Loïc Ivan AHOULOU

Network Security Analysis

From Reconnaissance to Intrusion Detection

AHOULOU Adouko Loïc Ivan – Cybersecurity's Student at ESATIC

+225 0768695702/ loicivan@protonmail.com

Friday, December 12th 2025 at 11H57 AM

Loïc Ivan AHOULOU

Table des matières

1. Introduction

This lab focused on practicing **network scanning**, **service enumeration**, and **packet sniffing** using **Nmap** and **Scapy**. We worked in a controlled internal network .

These exercises are foundational for ethical hacking, reconnaissance, and understanding how attackers map and analyze networks.

2. Lab Environment

- **Machine:**
  - Attacker : kali linux -> **192.168.121.128**
  - Target : Windows 10 -> **192.168.121.133**
- **Tools Used:**
  - Nmap
  - Scapy
  - Wireshark
- **Network: 192.168.121.0/24**

3. Nmap Practical Work

Nmap is a tool that scans networks to discover devices, open ports, services, and vulnerabilities.

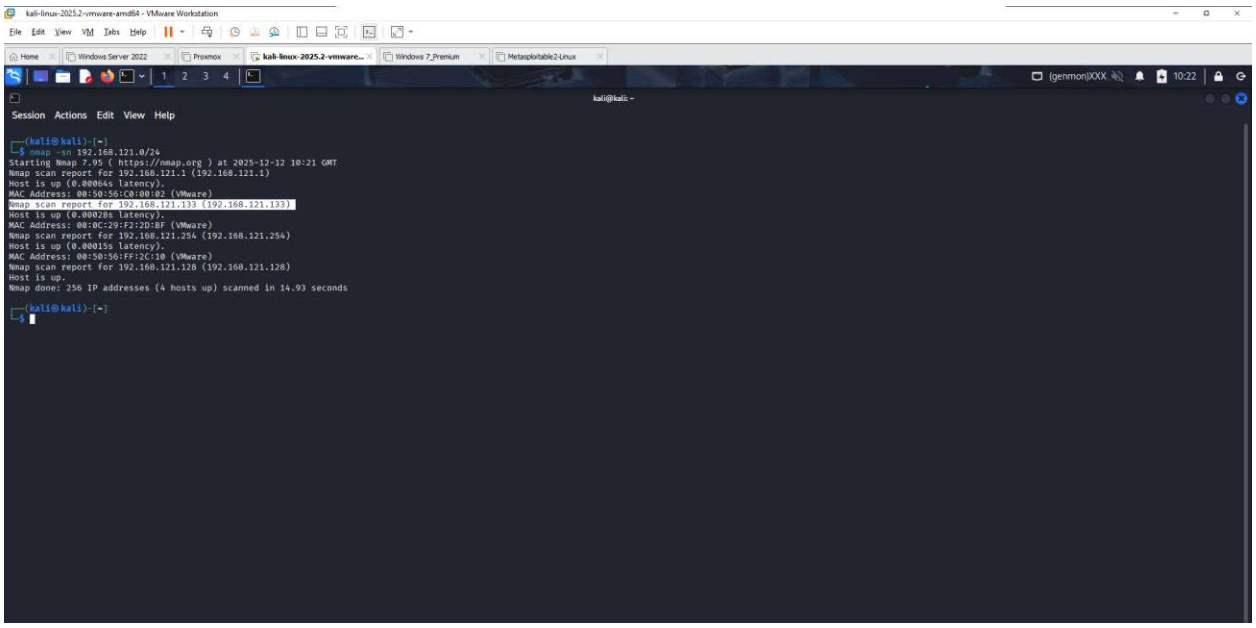
3.1 Host Discovery (Ping Sweep)

Command:

```
nmap -sn 192.168.121.0/24
```

Purpose: Check which hosts are alive on the network.

Result: 192.168.121.133 responded — host is UP.



```
┌──(kali㉿kali)-[~]
└─$ nmap -sn 192.168.121.0/24
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-12 10:21 GMT
Nmap scan report for 192.168.121.1 (192.168.121.1)
Host is up (0.00064s latency).
MAC Address: 00:50:56:C0:00:02 (VMware)
Nmap scan report for 192.168.121.133 (192.168.121.133)
Host is up (0.00028s latency).
MAC Address: 00:0C:29:F2:2D:BF (VMware)
Nmap scan report for 192.168.121.254 (192.168.121.254)
Host is up (0.00015s latency).
MAC Address: 00:50:56:FF:2C:10 (VMware)
Nmap scan report for 192.168.121.128 (192.168.121.128)
Host is up.
Nmap done: 256 IP addresses (4 hosts up) scanned in 14.93 seconds

┌──(kali㉿kali)-[~]
└─$ 
```

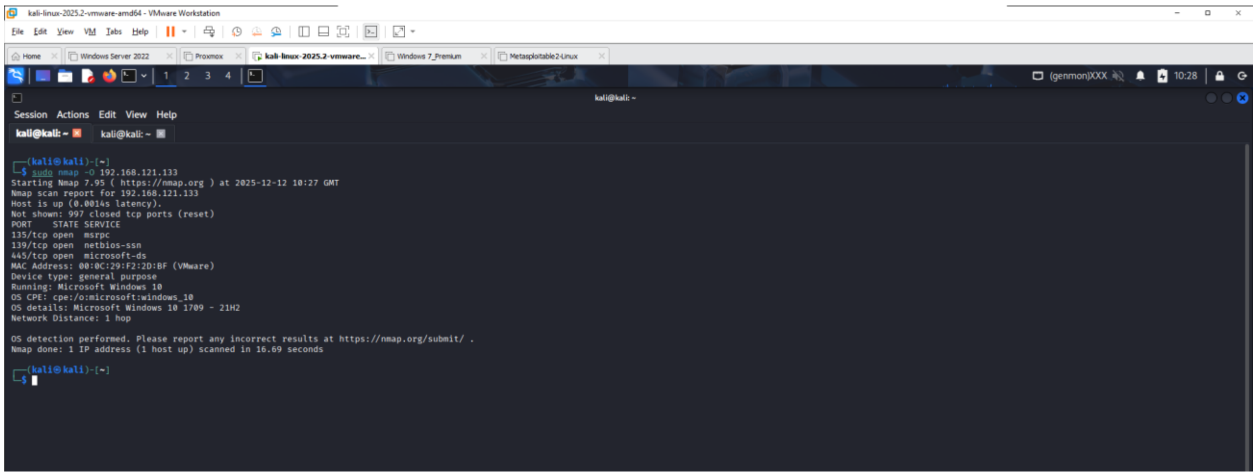
3.2 OS Detection

Command:

```
sudo nmap -O 192.168.121.133
```

Purpose: Identify the operating system by analyzing network responses.

