Prodigy Infotech Internship

Task NO_5

Network Packet Analyzer

Develop a packet sniffer tool that captures and analyzes network packets. Display relevant information such as source and destination IP addresses, protocols, and payload data.
Ensure the ethical use of the tool for educational purposes.

Prodigy Infotech Ltd

Submitted by: Ahmad Ali

Submission date: October 05,2025

Task 5

Network Packet Analyzer

- **Code**

```
def analyze_packet(packet):
    """
    Analyze a single packet and print relevant details.
    """
    if IP not in packet:
        return  # Skip non-IP packets (e.g., ARP)

    ip_layer = packet[IP]
    src_ip = ip_layer.src
    dst_ip = ip_layer.dst
    protocol = ip_layer.proto  # Protocol number (e.g., 6 for
        TCP)

    # Map common protocols
    if protocol == 6:
        proto_name = "TCP"
    elif protocol == 17:
        proto_name = "UDP"
    elif protocol == 1:
```

```
    elif protocol == 1:
        proto_name = "ICMP"
    else:
        proto_name = f"Unknown ({protocol})"

    print(f"\n[+] Packet Captured:")
    print(f"    Source IP: {src_ip}")
    print(f"    Destination IP: {dst_ip}")
    print(f"    Protocol: {proto_name}")

    # Show ports for TCP/UDP or type for ICMP
    if TCP in packet:
        print(f"    Source Port: {packet[TCP].sport}")
        print(f"    Destination Port: {packet[TCP].dport}")
    elif UDP in packet:
        print(f"    Source Port: {packet[UDP].sport}")
        print(f"    Destination Port: {packet[UDP].dport}")
    elif ICMP in packet:
        print(f"    ICMP Type: {packet[ICMP].type}")
```

