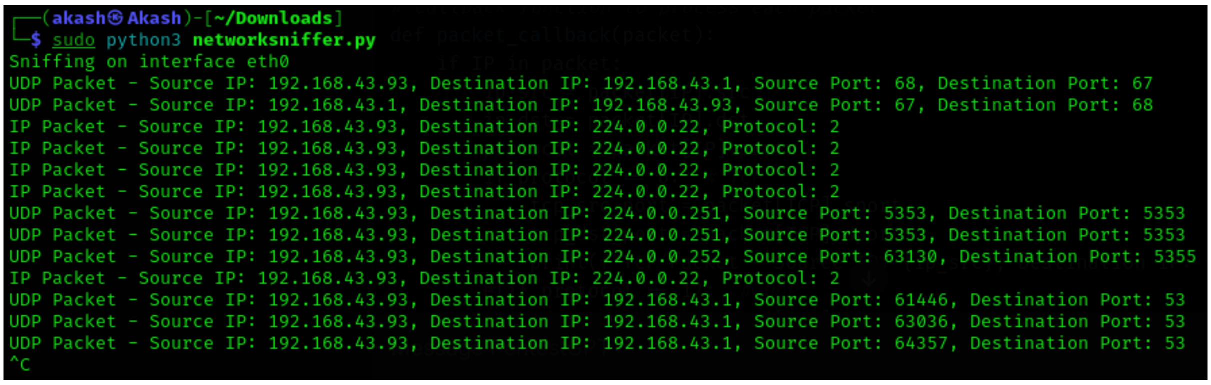
```
from scapy.all import sniff, IP, TCP, UDP

# Callback function to process each packet
def packet_callback(packet):
    if IP in packet:
        ip_src = packet[IP].src
        ip_dst = packet[IP].dst
        protocol = packet[IP].proto
        if protocol == 6:  # TCP
            tcp_src_port = packet[TCP].sport
            tcp_dst_port = packet[TCP].dport
            print(f"TCP Packet - Source IP: {ip_src}, Destination IP: 
{ip_dst}, Source Port: {tcp_src_port}, Destination Port: {tcp_dst_port}")
        elif protocol == 17:  # UDP
            udp_src_port = packet[UDP].sport
            udp_dst_port = packet[UDP].dport
            print(f"UDP Packet - Source IP: {ip_src}, Destination IP: 
{ip_dst}, Source Port: {udp_src_port}, Destination Port: 
{udp_dst_port}")
        else:
            print(f"IP Packet - Source IP: {ip_src}, Destination IP: {ip_dst}, 
Protocol: {protocol}")

# Sniff network traffic on the specified interface
def main():
    # Replace 'eth0' with your network interface
    interface = 'eth0'
    print(f"Sniffing on interface {interface}")
    sniff(iface=interface, prn=packet_callback, store=0)

if __name__ == "__main__":
    main()
```



File Actions Edit View Help

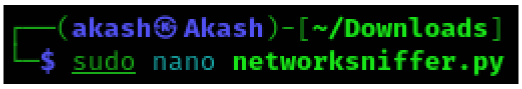
```
GNU nano 8.0                                networksniffer.py
from scapy.all import sniff, IP, TCP, UDP

# Callback function to process each packet
def packet_callback(packet):
    if IP in packet:
        ip_src = packet[IP].src
        ip_dst = packet[IP].dst
        protocol = packet[IP].proto
        if protocol == 6:  # TCP
            tcp_src_port = packet[TCP].sport
            tcp_dst_port = packet[TCP].dport
            print(f"TCP Packet - Source IP: {ip_src}, Destination IP: {ip_dst}, Source Port: {tcp_src_port}, Destination Port: {tcp_dst_port}")
        elif protocol == 17:  # UDP
            udp_src_port = packet[UDP].sport
            udp_dst_port = packet[UDP].dport
            print(f"UDP Packet - Source IP: {ip_src}, Destination IP: {ip_dst}, Source Port: {udp_src_port}, Destination Port: {udp_dst_port}")
        else:
            print(f"IP Packet - Source IP: {ip_src}, Destination IP: {ip_dst}, Protocol: {protocol}")

# Sniff network traffic on the specified interface
def main():
    # Replace 'eth0' with your network interface
    interface = 'eth0'
    print(f"Sniffing on interface {interface}")
    sniff(iface=interface, prn=packet_callback, store=0)

if __name__ == "__main__":
    main()
```

