Rafael Angelo Christianto

NMAP PACKET CAPTURE LAB

Introduction

Nmap is used widely across the world by both white hat and black hat hackers. See exactly what Nmap does at packet level using Wireshark. In that way I will know how to spot if someone is scanning our network.

Ubuntu is widely used by companies to host their servers. So that is why I chose Ubuntu as well to simulate real world scenarios. I will be performing 6 tests.

1. SYN Scan (Half scan)
2. Full TCP connection scan
3. Scan single port
4. UDP scan
5. Capture using tcpdump
6. See Firewall Behavior

Methodology

Lab Setup:

- Attacker = Kali
- Target = Ubuntu
- Network = Internal Network

I will be putting the Wireshark capture on the target machine, as it will show me, how the OS will receive packets

Since I am doing this under an internal network, there will not be DHCP in this case, therefore I had to configure static IP addresses for both machines.

This is how I set it for Kali:

```
sudo ip addr add 192.168.100.20/24 dev eth0
sudo ip link set eth0 up
```

Then, this is how I set it up for Ubuntu:

```
nmcli connection show
sudo nmcli connection delete "Wired connection 1"
```

sudo nmcli connection add type ethernet ifname enp0s3 con-name lab-static ip4 192.168.100.10/24

sudo nmcli connection modify lab-static ipv4.method manual

sudo nmcli connection up lab-static

But for the best project scenario, I will open 2 ports, an OpenSSH server (port 22) and Metasploit (port 4444) by using netcat.

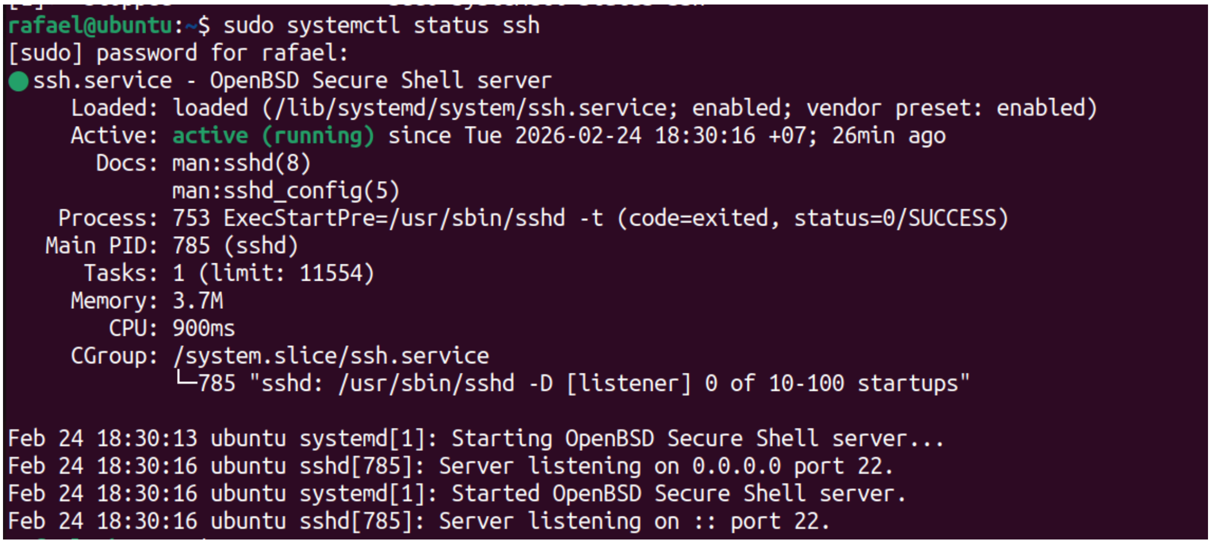
The setup for both is like these:

OpenSSH

sudo apt install openssh–server -y

sudo systemctl start ssh

sudo systemctl enable ssh

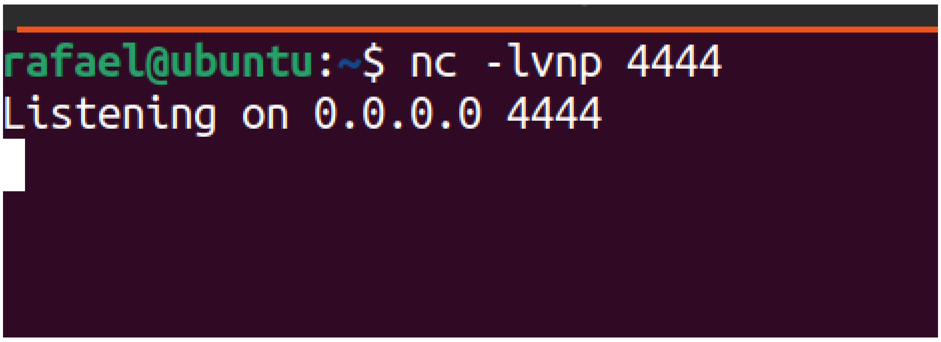


```
rafael@ubuntu:~$ sudo systemctl status ssh
[sudo] password for rafael:
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2026-02-24 18:30:16 +07; 26min ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 753 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 785 (sshd)
      Tasks: 1 (limit: 11554)
     Memory: 3.7M
        CPU: 900ms
     CGroup: /system.slice/ssh.service
             └─785 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Feb 24 18:30:13 ubuntu systemd[1]: Starting OpenBSD Secure Shell server...
Feb 24 18:30:16 ubuntu sshd[785]: Server listening on 0.0.0.0 port 22.
Feb 24 18:30:16 ubuntu systemd[1]: Started OpenBSD Secure Shell server.
Feb 24 18:30:16 ubuntu sshd[785]: Server listening on :: port 22.
```

Metasploit

sudo apt install netcat -y

nc -lvnp 4444



```
rafael@ubuntu:~$ nc -lvnp 4444
Listening on 0.0.0.0 4444
```

Results

First I have to install Wireshark to Ubuntu. And add the user to the Wireshark group to enable network packet capturing. This will be done by using the following 3 commands:

sudo apt update

sudo apt install wireshark -y

sudo usermod -aG wireshark $USER (for enabling to capture network packets)

Then make sure the Kali Linux VM and Ubuntu VM are connected through an Internal Network, and for my case the IPs are as follows:

- Kali Linux - 192.168.100.20
- Ubuntu - 192.168.100.10

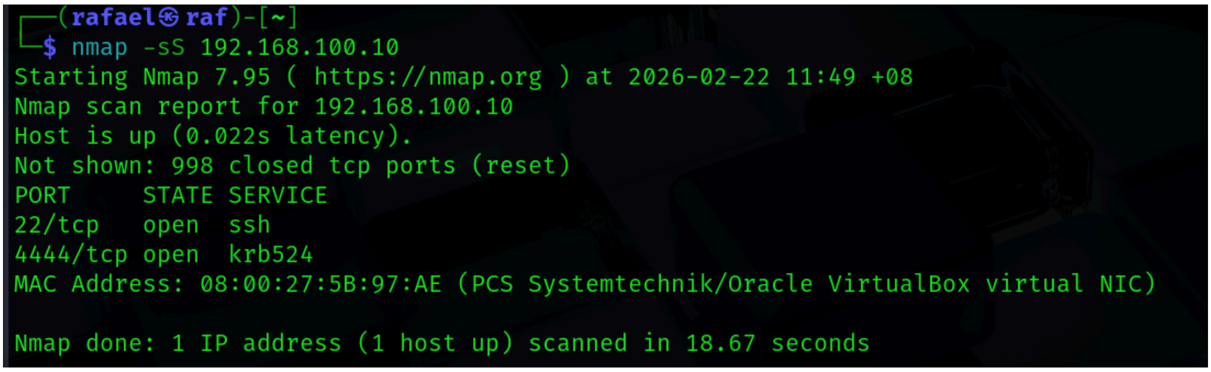
On Wireshark on the Ubuntu Desktop VM, I will capture from the enp0s3 adapter, since Ubuntu will be receiving from that.

Experiment 1 — SYN Scan (-sS)

First I will do the SYN Scan by using the nmap -sS flag, like the command shown below:

nmap -sS 192.168.100.10

From the screenshot below, we can see that 2 ports are open.