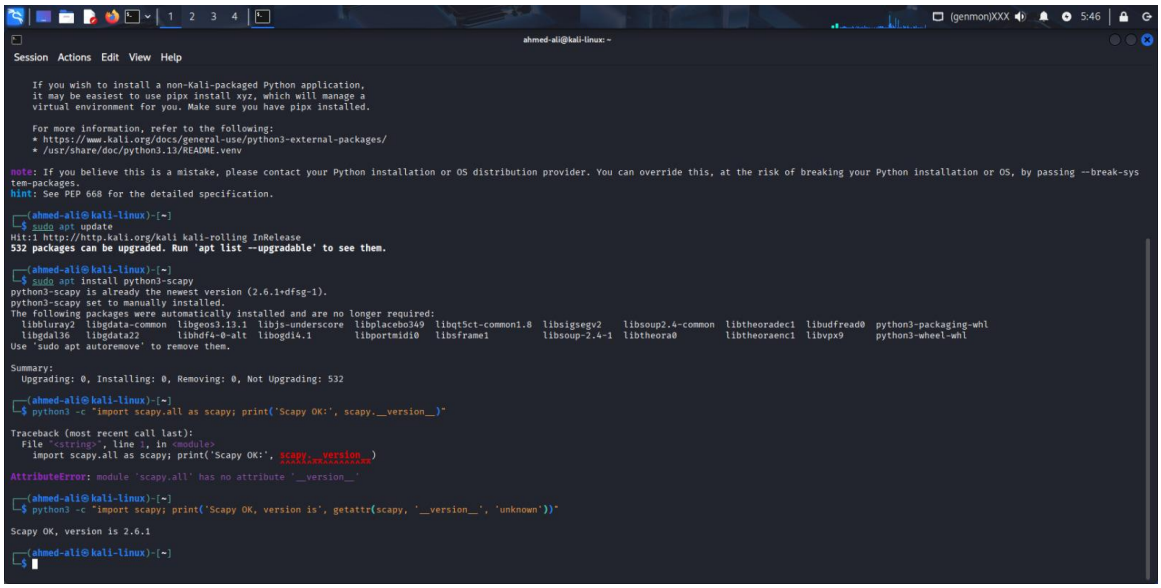
Task 5

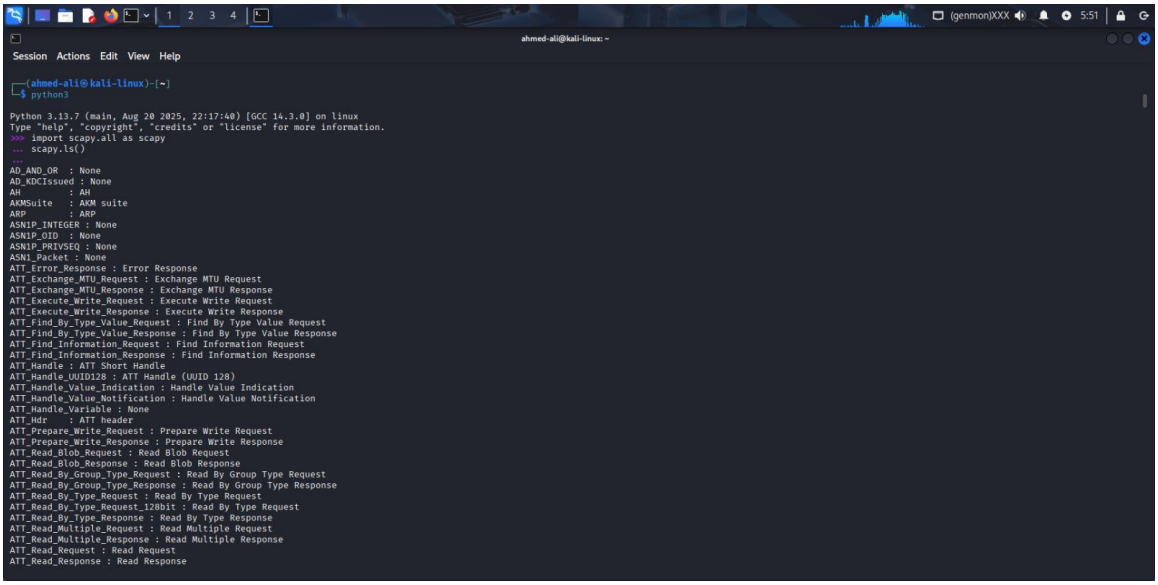
```
    if Raw in packet:
        payload = bytes(packet[Raw])
        if payload:
            print(f"    Payload (first 50 bytes, hex): {payload[
                :50].hex()}")
            if len(payload) > 50:
                print(f"    (Truncated; full payload length:
                    {len(payload)} bytes)")
        else:
            print("    No payload.")
    else:
        print("    No payload.")


def main(interface=None, count=20):
    """
    Start sniffing on the specified interface.
    """
    print(f"[*] Starting packet sniffer on {interface or
```

```
def main(interface=None, count=20):
    """
    Start sniffing on the specified interface.
    """
    print(f"[*] Starting packet sniffer on {interface or
        'default interface'} (capturing {count} packets)...")
    print("[*] Press Ctrl+C to stop early.\n")

    sniff(iface=interface, prn=analyze_packet, count=count,
        filter="ip")


if __name__ == "__main__":
    interface = sys.argv[1] if len(sys.argv) > 1 else None
    main(interface=interface)
```