6. I run the sniffer again.
   The output shows visible IPs in the packets.



```
┌──(akash㉿Akash)-[~/Downloads]
└─$ sudo python3 networksniffer.py
Sniffing on interface eth0
UDP Packet - Source IP: 192.168.43.93, Destination IP: 192.168.43.1, Source Port: 68, Destination Port: 67
UDP Packet - Source IP: 192.168.43.1, Destination IP: 192.168.43.93, Source Port: 67, Destination Port: 68
IP Packet - Source IP: 192.168.43.93, Destination IP: 224.0.0.22, Protocol: 2
IP Packet - Source IP: 192.168.43.93, Destination IP: 224.0.0.22, Protocol: 2
IP Packet - Source IP: 192.168.43.93, Destination IP: 224.0.0.22, Protocol: 2
IP Packet - Source IP: 192.168.43.93, Destination IP: 224.0.0.22, Protocol: 2
UDP Packet - Source IP: 192.168.43.93, Destination IP: 224.0.0.251, Source Port: 5353, Destination Port: 5353
UDP Packet - Source IP: 192.168.43.93, Destination IP: 224.0.0.251, Source Port: 5353, Destination Port: 5353
UDP Packet - Source IP: 192.168.43.93, Destination IP: 224.0.0.252, Source Port: 63130, Destination Port: 5355
IP Packet - Source IP: 192.168.43.93, Destination IP: 224.0.0.22, Protocol: 2
UDP Packet - Source IP: 192.168.43.93, Destination IP: 192.168.43.1, Source Port: 61446, Destination Port: 53
UDP Packet - Source IP: 192.168.43.93, Destination IP: 192.168.43.1, Source Port: 63036, Destination Port: 53
UDP Packet - Source IP: 192.168.43.93, Destination IP: 192.168.43.1, Source Port: 64357, Destination Port: 53
^C
```

7. For filtering specific traffic I modified the sniffer again.
   To capture only specific types of traffic, such as HTTP traffic, modify the script to include a filter. Open the script again:

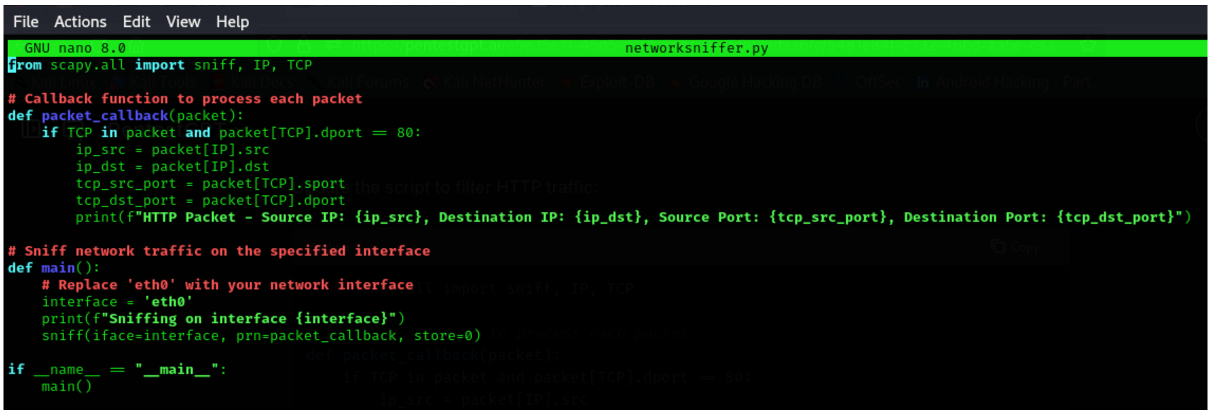


```
┌──(akash㉿Akash)-[~/Downloads]
└─$ sudo nano networksniffer.py
```

Script :

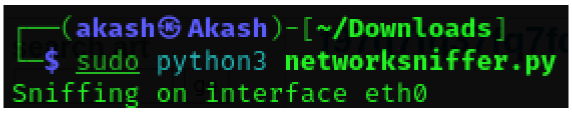
from scapy.all import sniff, IP, TCP

```
# Callback function to process each packet
def packet_callback(packet):
    if TCP in packet and packet[TCP].dport == 80:
        ip_src = packet[IP].src
        ip_dst = packet[IP].dst
        tcp_src_port = packet[TCP].sport
        tcp_dst_port = packet[TCP].dport
        print(f"HTTP Packet - Source IP: {ip_src}, Destination IP: {ip_dst},
Source Port: {tcp_src_port}, Destination Port: {tcp_dst_port}")

# Sniff network traffic on the specified interface
def main():
    # Replace 'eth0' with your network interface
    interface = 'eth0'
    print(f"Sniffing on interface {interface}")
    sniff(iface=interface, prn=packet_callback, store=0)

if __name__ == "__main__":
    main()
```