Result: Nmap returned an OS guess based on TCP/IP fingerprints.



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -O 192.168.121.133
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-12 10:27 GMT
Nmap scan report for 192.168.121.133
Host is up (0.0014s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
MAC Address: 00:0C:29:F2:2D:BF (VMware)
Device type: general purpose
Running: Microsoft Windows 10
OS CPE: cpe:/o:microsoft:windows_10
OS details: Microsoft Windows 10 1709 - 21H2
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.69 seconds

┌──(kali㉿kali)-[~]
└─$ 
```

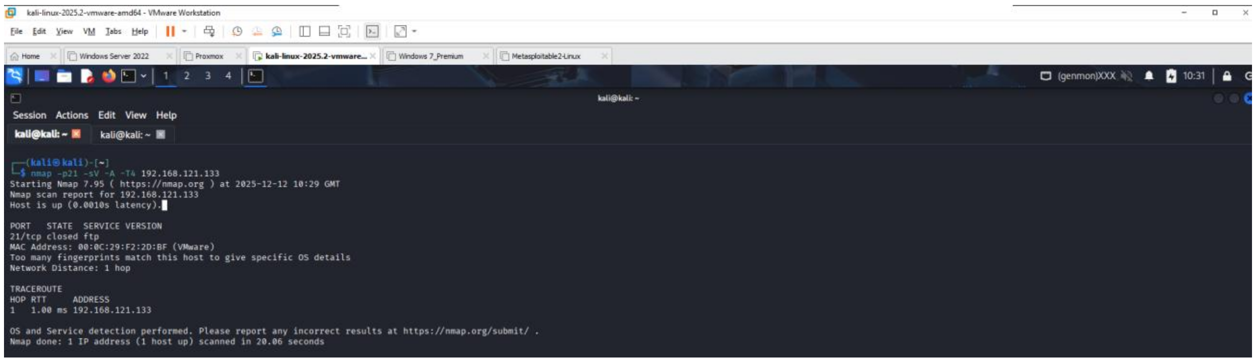
3.3 Service & Version Detection for FTP

Command:

```
nmap -p21 -sV -A -T4 192.168.121.133
```

Purpose: Enumerate services, detect versions, and gather extra details.

Result: Nmap detected the FTP service and additional information useful for penetration testing.


kali-linux-2025.2-vmware-amd64 - VMware Workstation
Home
Windows Server 2022
Proxmox
kali-linux-2025.2-vmware...
Windows 7_Premium
Metasploitable2-Linux
kali@kali: ~
Session Actions Edit View Help
kali@kali: ~
kali@kali: ~

```
┌──(kali㉿kali)-[~]
└─$ nmap -p21 -sV -A -T4 192.168.121.133
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-12 10:29 GMT
Nmap scan report for 192.168.121.133
Host is up (0.0010s latency).

PORT   STATE  SERVICE VERSION
21/tcp closed ftp
MAC Address: 00:0C:29:F2:2D:BF (VMware)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   1.00 ms 192.168.121.133

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.06 seconds
```

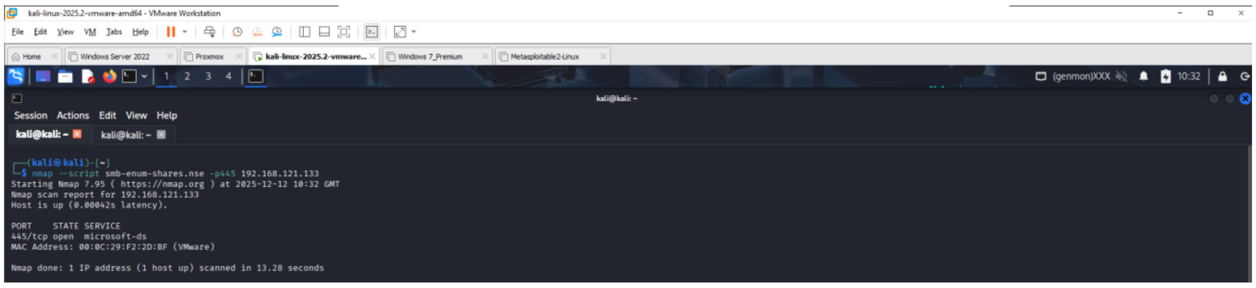
3.4 SMB Enumeration (Port 445)

Command:

```
nmap --script smb-enum-shares.nse -p445 10.6.6.23
```

Purpose: Identify accessible SMB shares.

Result: Showed available shares and potential misconfigurations.


kali-linux-2025.2-vmware-amd64 - VMware Workstation
Home
Windows Server 2022
Proxmox
kali-linux-2025.2-vmware...
Windows 7_Premium
Metasploitable2-Linux
kali@kali: ~
Session Actions Edit View Help
kali@kali: ~
kali@kali: ~

```
┌──(kali㉿kali)-[~]
└─$ nmap --script smb-enum-shares.nse -p445 192.168.121.133
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-12 10:32 GMT
Nmap scan report for 192.168.121.133
Host is up (0.00042s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 00:0C:29:F2:2D:BF (VMware)