Task 5

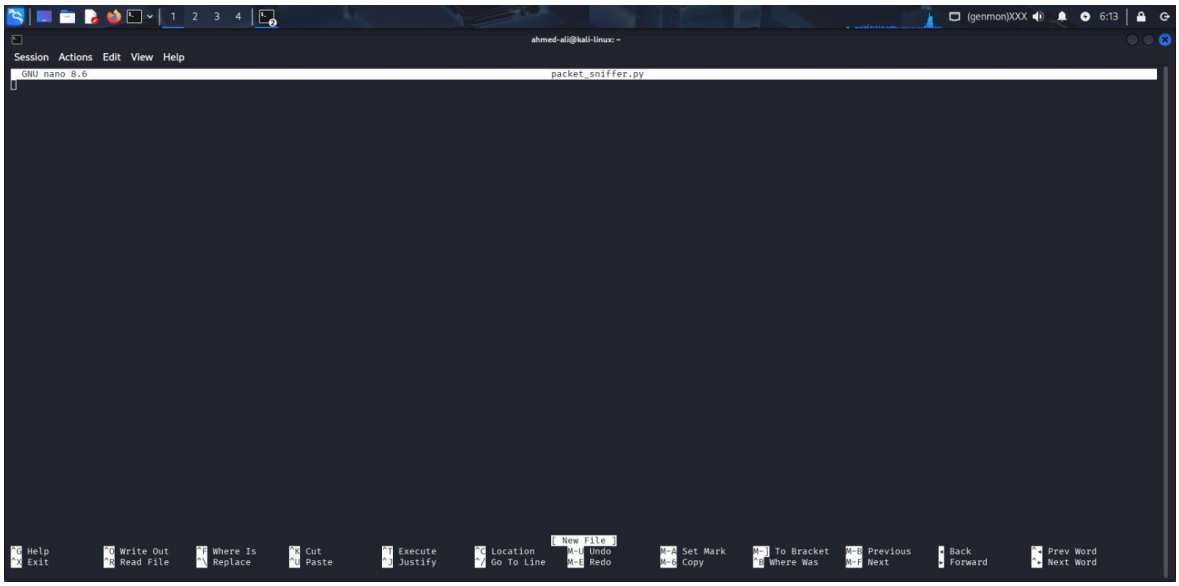
- **Install Scapy**



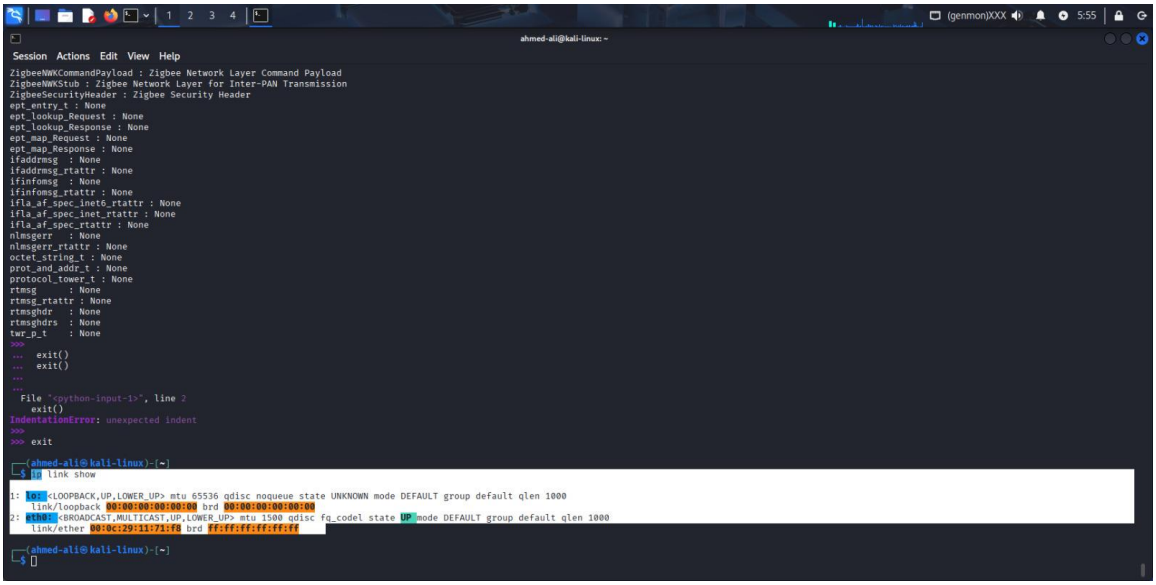


Task 5

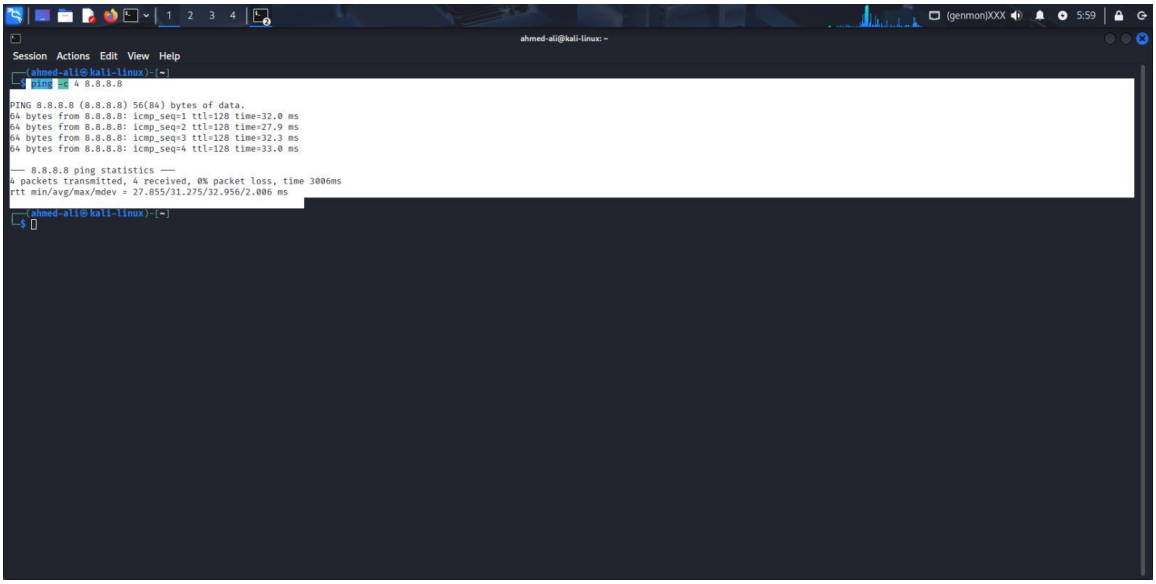
- **Upload code & Analyze**



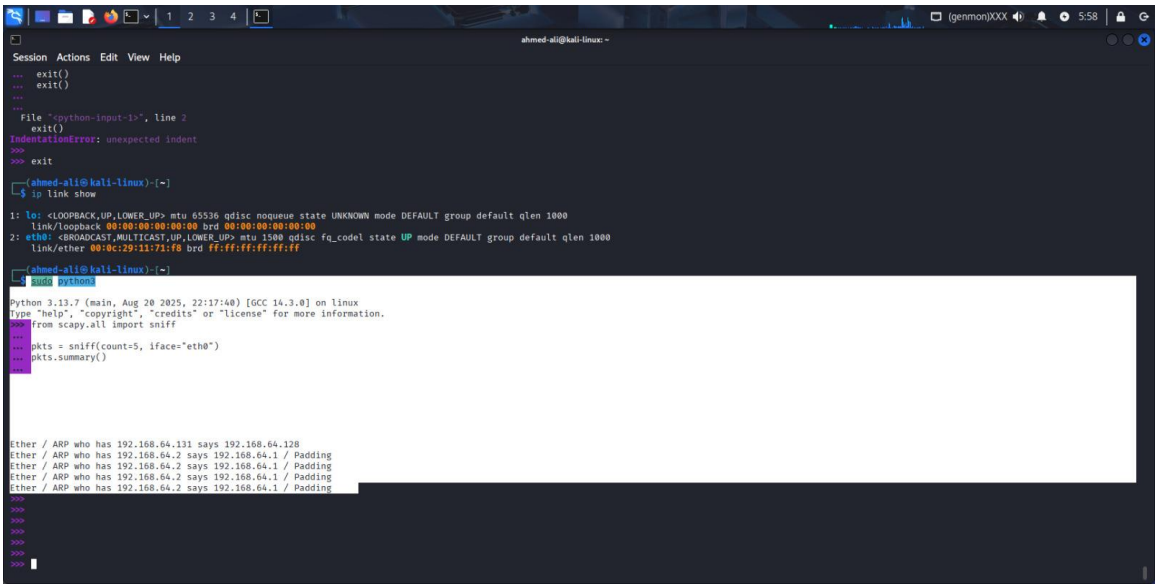
Task 5

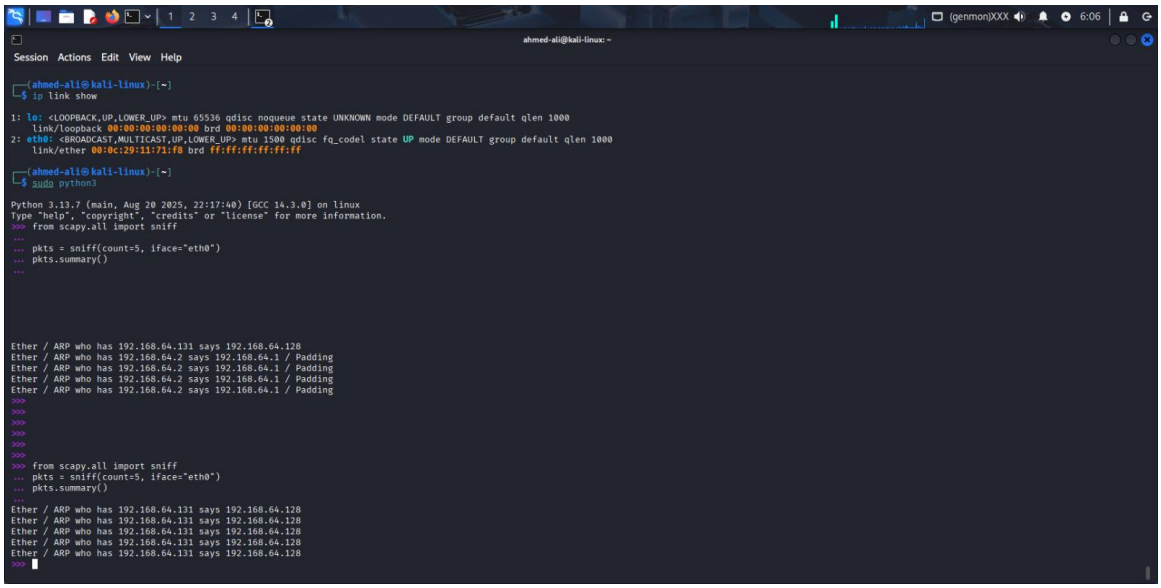


- **Packet Capturing**

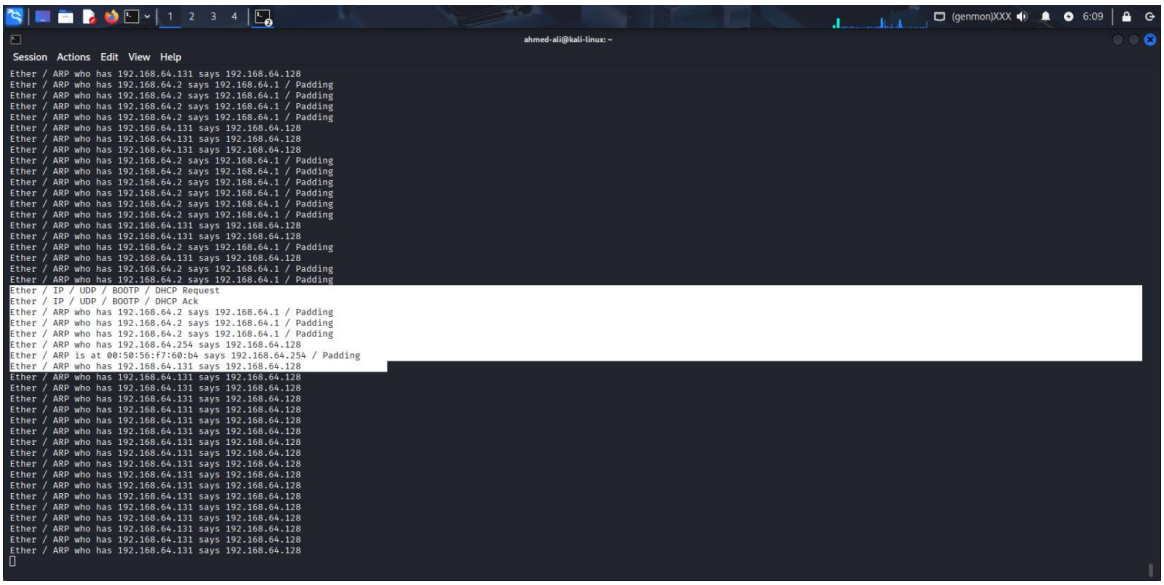


Task 5

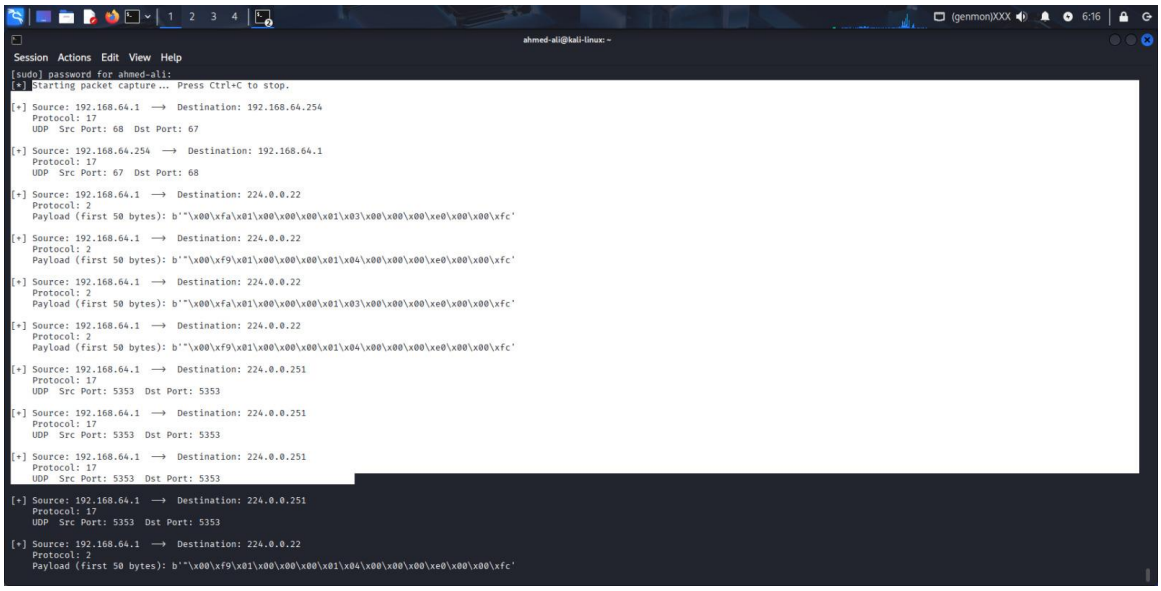

Session Actions Edit View Help
exit()
exit()
File "<python-input-1>", line 2
exit()
IndentationError: unexpected indent
>>> exit
(ahmed-ali㉿kali-linux)-[~]
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
link/ether 00:0c:29:11:71:f8 brd ff:ff:ff:ff:ff:ff
(ahmed-ali㉿kali-linux)-[~]
$ sudo python3
Python 3.13.7 (main, Aug 20 2025, 22:17:40) [GCC 14.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import sniff
... pkts = sniff(count=5, iface="eth0")
... pkts.summary()
Ether / ARP who has 192.168.64.131 says 192.168.64.128
Ether / ARP who has 192.168.64.2 says 192.168.64.1 / Padding
Ether / ARP who has 192.168.64.2 says 192.168.64.1 / Padding
Ether / ARP who has 192.168.64.2 says 192.168.64.1 / Padding
Ether / ARP who has 192.168.64.2 says 192.168.64.1 / Padding