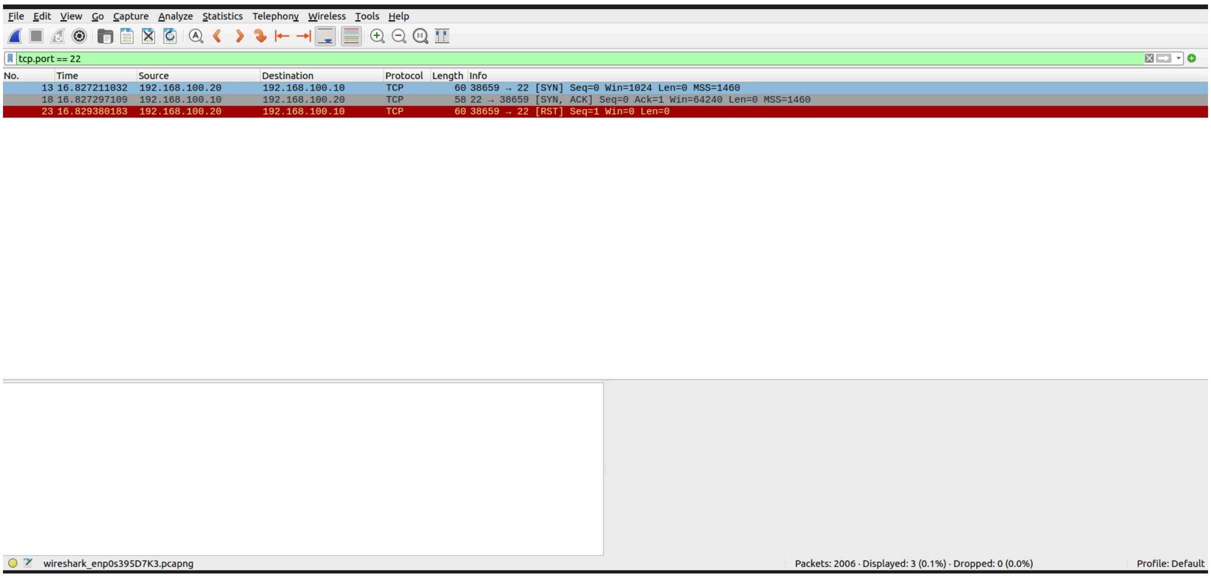


```
┌──(rafael㉿raf)-[~]
└─$ nmap -sS 192.168.100.10
Starting Nmap 7.95 ( https://nmap.org ) at 2026-02-22 11:49 +08
Nmap scan report for 192.168.100.10
Host is up (0.022s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
4444/tcp open  krb524
MAC Address: 08:00:27:5B:97:AE (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 18.67 seconds
```

On Ubuntu's Wireshark filter will be like below, since port 22 is one of the ports open:

tcp.port == 22

So here is the PCAP capture below, we can see that the Kali is sending SYN requests, but Ubuntu will SYN, ACK it back. But the Kali will give RST, terminating the 3 way handshake.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

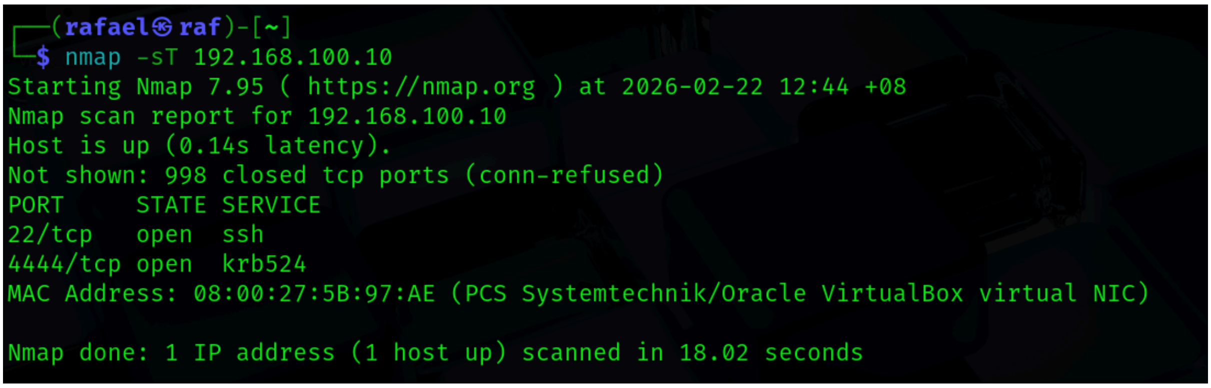
```
tcp.port == 22

No.   Time          Source           Destination      Protocol Length Info
   13 16.827211032  192.168.100.20   192.168.100.10   TCP        60 38659 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   18 16.827297109  192.168.100.10   192.168.100.20   TCP        58 22 → 38659 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
   23 16.829380183  192.168.100.20   192.168.100.10   TCP        60 38659 → 22 [RST] Seq=1 Win=0 Len=0

wireshark_enp0s395D7K3.pcapng        Packets: 2006 · Displayed: 3 (0.1%) · Dropped: 0 (0.0%)        Profile: Default
```

So the key takeaway is that, if there are a lot of uncompleted TCP handshakes, which could mean SYN > SYN/ACK > RST, like this example. If there were to be a port closed it is going to be SYN > RST.