Nmap done: 1 IP address (1 host up) scanned in 13.28 seconds
```

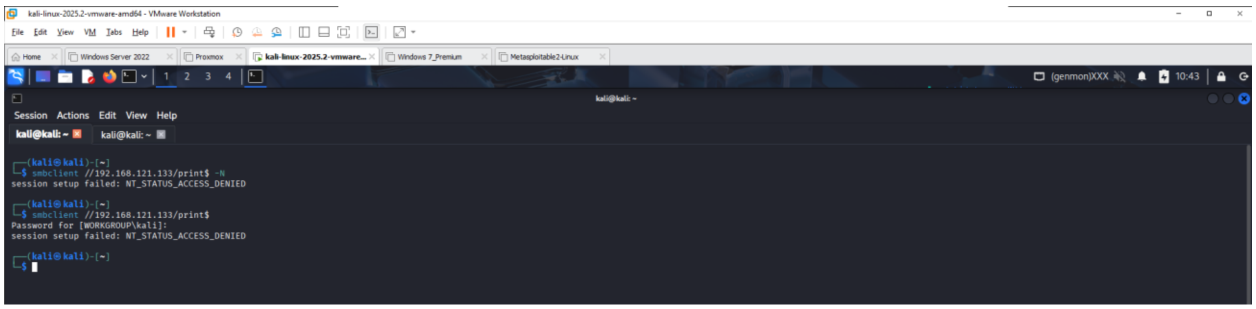
3.5 Manual SMB Access Test

Command:

```
smbclient //192.168.121.133 /print$ -N
```

Purpose: Check if anonymous SMB login is allowed.

Outcome: Access denied on the target wich is windows 10, for a password is required


kali-linux-2025.2-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Home
Windows Server 2022
Proxmox
kali-linux-2025.2-vmware...
Windows 7_Premium
Metasploitable2-Linux
(genmon)XXX
10:43
kali@kali: ~
Session Actions Edit View Help
kali@kali: ~
kali@kali: ~

```
┌──(kali㉿kali)-[~]
└─$ smbclient //192.168.121.133/print$ -N
session setup failed: NT_STATUS_ACCESS_DENIED

┌──(kali㉿kali)-[~]
└─$ smbclient //192.168.121.133/print$
Password for [WORKGROUP\kali]:
session setup failed: NT_STATUS_ACCESS_DENIED

┌──(kali㉿kali)-[~]
└─$ 
```

---

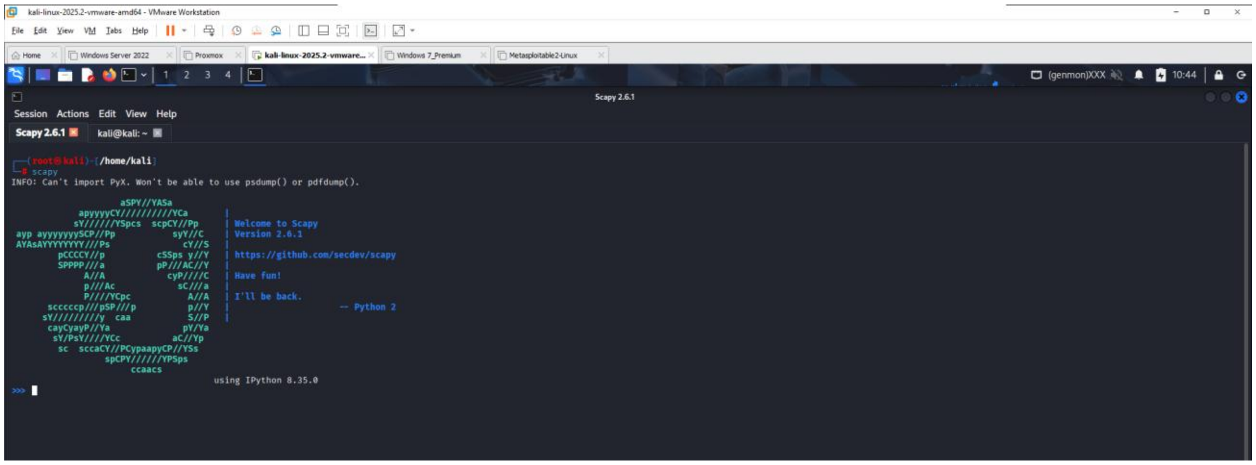
4. Scapy Practical Work

Scapy is a **powerful Python-based interactive packet manipulation tool** used for **network security, penetration testing, and protocol analysis**.

It can **craft**, **send**, **sniff**, **analyze**, and **manipulate** network packets at **ANY layer**, making it far more flexible than tools like ping, netcat, or even Wireshark.

4.1 Start Scapy

```
sudo su
scapy
```


kali-linux-2025.2-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Home
Windows Server 2022
Proxmox
kali-linux-2025.2-vmware...
Windows 7_Premium
Metasploitable2-Linux
Scapy 2.6.1
Session Actions Edit View Help
Scapy 2.6.1
kali@kali: ~

```
┌──(root㉿kali)-[/home/kali]
└─# scapy
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().

                     aSPY//YASa
             apyyyyCY//////////YCa       |
            sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.6.1
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | I'll be back.
       scccccp///pSP///p          p//Y   |                     -- Python 2
      sY/////////y  caa           S//P   |
       cayCyayP//Ya              pY/Ya
        sY/PsY////YCc          aC//Yp
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps
                       ccaacs
                                       using IPython 8.35.0
>>> 
```

4.2 Sniffing All Traffic

Command:

```
sniff()
```

Purpose: Capture live packets on the default interface.

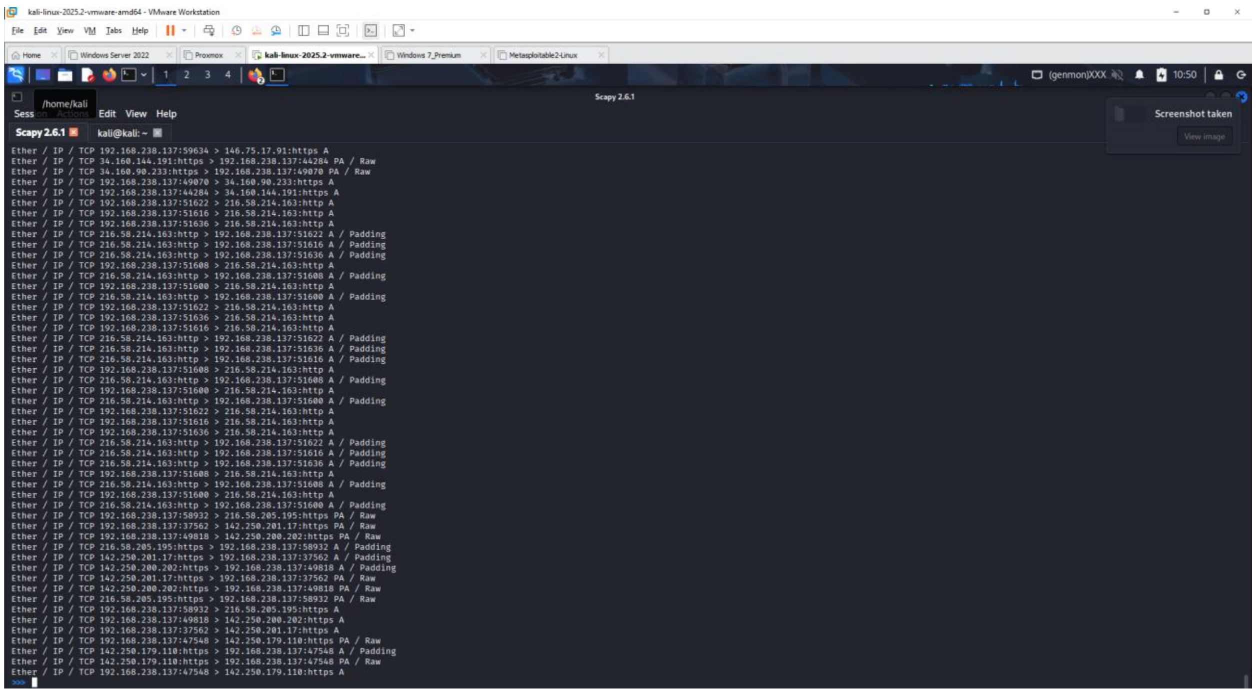
Action: Opened a second terminal and ran:

```
ping google.com
```

Result: Scapy captured ICMP, DNS, and other packets.

To stop sniffing: `Ctrl + C`.
You can store results:

```
paro = _
paro.summary()
```



```
Ether / IP / TCP 192.168.238.137:59634 > 146.75.17.91:https A
Ether / IP / TCP 34.160.144.191:https > 192.168.238.137:44284 PA / Raw
Ether / IP / TCP 34.160.90.233:https > 192.168.238.137:49070 PA / Raw
Ether / IP / TCP 192.168.238.137:49070 > 34.160.90.233:https A
Ether / IP / TCP 192.168.238.137:44284 > 34.160.144.191:https A
Ether / IP / TCP 192.168.238.137:51622 > 216.58.214.163:http A
Ether / IP / TCP 192.168.238.137:51616 > 216.58.214.163:http A
Ether / IP / TCP 192.168.238.137:51636 > 216.58.214.163:http A
Ether / IP / TCP 216.58.214.163:http > 192.168.238.137:51622 A / Padding
Ether / IP / TCP 216.58.214.163:http > 192.168.238.137:51616 A / Padding
Ether / IP / TCP 216.58.214.163:http > 192.168.238.137:51636 A / Padding
Ether / IP / TCP 192.168.238.137:51608 > 216.58.214.163:http A
Ether / IP / TCP 216.58.214.163:http > 192.168.238.137:51608 A / Padding
Ether / IP / TCP 192.168.238.137:51600 > 216.58.214.163:http A
Ether / IP / TCP 216.58.214.163:http > 192.168.238.137:51600 A / Padding
Ether / IP / TCP 192.168.238.137:51622 > 216.58.214.163:http A
Ether / IP / TCP 192.168.238.137:51636 > 216.58.214.163:http A
Ether / IP / TCP 192.168.238.137:51616 > 216.58.214.163:http A
Ether / IP / TCP 216.58.214.163:http > 192.168.238.137:51622 A / Padding
Ether / IP / TCP 216.58.214.163:http > 192.168.238.137:51636 A / Padding
Ether / IP / TCP 216.58.214.163:http > 192.168.238.137:51616 A / Padding
Ether / IP / TCP 192.168.238.137:51608 > 216.58.214.163:http A
Ether / IP / TCP 216.58.214.163:http > 192.168.238.137:51608 A / Padding
Ether / IP / TCP 192.168.238.137:51600 > 216.58.214.163:http A
Ether / IP / TCP 216.58.214.163:http > 192.168.238.137:51600 A / Padding
Ether / IP / TCP 192.168.238.137:51622 > 216.58.214.163:http A
Ether / IP / TCP 192.168.238.137:51616 > 216.58.214.163:http A
Ether / IP / TCP 192.168.238.137:51636 > 216.58.214.163:http A
Ether / IP / TCP 216.58.214.163:http > 192.168.238.137:51622 A / Padding
Ether / IP / TCP 216.58.214.163:http > 192.168.238.137:51616 A / Padding
Ether / IP / TCP 216.58.214.163:http > 192.168.238.137:51636 A / Padding
Ether / IP / TCP 192.168.238.137:51608 > 216.58.214.163:http A
Ether / IP / TCP 216.58.214.163:http > 192.168.238.137:51608 A / Padding
Ether / IP / TCP 192.168.238.137:51600 > 216.58.214.163:http A
Ether / IP / TCP 216.58.214.163:http > 192.168.238.137:51600 A / Padding
Ether / IP / TCP 192.168.238.137:58932 > 216.58.205.195:https PA / Raw
Ether / IP / TCP 192.168.238.137:37562 > 142.250.201.17:https PA / Raw
Ether / IP / TCP 192.168.238.137:49818 > 142.250.200.202:https PA / Raw
Ether / IP / TCP 216.58.205.195:https > 192.168.238.137:58932 A / Padding
Ether / IP / TCP 142.250.201.17:https > 192.168.238.137:37562 A / Padding
Ether / IP / TCP 142.250.200.202:https > 192.168.238.137:49818 A / Padding
Ether / IP / TCP 142.250.201.17:https > 192.168.238.137:37562 PA / Raw
Ether / IP / TCP 142.250.200.202:https > 192.168.238.137:49818 PA / Raw
Ether / IP / TCP 216.58.205.195:https > 192.168.238.137:58932 PA / Raw
Ether / IP / TCP 192.168.238.137:58932 > 216.58.205.195:https A
Ether / IP / TCP 192.168.238.137:49818 > 142.250.200.202:https A
Ether / IP / TCP 192.168.238.137:37562 > 142.250.201.17:https A
Ether / IP / TCP 192.168.238.137:47548 > 142.250.179.110:https PA / Raw
Ether / IP / TCP 142.250.179.110:https > 192.168.238.137:47548 A / Padding
Ether / IP / TCP 142.250.179.110:https > 192.168.238.137:47548 PA / Raw
Ether / IP / TCP 192.168.238.137:47548 > 142.250.179.110:https A
>>>
```

4.3 Sniffing on a Specific Interface

Command:

```
sniff(iface="eth1")
```

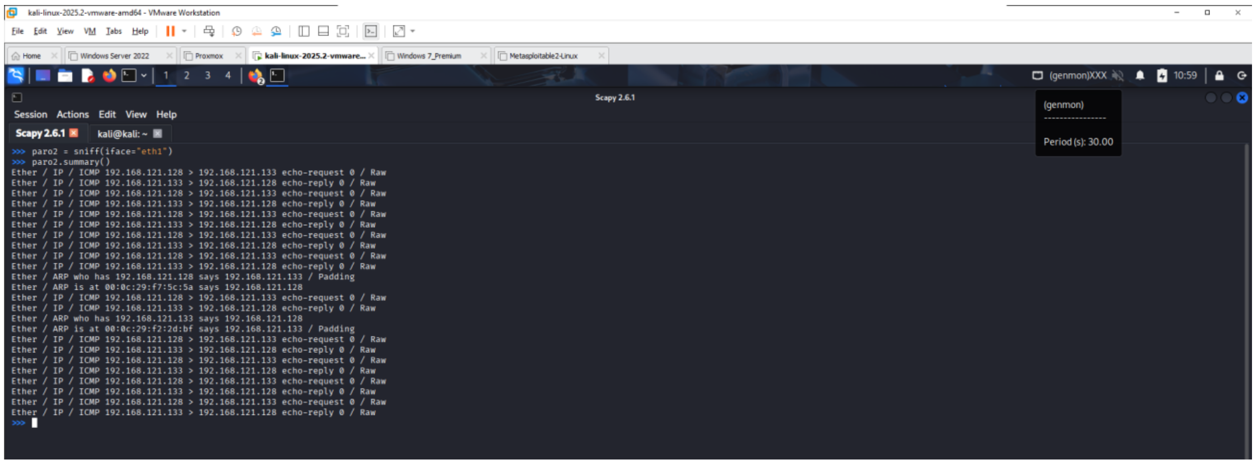
Purpose: Capture traffic on the network that contains our machines.

Triggered traffic by:

- Doing network pings

Saved results:

```
paro2 = sniff(face="eth1")
paro2.summary()
```



```
>>> paro2 = sniff(iface="eth1")
>>> paro2.summary()
Ether / IP / ICMP 192.168.121.128 > 192.168.121.133 echo-request 0 / Raw
Ether / IP / ICMP 192.168.121.133 > 192.168.121.128 echo-reply 0 / Raw
Ether / IP / ICMP 192.168.121.128 > 192.168.121.133 echo-request 0 / Raw
Ether / IP / ICMP 192.168.121.133 > 192.168.121.128 echo-reply 0 / Raw
Ether / IP / ICMP 192.168.121.128 > 192.168.121.133 echo-request 0 / Raw
Ether / IP / ICMP 192.168.121.133 > 192.168.121.128 echo-reply 0 / Raw
Ether / IP / ICMP 192.168.121.128 > 192.168.121.133 echo-request 0 / Raw
Ether / IP / ICMP 192.168.121.133 > 192.168.121.128 echo-reply 0 / Raw
Ether / IP / ICMP 192.168.121.128 > 192.168.121.133 echo-request 0 / Raw
Ether / IP / ICMP 192.168.121.133 > 192.168.121.128 echo-reply 0 / Raw
Ether / ARP who has 192.168.121.128 says 192.168.121.133 / Padding
Ether / ARP is at 00:0c:29:f7:5c:5a says 192.168.121.128
Ether / IP / ICMP 192.168.121.128 > 192.168.121.133 echo-request 0 / Raw
Ether / IP / ICMP 192.168.121.133 > 192.168.121.128 echo-reply 0 / Raw
Ether / ARP who has 192.168.121.133 says 192.168.121.128
Ether / ARP is at 00:0c:29:f2:2d:bf says 192.168.121.133 / Padding
Ether / IP / ICMP 192.168.121.128 > 192.168.121.133 echo-request 0 / Raw
Ether / IP / ICMP 192.168.121.133 > 192.168.121.128 echo-reply 0 / Raw
Ether / IP / ICMP 192.168.121.128 > 192.168.121.133 echo-request 0 / Raw
Ether / IP / ICMP 192.168.121.133 > 192.168.121.128 echo-reply 0 / Raw
Ether / IP / ICMP 192.168.121.128 > 192.168.121.133 echo-request 0 / Raw
Ether / IP / ICMP 192.168.121.133 > 192.168.121.128 echo-reply 0 / Raw
Ether / IP / ICMP 192.168.121.128 > 192.168.121.133 echo-request 0 / Raw
Ether / IP / ICMP 192.168.121.133 > 192.168.121.128 echo-reply 0 / Raw
>>> 
```

4.4 Filtering Only ICMP

Command:

```
sniff(iface="eth1", filter="icmp", count=5)
```

Test traffic:

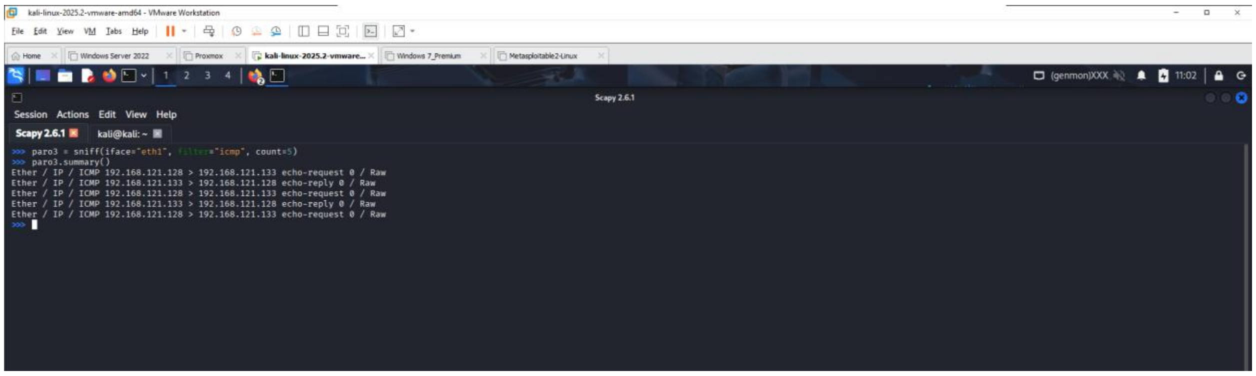
```
ping 192.168.121.133
```

Captured exactly 5 ICMP packets.

Stored and inspected:

```
paro3 = sniff(iface="eth1", filter="icmp", count=5)

