packet capture using Scapy

Packet capture with Scapy involves the interception and analysis of network packets as they move through a network interface. Scapy is a versatile Python library that enables users to capture, manipulate, and examine network packets with significant flexibility.

What It Entails
Packet Interception: Scapy monitors a designated network interface, collecting packets in real-time.
Packet Filtering: Users can apply specific rules to capture particular packet types, such as TCP, UDP, HTTP, or ICMP, or to focus on packets associated with certain IP addresses.
Packet Inspection: The captured packets can be displayed, parsed, and analyzed, allowing users to examine details such as IP addresses, ports, payloads, and protocols.
Packet Logging: Users have the option to save captured packets to a file for subsequent offline analysis.

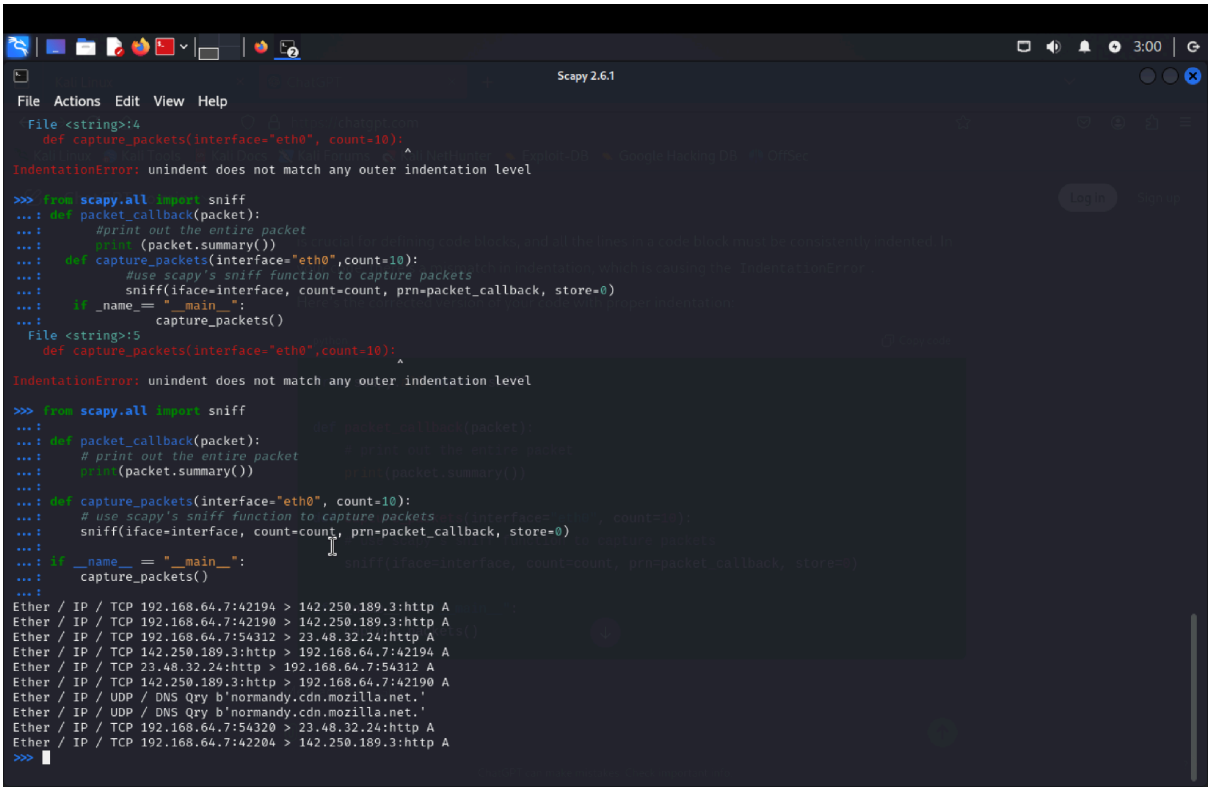
Key Features of Packet Capture with Scapy
Real-Time Analysis: Scapy facilitates the examination of packets as they are being captured.
Extensibility: Users can implement custom logic to manage or process packets.
Protocol Support: Scapy accommodates a broad range of network protocols.
Cross-Platform Compatibility: It operates on Linux, macOS, and Windows (with certain dependencies).

Use Cases
Network Monitoring: Evaluate traffic for debugging or oversight purposes.
Security Analysis: Identify intrusions or unusual activities.
Educational Tool: Gain insights into networking protocols by observing their behavior.
Debugging Assistance: Resolve network-related challenges.

A Basic Process for Packet Capture
Select an Interface: Identify the network interface to monitor (e.g., eth0, wlan0).
Set Up a Filter: Optionally, define a filter to capture only relevant packets.
Capture Packets: Initiate the listening process for packets.
Process the Data: Analyze or archive the packets.

For instance, capturing and displaying summary information of packets on eth0:

Why Opt for Scapy for Packet Capture?
Although tools like Wireshark or tcpdump offer comprehensive packet capture capabilities, Scapy provides unique advantages.


Scapy 2.6.1
File Actions Edit View Help
File <string>:4
def capture_packets(interface="eth0", count=10):
^
IndentationError: unindent does not match any outer indentation level
>>> from scapy.all import sniff
... : def packet_callback(packet):
... : #print out the entire packet
... : print (packet.summary())
... : def capture_packets(interface="eth0",count=10):
... : #use scapy's sniff function to capture packets
... : sniff(iface=interface, count=count, prn=packet_callback, store=0)
... : if _name_== "__main__":
... : capture_packets()
File <string>:5
def capture_packets(interface="eth0",count=10):
^
IndentationError: unindent does not match any outer indentation level
>>> from scapy.all import sniff
... :
... : def packet_callback(packet):
... : # print out the entire packet
... : print(packet.summary())
... :
... : def capture_packets(interface="eth0", count=10):
... : # use scapy's sniff function to capture packets
... : sniff(iface=interface, count=count, prn=packet_callback, store=0)
... :
... : if __name__ == "__main__":
... : capture_packets()
... :
Ether / IP / TCP 192.168.64.7:42194 > 142.250.189.3:http A
Ether / IP / TCP 192.168.64.7:42190 > 142.250.189.3:http A
Ether / IP / TCP 192.168.64.7:54312 > 23.48.32.24:http A
Ether / IP / TCP 142.250.189.3:http > 192.168.64.7:42194 A
Ether / IP / TCP 23.48.32.24:http > 192.168.64.7:54312 A
Ether / IP / TCP 142.250.189.3:http > 192.168.64.7:42190 A
Ether / IP / UDP / DNS Qry b'normandy.cdn.mozilla.net.'
Ether / IP / UDP / DNS Qry b'normandy.cdn.mozilla.net.'
Ether / IP / TCP 192.168.64.7:54320 > 23.48.32.24:http A
Ether / IP / TCP 192.168.64.7:42204 > 142.250.189.3:http A
>>>