Experiment 2 — Full TCP Connect Scan (-sT)

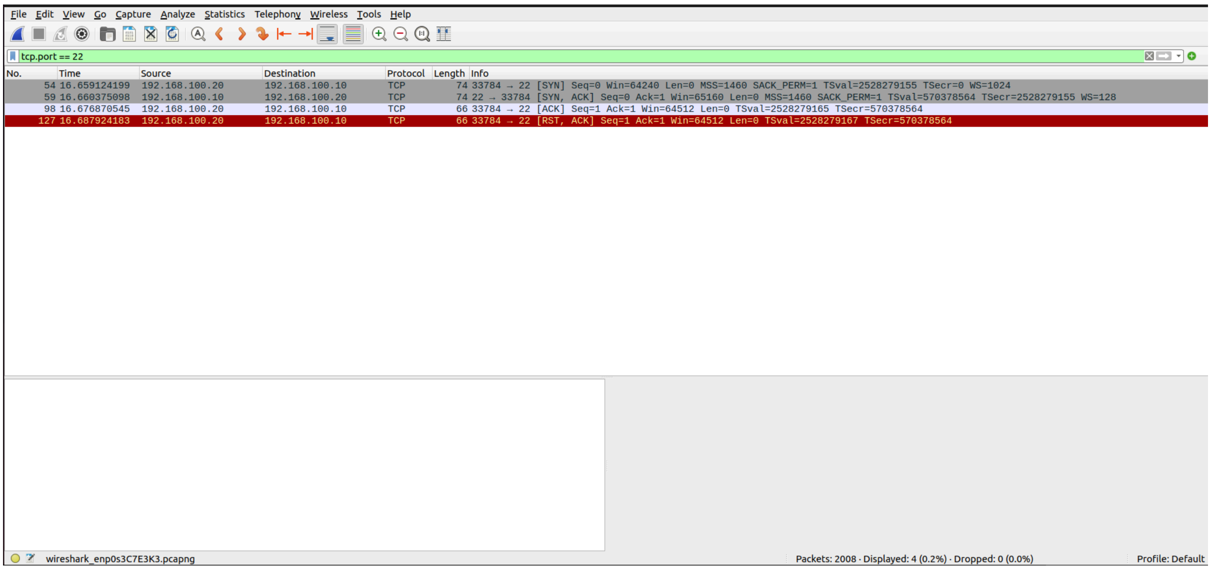
I will do the full TCP connection scan this time with the -sT flag, so it will go like this.

nmap -sT 192.168.100.10



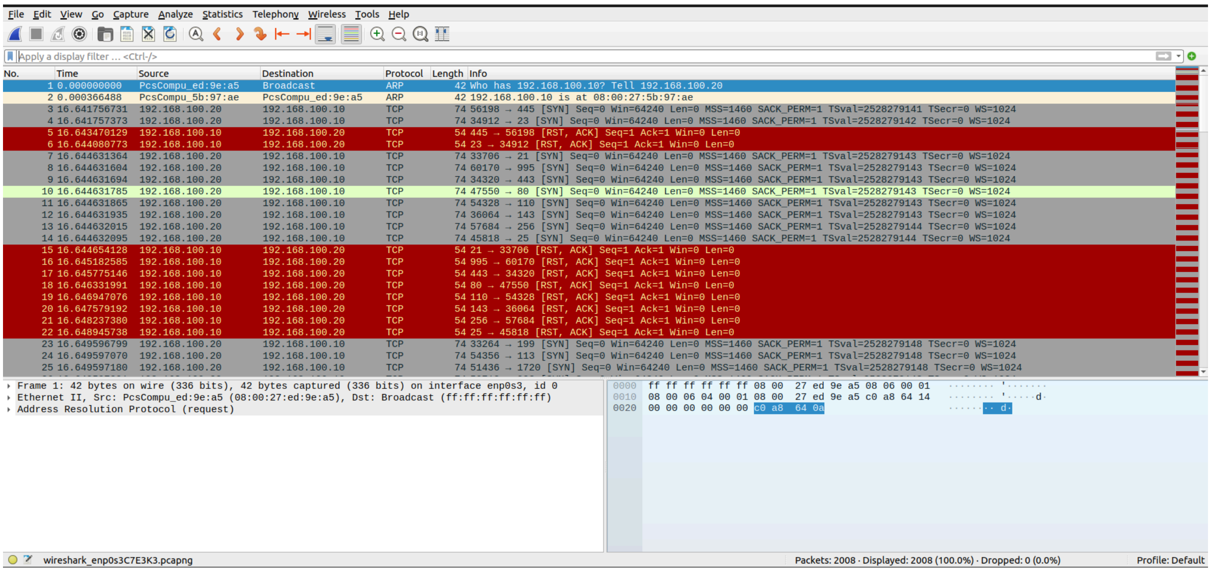
```
┌──(rafael㉿raf)-[~]
└─$ nmap -sT 192.168.100.10
Starting Nmap 7.95 ( https://nmap.org ) at 2026-02-22 12:44 +08
Nmap scan report for 192.168.100.10
Host is up (0.14s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT     STATE SERVICE
22/tcp   open  ssh
4444/tcp open  krb524
MAC Address: 08:00:27:5B:97:AE (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 18.02 seconds
```