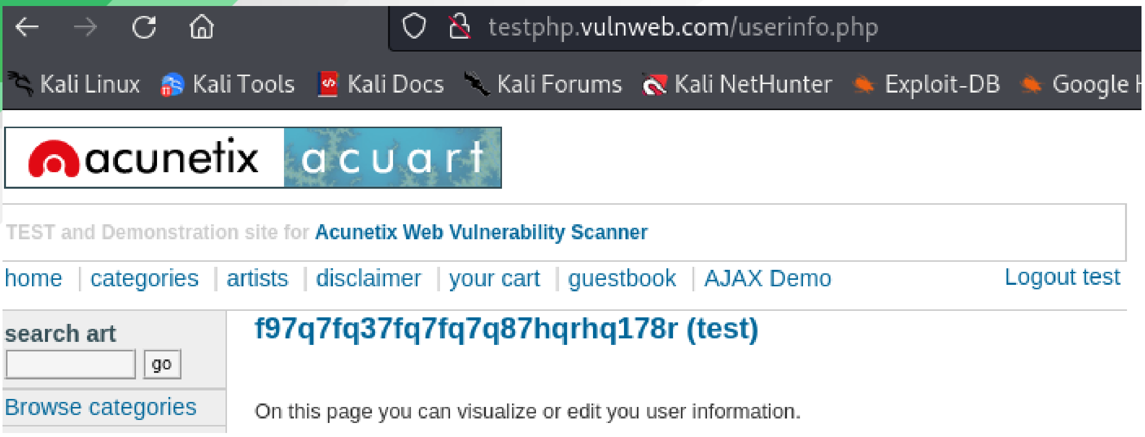
```
from scapy.all import sniff, IP, TCP

# Callback function to process each packet
def packet_callback(packet):
    if TCP in packet and packet[TCP].dport == 80:
        ip_src = packet[IP].src
        ip_dst = packet[IP].dst
        tcp_src_port = packet[TCP].sport
        tcp_dst_port = packet[TCP].dport
        print(f"HTTP Packet - Source IP: {ip_src}, Destination IP: {ip_dst}, Source Port: {tcp_src_port}, Destination Port: {tcp_dst_port}")

# Sniff network traffic on the specified interface
def main():
    # Replace 'eth0' with your network interface
    interface = 'eth0'
    print(f"Sniffing on interface {interface}")
    sniff(iface=interface, prn=packet_callback, store=0)

if __name__ == "__main__":
    main()
```

8. I run the sniffer again.
   Sniffer is waiting to capture any http request and response in the network.



```
┌──(akash㉿Akash)-[~/Downloads]
└─$ sudo python3 networksniffer.py
Sniffing on interface eth0
```

9. I browse a website to generate http requests.



TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo   Logout test

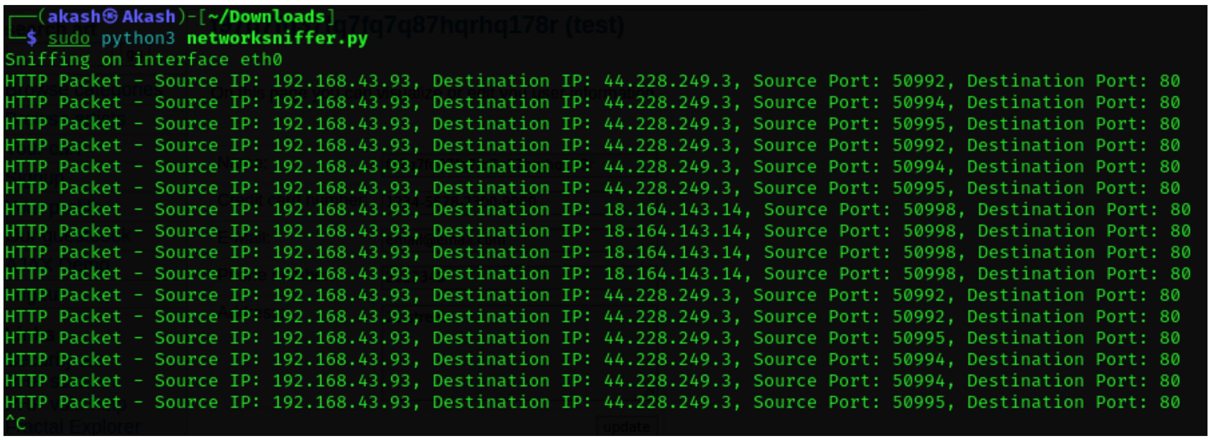
search art [ ] go

Browse categories

f97q7fq37fq7fq7q87hqrhq178r (test)

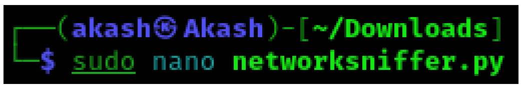
On this page you can visualize or edit you user information.