Session Actions Edit View Help
(ahmed-ali㉿kali-linux)-[~]
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
link/ether 00:0c:29:11:71:f8 brd ff:ff:ff:ff:ff:ff
(ahmed-ali㉿kali-linux)-[~]
$ sudo python3
Python 3.13.7 (main, Aug 20 2025, 22:17:40) [GCC 14.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import sniff
... pkts = sniff(count=5, iface="eth0")
... pkts.summary()
Ether / ARP who has 192.168.64.131 says 192.168.64.128
Ether / ARP who has 192.168.64.2 says 192.168.64.1 / Padding
Ether / ARP who has 192.168.64.2 says 192.168.64.1 / Padding
Ether / ARP who has 192.168.64.2 says 192.168.64.1 / Padding
Ether / ARP who has 192.168.64.2 says 192.168.64.1 / Padding
>>> from scapy.all import sniff
... pkts = sniff(count=5, iface="eth0")
... pkts.summary()
Ether / ARP who has 192.168.64.131 says 192.168.64.128
Ether / ARP who has 192.168.64.131 says 192.168.64.128
Ether / ARP who has 192.168.64.131 says 192.168.64.128
Ether / ARP who has 192.168.64.131 says 192.168.64.128
Ether / ARP who has 192.168.64.131 says 192.168.64.128