On the Ubuntu machine, I will just filter with tcp.port==22 again. So here, we can see a fully completed 3 way handshake (SYN > SYN ACK > SYN). However, after that we can see Kali giving RST, which means after completing the 3 way handshake, it will reset the connection.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
tcp.port == 22

No.   Time          Source           Destination      Protocol Length Info
   54 16.659124199  192.168.100.20   192.168.100.10   TCP        74 33784 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2528279155 TSecr=0 WS=1024
   59 16.660375098  192.168.100.10   192.168.100.20   TCP        74 22 → 33784 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=570378564 TSecr=2528279155 WS=128
   98 16.676870545  192.168.100.20   192.168.100.10   TCP        66 33784 → 22 [ACK] Seq=1 Ack=1 Win=64512 Len=0 TSval=2528279165 TSecr=570378564
  127 16.687924183  192.168.100.20   192.168.100.10   TCP        66 33784 → 22 [RST, ACK] Seq=1 Ack=1 Win=64512 Len=0 TSval=2528279167 TSecr=570378564

wireshark_enp0s3C7E3K3.pcapng      Packets: 2008 · Displayed: 4 (0.2%) · Dropped: 0 (0.0%)      Profile: Default
```

For the rest of the ports, which are closed, they just give SYN > RST ACK connections, like shown in the picture below.

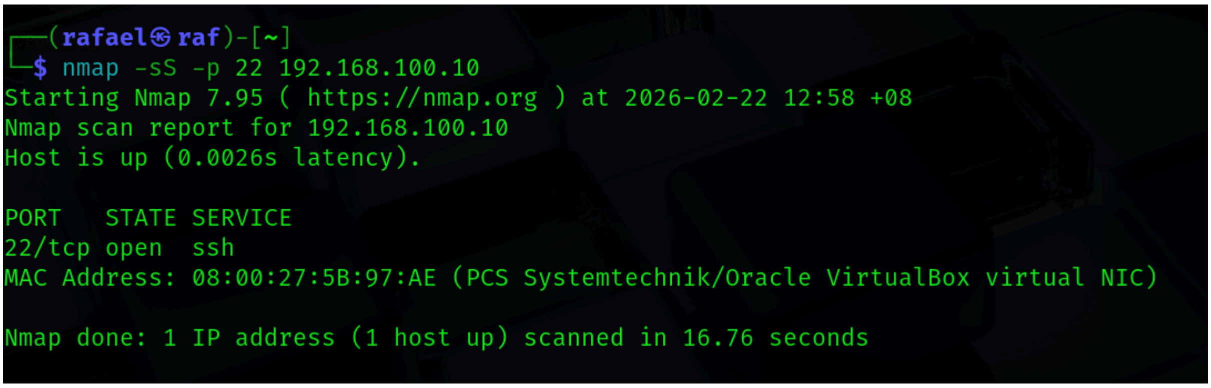

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
Apply a display filter ... <Ctrl-/>

No.   Time          Source              Destination         Protocol Length Info
    1 0.000000000   PcsCompu_ed:9e:a5   Broadcast           ARP        42 Who has 192.168.100.10? Tell 192.168.100.20
    2 0.000366488   PcsCompu_5b:97:ae   PcsCompu_ed:9e:a5   ARP        42 192.168.100.10 is at 08:00:27:5b:97:ae
    3 16.641756731  192.168.100.20      192.168.100.10      TCP        74 56198 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2528279141 TSecr=0 WS=1024
    4 16.641757373  192.168.100.20      192.168.100.10      TCP        74 34912 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2528279142 TSecr=0 WS=1024
    5 16.643470129  192.168.100.10      192.168.100.20      TCP        54 445 → 56198 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
    6 16.644080773  192.168.100.10      192.168.100.20      TCP        54 23 → 34912 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
    7 16.644631364  192.168.100.20      192.168.100.10      TCP        74 33706 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2528279143 TSecr=0 WS=1024
    8 16.644631604  192.168.100.20      192.168.100.10      TCP        74 60170 → 995 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2528279143 TSecr=0 WS=1024
    9 16.644631694  192.168.100.20      192.168.100.10      TCP        74 34320 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2528279143 TSecr=0 WS=1024
   10 16.644631785  192.168.100.20      192.168.100.10      TCP        74 47550 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2528279143 TSecr=0 WS=1024
   11 16.644631865  192.168.100.20      192.168.100.10      TCP        74 54328 → 110 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2528279143 TSecr=0 WS=1024
   12 16.644631935  192.168.100.20      192.168.100.10      TCP        74 36064 → 143 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2528279143 TSecr=0 WS=1024
   13 16.644632015  192.168.100.20      192.168.100.10      TCP        74 57684 → 256 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2528279144 TSecr=0 WS=1024
   14 16.644632095  192.168.100.20      192.168.100.10      TCP        74 45818 → 25 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2528279144 TSecr=0 WS=1024
   15 16.644654128  192.168.100.10      192.168.100.20      TCP        54 21 → 33706 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   16 16.645182585  192.168.100.10      192.168.100.20      TCP        54 995 → 60170 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   17 16.645775146  192.168.100.10      192.168.100.20      TCP        54 443 → 34320 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   18 16.646331991  192.168.100.10      192.168.100.20      TCP        54 80 → 47550 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   19 16.646947076  192.168.100.10      192.168.100.20      TCP        54 110 → 54328 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   20 16.647579192  192.168.100.10      192.168.100.20      TCP        54 143 → 36064 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   21 16.648237380  192.168.100.10      192.168.100.20      TCP        54 256 → 57684 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   22 16.648945738  192.168.100.10      192.168.100.20      TCP        54 25 → 45818 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
   23 16.649596799  192.168.100.20      192.168.100.10      TCP        74 33264 → 199 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2528279148 TSecr=0 WS=1024
   24 16.649597070  192.168.100.20      192.168.100.10      TCP        74 54356 → 113 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2528279148 TSecr=0 WS=1024
   25 16.649597180  192.168.100.20      192.168.100.10      TCP        74 51436 → 1720 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2528279148 TSecr=0 WS=1024

› Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface enp0s3, id 0
› Ethernet II, Src: PcsCompu_ed:9e:a5 (08:00:27:ed:9e:a5), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
› Address Resolution Protocol (request)

0000  ff ff ff ff ff ff 08 00  27 ed 9e a5 08 06 00 01   ········ '·······
0010  08 00 06 04 00 01 08 00  27 ed 9e a5 c0 a8 64 14   ········ '·····d·
0020  00 00 00 00 00 00 c0 a8  64 0a                     ········ d·

wireshark_enp0s3C7E3K3.pcapng      Packets: 2008 · Displayed: 2008 (100.0%) · Dropped: 0 (0.0%)      Profile: Default
```