10.HTTP traffic generated while connecting to https://testphp.. filtered and captured successfully.



```
┌──(akash㉿Akash)-[~/Downloads]
└─$ sudo python3 networksniffer.py
Sniffing on interface eth0
HTTP Packet - Source IP: 192.168.43.93, Destination IP: 44.228.249.3, Source Port: 50992, Destination Port: 80
HTTP Packet - Source IP: 192.168.43.93, Destination IP: 44.228.249.3, Source Port: 50994, Destination Port: 80
HTTP Packet - Source IP: 192.168.43.93, Destination IP: 44.228.249.3, Source Port: 50995, Destination Port: 80
HTTP Packet - Source IP: 192.168.43.93, Destination IP: 44.228.249.3, Source Port: 50992, Destination Port: 80
HTTP Packet - Source IP: 192.168.43.93, Destination IP: 44.228.249.3, Source Port: 50994, Destination Port: 80
HTTP Packet - Source IP: 192.168.43.93, Destination IP: 44.228.249.3, Source Port: 50995, Destination Port: 80
HTTP Packet - Source IP: 192.168.43.93, Destination IP: 18.164.143.14, Source Port: 50998, Destination Port: 80
HTTP Packet - Source IP: 192.168.43.93, Destination IP: 18.164.143.14, Source Port: 50998, Destination Port: 80
HTTP Packet - Source IP: 192.168.43.93, Destination IP: 18.164.143.14, Source Port: 50998, Destination Port: 80
HTTP Packet - Source IP: 192.168.43.93, Destination IP: 18.164.143.14, Source Port: 50998, Destination Port: 80
HTTP Packet - Source IP: 192.168.43.93, Destination IP: 44.228.249.3, Source Port: 50992, Destination Port: 80
HTTP Packet - Source IP: 192.168.43.93, Destination IP: 44.228.249.3, Source Port: 50992, Destination Port: 80
HTTP Packet - Source IP: 192.168.43.93, Destination IP: 44.228.249.3, Source Port: 50995, Destination Port: 80
HTTP Packet - Source IP: 192.168.43.93, Destination IP: 44.228.249.3, Source Port: 50994, Destination Port: 80
HTTP Packet - Source IP: 192.168.43.93, Destination IP: 44.228.249.3, Source Port: 50994, Destination Port: 80
HTTP Packet - Source IP: 192.168.43.93, Destination IP: 44.228.249.3, Source Port: 50995, Destination Port: 80
^C
```

11.Saving captured traffic.
To save the captured packets to a file for later analysis, I modified the script to include saving functionality. Open the script again:

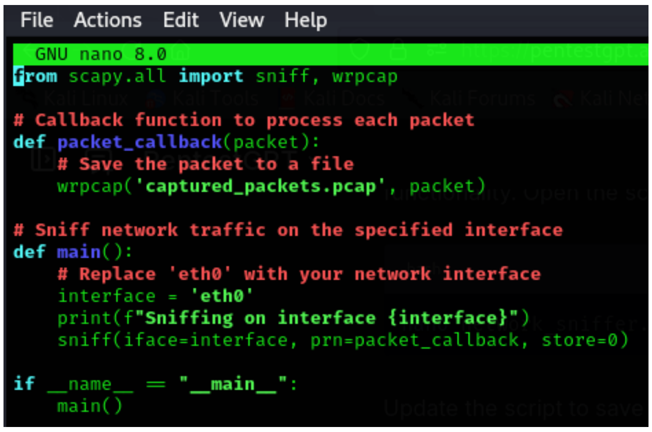


```
┌──(akash㉿Akash)-[~/Downloads]
└─$ sudo nano networksniffer.py
```

I Updated the script to save packets :