paro3.summary()
```

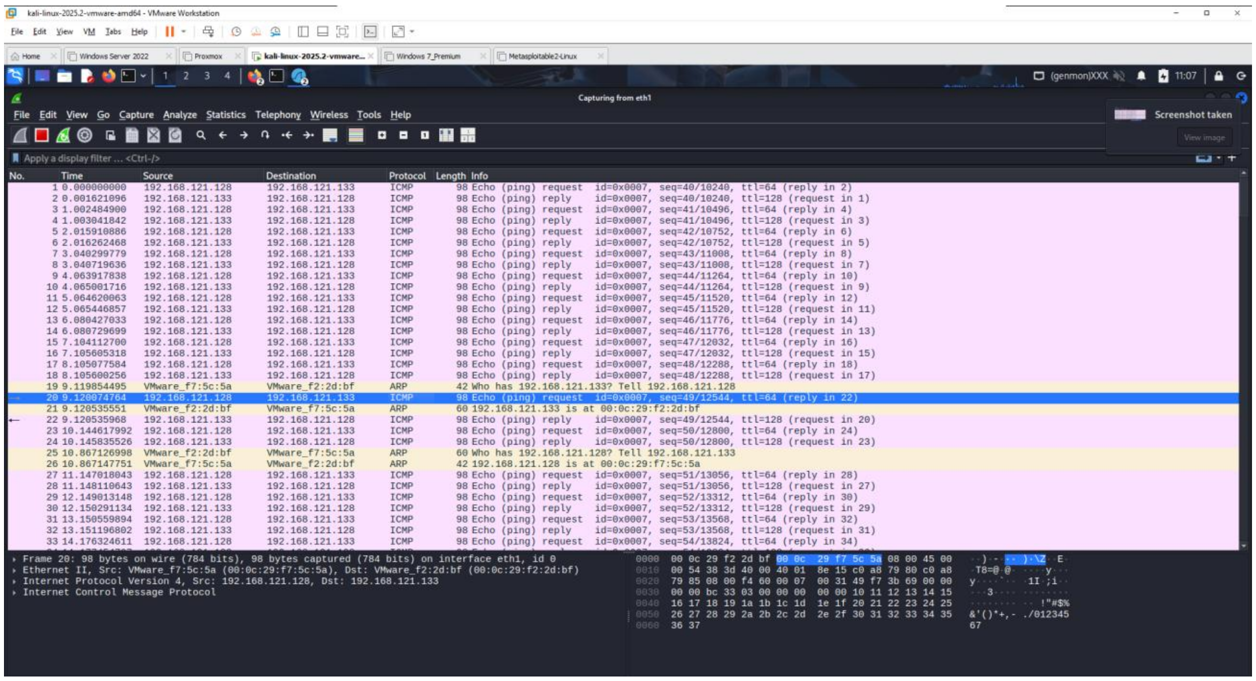


```
>>> paro3 = sniff(iface="eth1", filter="icmp", count=5)
>>> paro3.summary()
Ether / IP / ICMP 192.168.121.128 > 192.168.121.133 echo-request 0 / Raw
Ether / IP / ICMP 192.168.121.133 > 192.168.121.128 echo-reply 0 / Raw
Ether / IP / ICMP 192.168.121.128 > 192.168.121.133 echo-request 0 / Raw
Ether / IP / ICMP 192.168.121.133 > 192.168.121.128 echo-reply 0 / Raw
Ether / IP / ICMP 192.168.121.128 > 192.168.121.133 echo-request 0 / Raw
>>> 
```

5. Wireshark

5.1 Live packets sniffing

Capturing packets *as they travel* on a network interface eth0.


kali-linux-2025.2-vmware-amd64 - VMware Workstation
Capturing from eth1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Screenshot taken
View image

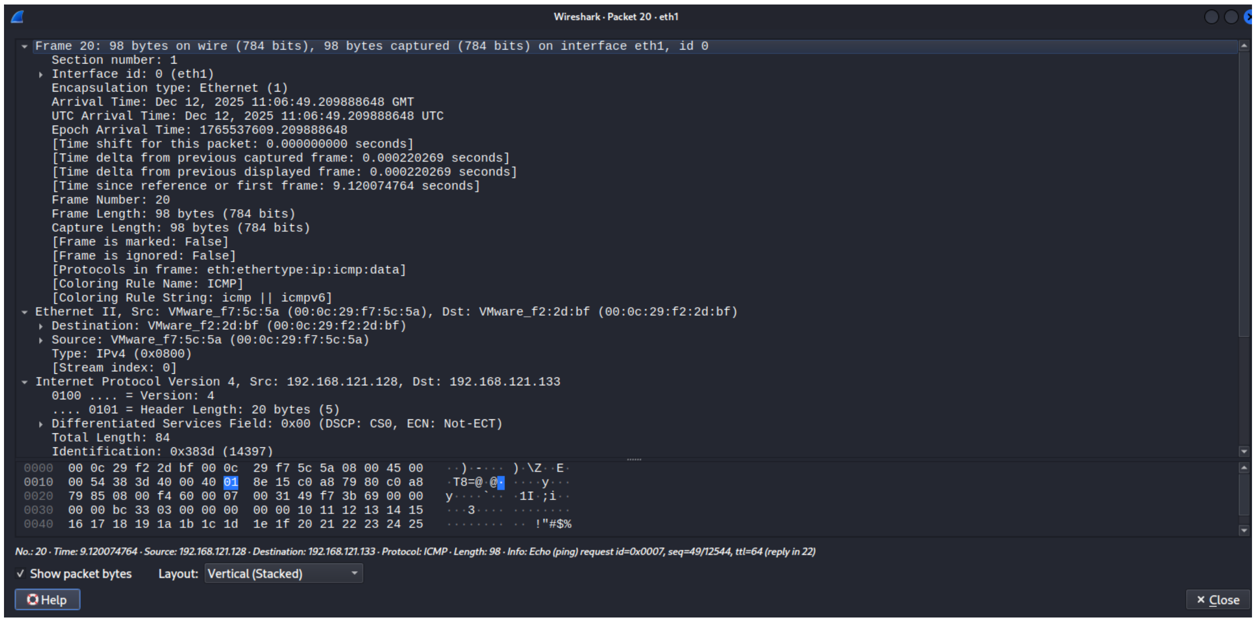
```
No.   Time           Source             Destination        Protocol Length Info
    1 0.000000000    192.168.121.128    192.168.121.133    ICMP     98 Echo (ping) request  id=0x0007, seq=40/10240, ttl=64 (reply in 2)
    2 0.001621096    192.168.121.133    192.168.121.128    ICMP     98 Echo (ping) reply    id=0x0007, seq=40/10240, ttl=128 (request in 1)
    3 1.002484900    192.168.121.128    192.168.121.133    ICMP     98 Echo (ping) request  id=0x0007, seq=41/10496, ttl=64 (reply in 4)
    4 1.003041842    192.168.121.133    192.168.121.128    ICMP     98 Echo (ping) reply    id=0x0007, seq=41/10496, ttl=128 (request in 3)
    5 2.015910886    192.168.121.128    192.168.121.133    ICMP     98 Echo (ping) request  id=0x0007, seq=42/10752, ttl=64 (reply in 6)
    6 2.016262468    192.168.121.133    192.168.121.128    ICMP     98 Echo (ping) reply    id=0x0007, seq=42/10752, ttl=128 (request in 5)
    7 3.040299779    192.168.121.128    192.168.121.133    ICMP     98 Echo (ping) request  id=0x0007, seq=43/11008, ttl=64 (reply in 8)
    8 3.040719636    192.168.121.133    192.168.121.128    ICMP     98 Echo (ping) reply    id=0x0007, seq=43/11008, ttl=128 (request in 7)
    9 4.063917838    192.168.121.128    192.168.121.133    ICMP     98 Echo (ping) request  id=0x0007, seq=44/11264, ttl=64 (reply in 10)
   10 4.065001716    192.168.121.133    192.168.121.128    ICMP     98 Echo (ping) reply    id=0x0007, seq=44/11264, ttl=128 (request in 9)
   11 5.064620063    192.168.121.128    192.168.121.133    ICMP     98 Echo (ping) request  id=0x0007, seq=45/11520, ttl=64 (reply in 12)
   12 5.065446857    192.168.121.133    192.168.121.128    ICMP     98 Echo (ping) reply    id=0x0007, seq=45/11520, ttl=128 (request in 11)
   13 6.080427033    192.168.121.128    192.168.121.133    ICMP     98 Echo (ping) request  id=0x0007, seq=46/11776, ttl=64 (reply in 14)
   14 6.080729699    192.168.121.133    192.168.121.128    ICMP     98 Echo (ping) reply    id=0x0007, seq=46/11776, ttl=128 (request in 13)
   15 7.104112700    192.168.121.128    192.168.121.133    ICMP     98 Echo (ping) request  id=0x0007, seq=47/12032, ttl=64 (reply in 16)
   16 7.105605318    192.168.121.133    192.168.121.128    ICMP     98 Echo (ping) reply    id=0x0007, seq=47/12032, ttl=128 (request in 15)
   17 8.105077584    192.168.121.128    192.168.121.133    ICMP     98 Echo (ping) request  id=0x0007, seq=48/12288, ttl=64 (reply in 18)
   18 8.105600256    192.168.121.133    192.168.121.128    ICMP     98 Echo (ping) reply    id=0x0007, seq=48/12288, ttl=128 (request in 17)
   19 9.119854495    VMware_f7:5c:5a    VMware_f2:2d:bf    ARP      42 Who has 192.168.121.133? Tell 192.168.121.128
   20 9.120074764    192.168.121.128    192.168.121.133    ICMP     98 Echo (ping) request  id=0x0007, seq=49/12544, ttl=64 (reply in 22)
   21 9.120535551    VMware_f2:2d:bf    VMware_f7:5c:5a    ARP      60 192.168.121.133 is at 00:0c:29:f2:2d:bf
   22 9.120535968    192.168.121.133    192.168.121.128    ICMP     98 Echo (ping) reply    id=0x0007, seq=49/12544, ttl=128 (request in 20)
   23 10.144617992   192.168.121.128    192.168.121.133    ICMP     98 Echo (ping) request  id=0x0007, seq=50/12800, ttl=64 (reply in 24)
   24 10.145835526   192.168.121.133    192.168.121.128    ICMP     98 Echo (ping) reply    id=0x0007, seq=50/12800, ttl=128 (request in 23)
   25 10.867126998   VMware_f2:2d:bf    VMware_f7:5c:5a    ARP      60 Who has 192.168.121.128? Tell 192.168.121.133
   26 10.867147751   VMware_f7:5c:5a    VMware_f2:2d:bf    ARP      42 192.168.121.128 is at 00:0c:29:f7:5c:5a
   27 11.147018043   192.168.121.128    192.168.121.133    ICMP     98 Echo (ping) request  id=0x0007, seq=51/13056, ttl=64 (reply in 28)
   28 11.148110643   192.168.121.133    192.168.121.128    ICMP     98 Echo (ping) reply    id=0x0007, seq=51/13056, ttl=128 (request in 27)
   29 12.149013148   192.168.121.128    192.168.121.133    ICMP     98 Echo (ping) request  id=0x0007, seq=52/13312, ttl=64 (reply in 30)
   30 12.150291134   192.168.121.133    192.168.121.128    ICMP     98 Echo (ping) reply    id=0x0007, seq=52/13312, ttl=128 (request in 29)
   31 13.150559894   192.168.121.128    192.168.121.133    ICMP     98 Echo (ping) request  id=0x0007, seq=53/13568, ttl=64 (reply in 32)
   32 13.151196802   192.168.121.133    192.168.121.128    ICMP     98 Echo (ping) reply    id=0x0007, seq=53/13568, ttl=128 (request in 31)
   33 14.176324611   192.168.121.128    192.168.121.133    ICMP     98 Echo (ping) request  id=0x0007, seq=54/13824, ttl=64 (reply in 34)
```