Experiment 3 — Scan Single Port

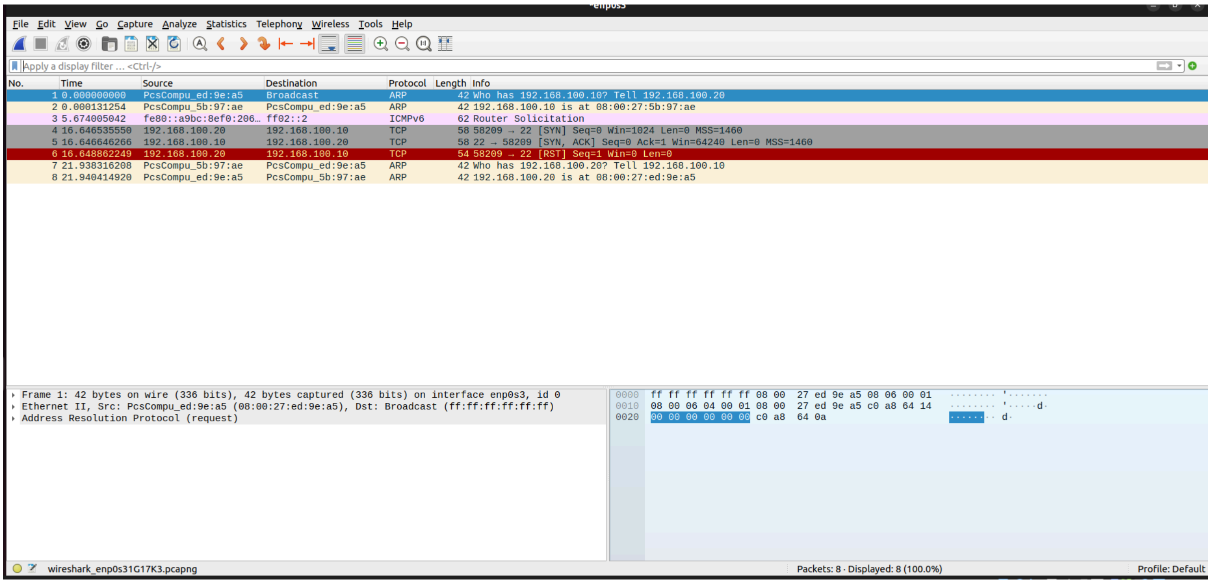
I will perform a one port scan only, since some attackers do just scan one port, popular ones such as port 22 (ssh), port 80 (http), port 3389 (RDP). This time since one of the ports is port 22, I will use it again. The command is as follows:

```
nmap -sS -p 22 192.168.100.10
```



```
┌──(rafael㉿raf)-[~]
└─$ nmap -sS -p 22 192.168.100.10
Starting Nmap 7.95 ( https://nmap.org ) at 2026-02-22 12:58 +08
Nmap scan report for 192.168.100.10
Host is up (0.0026s latency).

PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 08:00:27:5B:97:AE (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 16.76 seconds
```

In the Ubuntu machine, specifically the Wireshark, this time we do not have to provide any filters, since I only scanned 1 port. And in total I only got 8 packets. We can see though now more clearly. It begins with an ARP request, and then Kali will give a SYN and then Ubuntu responds with SYN ACK, and as expected since I also used the -sS flag here, Kali RST (resetted) the connection. The picture is given below.