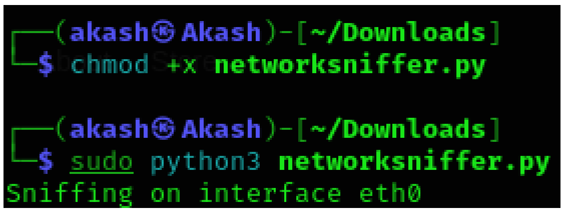
```
from scapy.all import sniff, wrpcap

# Callback function to process each packet
def packet_callback(packet):
    # Save the packet to a file
    wrpcap('captured_packets.pcap', packet)
```

```
# Sniff network traffic on the specified interface
def main():
    # Replace 'eth0' with your network interface
    interface = 'eth0'
    print(f"Sniffing on interface {interface}")
    sniff(iface=interface, prn=packet_callback, store=0)

if __name__ ==
"__main__":
    main()
```



```
from scapy.all import sniff, wrpcap

# Callback function to process each packet
def packet_callback(packet):
    # Save the packet to a file
    wrpcap('captured_packets.pcap', packet)

# Sniff network traffic on the specified interface
def main():
    # Replace 'eth0' with your network interface
    interface = 'eth0'
    print(f"Sniffing on interface {interface}")
    sniff(iface=interface, prn=packet_callback, store=0)

if __name__ == "__main__":
    main()
```

12. I give executable permission and run the sniffer again.

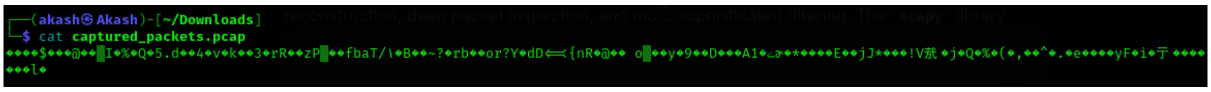


```
┌──(akash㉿Akash)-[~/Downloads]
└─$ chmod +x networksniffer.py

┌──(akash㉿Akash)-[~/Downloads]
└─$ sudo python3 networksniffer.py
Sniffing on interface eth0
```

13. The captured packets were saved in a file named captured_packets.pcap.

```
captured_packets.pcap
```

14. Check the packets.



```
┌──(akash㉿Akash)-[~/Downloads]
└─$ cat captured_packets.pcap
����$���@��▓I�%�Q�5.d��4�v�k��3�rR��zP▓��fbaT/\�B��~?�rb��or?Y�dD⟸{nR�@�� o▓��y�9��D���A1�ﺯ�*����E��jJ*���!V犹�j�Q�%�(�,��^�.�e����yF�i�亍����
���l�
```

Skills Acquired

During the CodeAlpha Internship, I acquired a comprehensive set of skills that are crucial for network security and analysis. These skills include:

- **Network Protocol Analysis:** Understanding the structure and behavior of various network protocols such as TCP, UDP, and IP.
- **Packet Inspection:** Gaining the ability to inspect and analyze network packets to identify patterns and anomalies.
- **Python Programming:** Enhancing my proficiency in Python, particularly in using libraries like Scapy for network packet manipulation.
- **Data Visualization:** Learning how to visualize network traffic data for better analysis and reporting.
- **Filtering and Searching:** Developing the ability to filter and search specific types of network traffic for targeted analysis.
- **Logging and Reporting:** Implementing logging and reporting mechanisms to document network traffic for future reference and analysis.

Conclusion

The CodeAlpha Internship provided an invaluable opportunity to delve into the intricacies of network traffic analysis. By building a network sniffer in Python, I gained hands-on experience in capturing, analyzing, and visualizing network packets. This project not only enhanced my understanding of network protocols and packet structures but also equipped me with the skills necessary for developing tools for network security and monitoring. The experience has been enriching and has laid a strong foundation for my future endeavors in cybersecurity.

Feedback and Learning

Feedback:

- The internship was well-structured, providing clear objectives and steps to complete the task.
- The use of Scapy for packet manipulation was particularly insightful and practical.
- The hands-on approach allowed for a deep understanding of network protocols and packet structures.

Learning:

- I learned the importance of detailed packet analysis in identifying potential security threats.
- The experience highlighted the need for continuous monitoring and logging of network traffic for effective security management.
- I gained practical skills in Python programming, which are essential for developing network security tools.

Acknowledgements

I would like to extend my gratitude to the CodeAlpha team for providing this enriching internship opportunity. Special thanks to [Mentor's Name] for their guidance and support throughout the project. The knowledge and skills I acquired during this internship will be invaluable in my future career in cybersecurity.

Future Goals

Moving forward, I aim to further enhance my skills in network security and analysis. My future goals include:

- **Advanced Network Analysis:** Exploring more advanced techniques for network traffic analysis and protocol inspection.
- **Tool Development:** Developing more sophisticated tools for network security and monitoring.
- **Certification:** Pursuing relevant certifications in network security and cybersecurity to validate my skills and knowledge.
- **Contribution to the Community:** Sharing my knowledge and experiences with the cybersecurity community through blogs, tutorials, and open-source projects.

Created by : Akash Das

Sign : Akash

Date : 08-03-2025