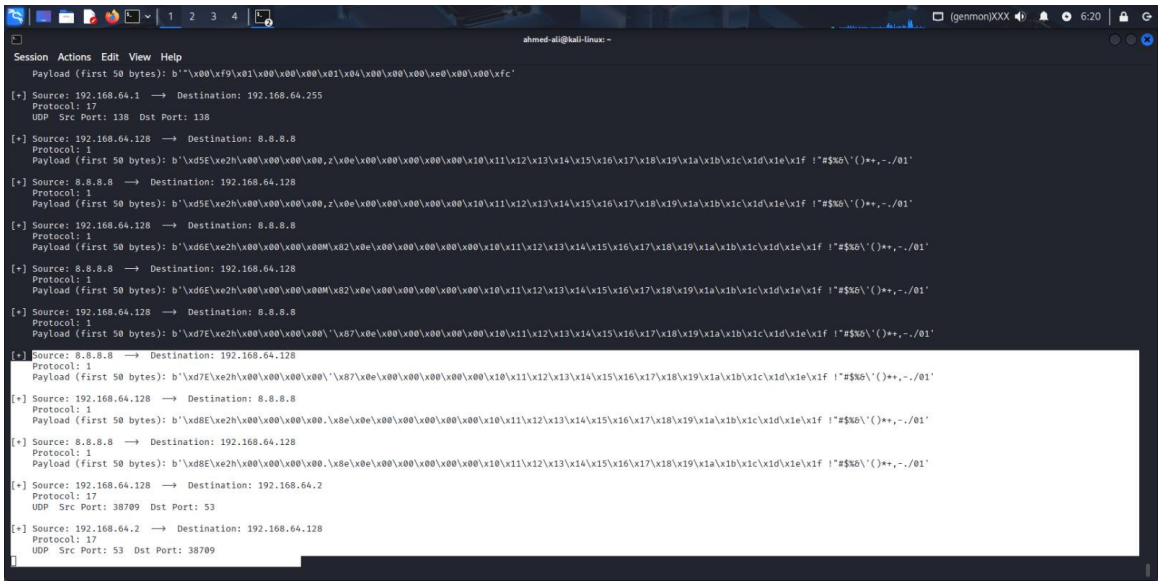
Task 5

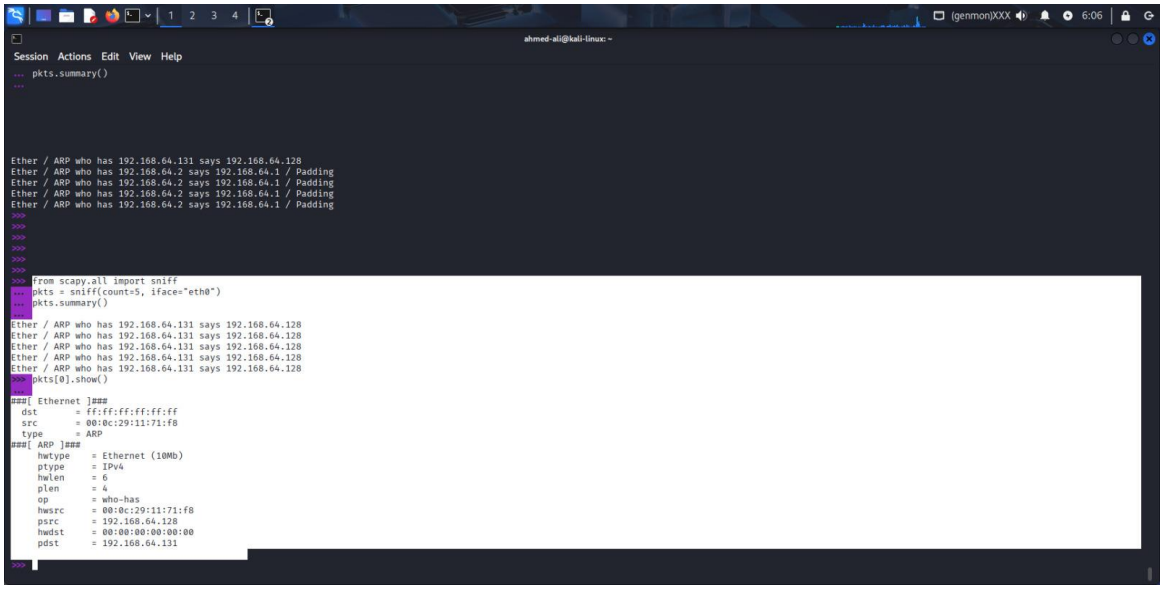


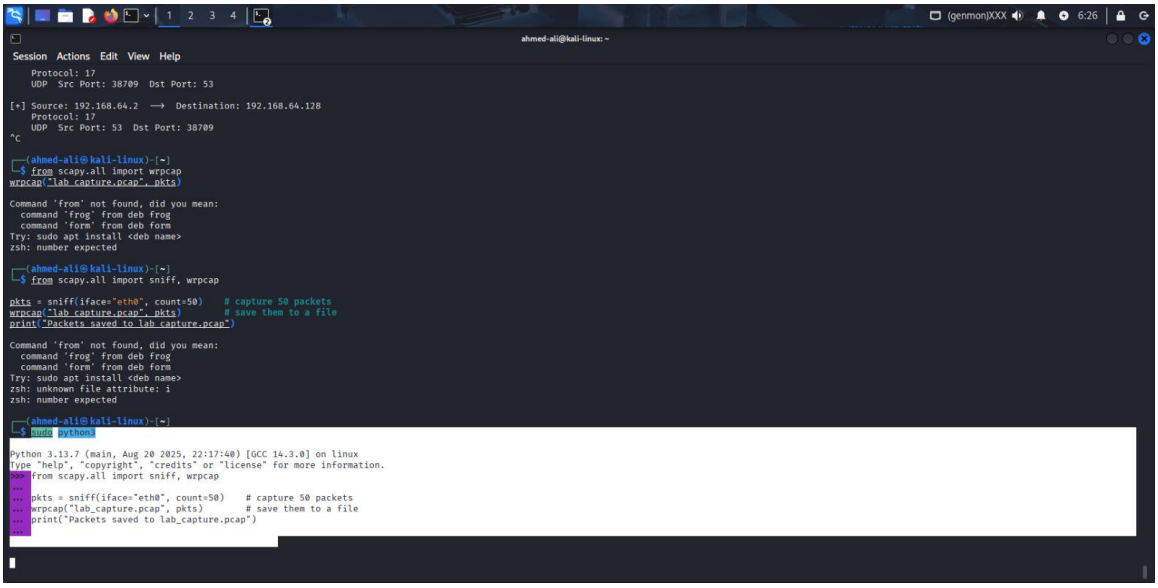
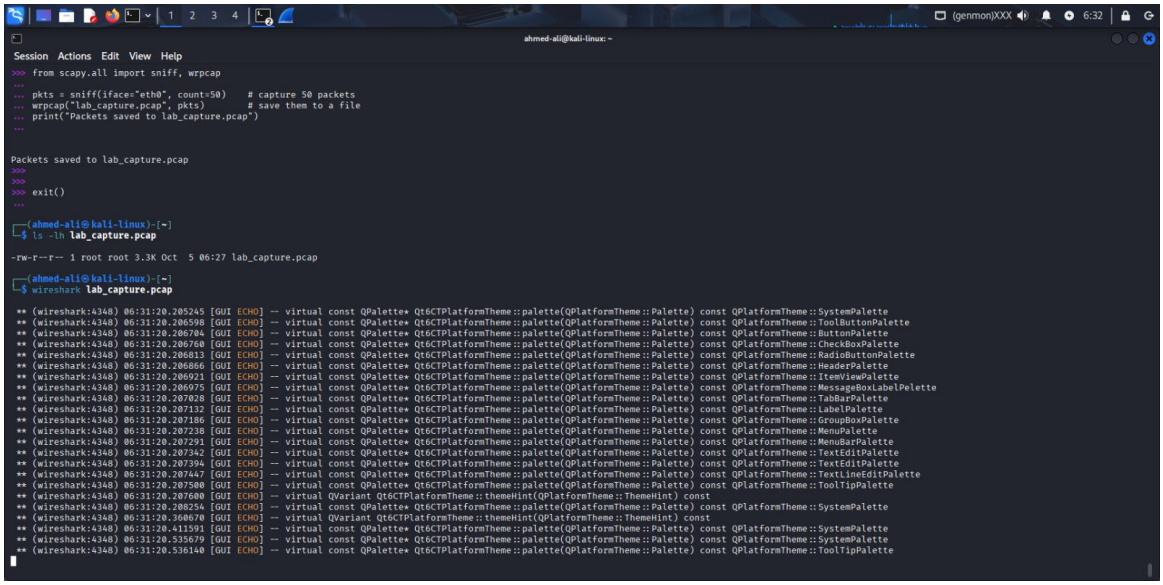
- **Simulate live Capture**