*enp0s3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.  Time          Source               Destination        Protocol Length Info
   1 0.000000000   PcsCompu_ed:9e:a5    Broadcast          ARP        42 Who has 192.168.100.10? Tell 192.168.100.20
   2 0.000131254   PcsCompu_5b:97:ae    PcsCompu_ed:9e:a5  ARP        42 192.168.100.10 is at 08:00:27:5b:97:ae
   3 5.674005042   fe80::a9bc:8ef0:206… ff02::2            ICMPv6     62 Router Solicitation
   4 16.646535550  192.168.100.20       192.168.100.10     TCP        58 58209 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   5 16.646646266  192.168.100.10       192.168.100.20     TCP        58 22 → 58209 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
   6 16.648862249  192.168.100.20       192.168.100.10     TCP        54 58209 → 22 [RST] Seq=1 Win=0 Len=0
   7 21.938316208  PcsCompu_5b:97:ae    PcsCompu_ed:9e:a5  ARP        42 Who has 192.168.100.20? Tell 192.168.100.10
   8 21.940414920  PcsCompu_ed:9e:a5    PcsCompu_5b:97:ae  ARP        42 192.168.100.20 is at 08:00:27:ed:9e:a5
```

```
› Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface enp0s3, id 0
› Ethernet II, Src: PcsCompu_ed:9e:a5 (08:00:27:ed:9e:a5), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
› Address Resolution Protocol (request)
```

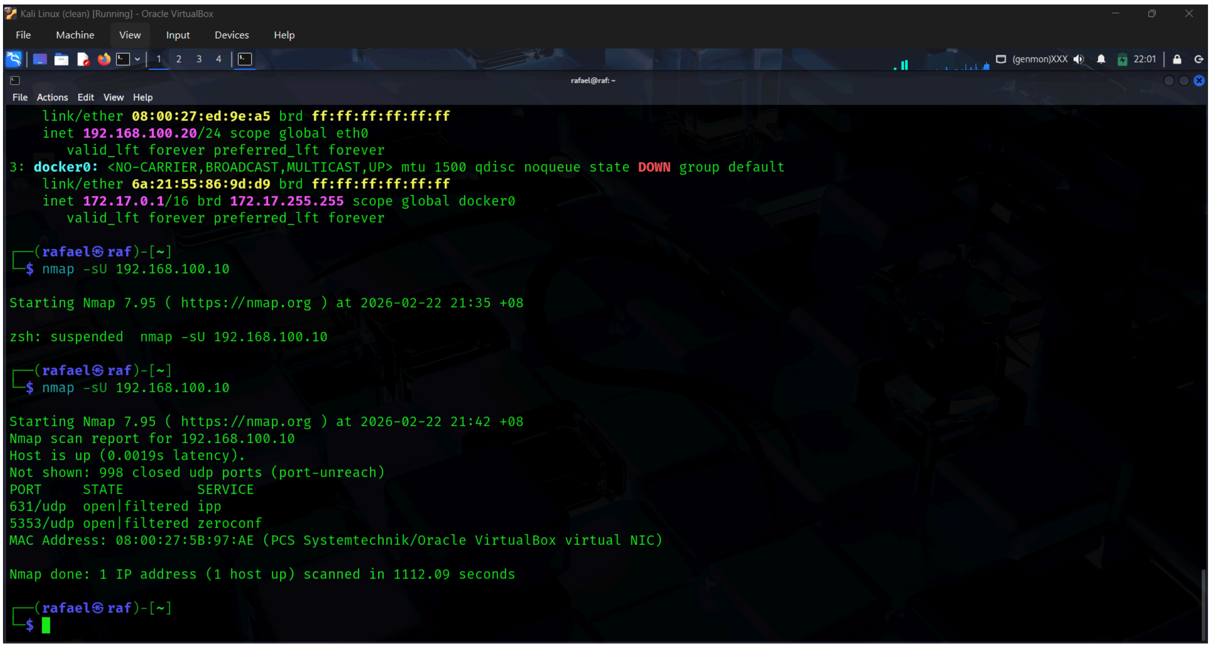
```
0000  ff ff ff ff ff ff 08 00  27 ed 9e a5 08 06 00 01   ········ '·······
0010  08 00 06 04 00 01 08 00  27 ed 9e a5 c0 a8 64 14   ········ '·····d·
0020  00 00 00 00 00 00 c0 a8  64 0a                     ········ d·
```

wireshark_enp0s31G17K3.pcapng
Packets: 8 · Displayed: 8 (100.0%)
Profile: Default

Experiment 4 — UDP Scan (-sU)