```
› Frame 20: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth1, id 0
› Ethernet II, Src: VMware_f7:5c:5a (00:0c:29:f7:5c:5a), Dst: VMware_f2:2d:bf (00:0c:29:f2:2d:bf)
› Internet Protocol Version 4, Src: 192.168.121.128, Dst: 192.168.121.133
› Internet Control Message Protocol
```

```
0000  00 0c 29 f2 2d bf 00 0c  29 f7 5c 5a 08 00 45 00   ··)·-···)·\Z··E·
0010  00 54 38 3d 40 00 40 01  8e 15 c0 a8 79 80 c0 a8   ·T8=@·@·····y···
0020  79 85 08 00 f4 60 00 07  00 31 49 f7 3b 69 00 00   y····`···1I·;i··
0030  00 00 bc 33 03 00 00 00  00 00 10 11 12 13 14 15   ···3············
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ·········· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,-./012345
0060  36 37                                              67
```

5.2 Packets analyzing

Analyzing the packets immediately (real-time). We can see more information according to the TCP/IP model.



```
▾ Frame 20: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth1, id 0
    Section number: 1
  ▸ Interface id: 0 (eth1)
    Encapsulation type: Ethernet (1)
    Arrival Time: Dec 12, 2025 11:06:49.209888648 GMT
    UTC Arrival Time: Dec 12, 2025 11:06:49.209888648 UTC
    Epoch Arrival Time: 1765537609.209888648
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.000220269 seconds]
    [Time delta from previous displayed frame: 0.000220269 seconds]
    [Time since reference or first frame: 9.120074764 seconds]
    Frame Number: 20
    Frame Length: 98 bytes (784 bits)
    Capture Length: 98 bytes (784 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:icmp:data]
    [Coloring Rule Name: ICMP]
    [Coloring Rule String: icmp || icmpv6]
▾ Ethernet II, Src: VMware_f7:5c:5a (00:0c:29:f7:5c:5a), Dst: VMware_f2:2d:bf (00:0c:29:f2:2d:bf)
  ▸ Destination: VMware_f2:2d:bf (00:0c:29:f2:2d:bf)
  ▸ Source: VMware_f7:5c:5a (00:0c:29:f7:5c:5a)
    Type: IPv4 (0x0800)
    [Stream index: 0]
▾ Internet Protocol Version 4, Src: 192.168.121.128, Dst: 192.168.121.133
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x383d (14397)

0000  00 0c 29 f2 2d bf 00 0c  29 f7 5c 5a 08 00 45 00   ··)·-··· )·\Z··E·
0010  00 54 38 3d 40 00 40 01  8e 15 c0 a8 79 80 c0 a8   ·T8=@·@· ····y···
0020  79 85 08 00 f4 60 00 07  00 31 49 f7 3b 69 00 00   y····`·· ·1I·;i··
0030  00 00 bc 33 03 00 00 00  00 00 10 11 12 13 14 15   ···3···· ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········ ·· !"#$%
```

No.: 20 · Time: 9.120074764 · Source: 192.168.121.128 · Destination: 192.168.121.133 · Protocol: ICMP · Length: 98 · Info: Echo (ping) request id=0x0007, seq=49/12544, ttl=64 (reply in 22)

✓ Show packet bytes   Layout: Vertical (Stacked)

6. Key Learnings

- Nmap can detect live hosts, services, versions, and OS fingerprints.
- SMB enumeration revealed possible anonymous access weaknesses.
- Scapy allows live packet sniffing and analyzing ICMP, DNS, ARP, and more.
- Using filters makes packet capture more targeted and efficient.
- Hands-on testing shows how attackers gather information early in a penetration test.

7. Conclusion

This lab successfully demonstrated the practical application of three essential network security tools: Nmap for reconnaissance and service enumeration, Scapy for packet crafting and analysis, and Wireshark for traffic inspection. Through hands-on exercises, we explored the complete lifecycle of network security assessment—from initial host discovery to detailed packet-level analysis. These tools form the foundation of both offensive security assessments and defensive network monitoring, highlighting the importance of understanding network protocols and traffic patterns in cybersecurity operations.

8. Annexes

- https://nmap.org/docs.html
- https://scapy.readthedocs.io/en/latest/
- https://www.wireshark.org/docs/