Task 5


ahmed-ali@kali-linux: ~
Session Actions Edit View Help
Payload (first 50 bytes): b'"\x00\xf9\x01\x00\x00\x00\x01\x04\x00\x00\x00\xe0\x00\x00\xfc'
[+] Source: 192.168.64.1 ⟶ Destination: 192.168.64.255
Protocol: 17
UDP Src Port: 138 Dst Port: 138
[+] Source: 192.168.64.128 ⟶ Destination: 8.8.8.8
Protocol: 1
Payload (first 50 bytes): b'\xd5E\xe2h\x00\x00\x00\x00,z\x0e\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01'
[+] Source: 8.8.8.8 ⟶ Destination: 192.168.64.128
Protocol: 1
Payload (first 50 bytes): b'\xd5E\xe2h\x00\x00\x00\x00,z\x0e\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01'
[+] Source: 192.168.64.128 ⟶ Destination: 8.8.8.8
Protocol: 1
Payload (first 50 bytes): b'\xd6E\xe2h\x00\x00\x00\x00M\x82\x0e\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01'
[+] Source: 8.8.8.8 ⟶ Destination: 192.168.64.128
Protocol: 1
Payload (first 50 bytes): b'\xd6E\xe2h\x00\x00\x00\x00M\x82\x0e\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01'
[+] Source: 192.168.64.128 ⟶ Destination: 8.8.8.8
Protocol: 1
Payload (first 50 bytes): b'\xd7E\xe2h\x00\x00\x00\x00\'\x87\x0e\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01'
[+] Source: 8.8.8.8 ⟶ Destination: 192.168.64.128
Protocol: 1
Payload (first 50 bytes): b'\xd7E\xe2h\x00\x00\x00\x00\'\x87\x0e\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01'
[+] Source: 192.168.64.128 ⟶ Destination: 8.8.8.8
Protocol: 1
Payload (first 50 bytes): b'\xd8E\xe2h\x00\x00\x00\x00.\x8e\x0e\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01'
[+] Source: 8.8.8.8 ⟶ Destination: 192.168.64.128
Protocol: 1
Payload (first 50 bytes): b'\xd8E\xe2h\x00\x00\x00\x00.\x8e\x0e\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01'
[+] Source: 192.168.64.128 ⟶ Destination: 192.168.64.2
Protocol: 17
UDP Src Port: 38709 Dst Port: 53
[+] Source: 192.168.64.2 ⟶ Destination: 192.168.64.128
Protocol: 17
UDP Src Port: 53 Dst Port: 38709


ahmed-ali@kali-linux: ~
Session Actions Edit View Help
... pkts.summary()
Ether / ARP who has 192.168.64.131 says 192.168.64.128
Ether / ARP who has 192.168.64.2 says 192.168.64.1 / Padding
Ether / ARP who has 192.168.64.2 says 192.168.64.1 / Padding
Ether / ARP who has 192.168.64.2 says 192.168.64.1 / Padding
Ether / ARP who has 192.168.64.2 says 192.168.64.1 / Padding
>>> from scapy.all import sniff
... pkts = sniff(count=5, iface="eth0")
... pkts.summary()
Ether / ARP who has 192.168.64.131 says 192.168.64.128
Ether / ARP who has 192.168.64.131 says 192.168.64.128
Ether / ARP who has 192.168.64.131 says 192.168.64.128
Ether / ARP who has 192.168.64.131 says 192.168.64.128
Ether / ARP who has 192.168.64.131 says 192.168.64.128
>>> pkts[0].show()
###[Ethernet]###
dst = ff:ff:ff:ff:ff:ff
src = 00:0c:29:11:71:f8
type = ARP
###[ARP]###
hwtype = Ethernet (10Mb)
ptype = IPv4
hwlen = 6
plen = 4
op = who-has
hwsrc = 00:0c:29:11:71:f8
psrc = 192.168.64.128
hwdst = 00:00:00:00:00:00
pdst = 192.168.64.131

Task 5

Save Analysis Results