On my Kali machine I will do the UDP nmap scan by using the -sU flag. The command will be shown like follows:

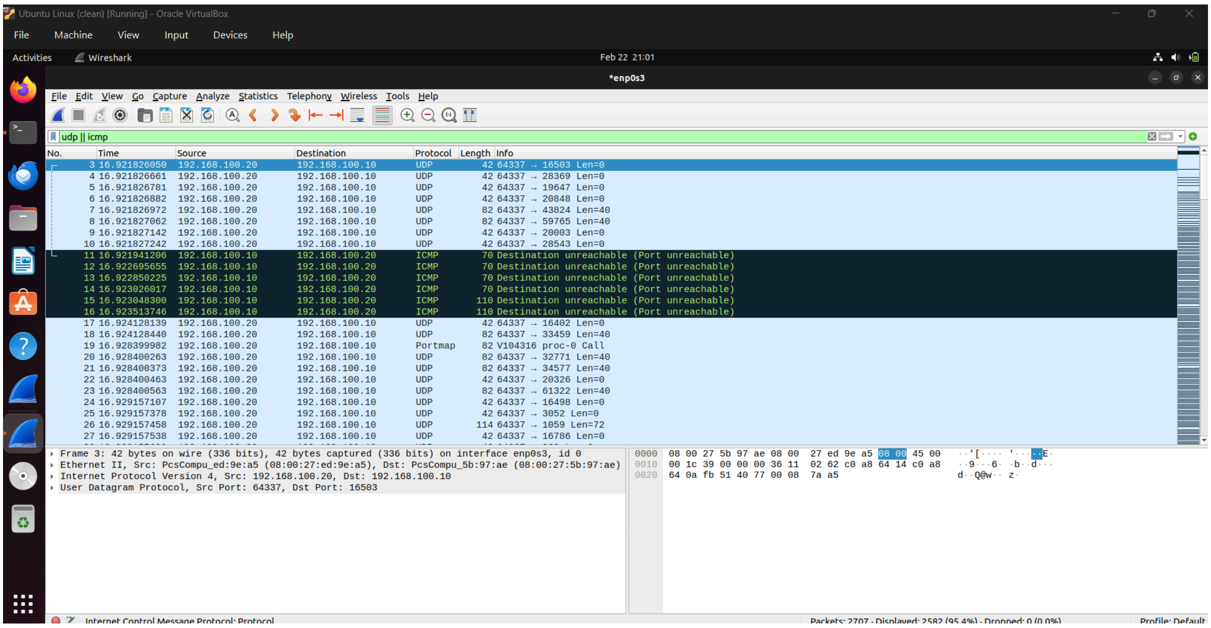
nmap -sU 192.168,100.10


Kali Linux (clean) [Running] - Oracle VirtualBox
File Machine View Input Devices Help
rafael@raf: ~
File Actions Edit View Help

```
    link/ether 08:00:27:ed:9e:a5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.20/24 scope global eth0
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 6a:21:55:86:9d:d9 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever

┌──(rafael㉿raf)-[~]
└─$ nmap -sU 192.168.100.10

Starting Nmap 7.95 ( https://nmap.org ) at 2026-02-22 21:35 +08

zsh: suspended  nmap -sU 192.168.100.10

┌──(rafael㉿raf)-[~]
└─$ nmap -sU 192.168.100.10

Starting Nmap 7.95 ( https://nmap.org ) at 2026-02-22 21:42 +08
Nmap scan report for 192.168.100.10
Host is up (0.0019s latency).
Not shown: 998 closed udp ports (port-unreach)
PORT      STATE         SERVICE
631/udp   open|filtered ipp
5353/udp  open|filtered zeroconf
MAC Address: 08:00:27:5B:97:AE (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 1112.09 seconds

┌──(rafael㉿raf)-[~]
└─$ 
```

On the other hand, in the Ubuntu machine, this time since it is a UDP scan I cannot do tcp filter instead I will filter for UDP or ICMP packets like shown below:

udp || icmp


Ubuntu Linux (clean) [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Activities
Wireshark
Feb 22 21:01
*enp0s3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

udp || icmp

```
No.  Time          Source          Destination     Protocol Length Info
   3 16.921826050  192.168.100.20  192.168.100.10  UDP        42 64337 → 16503 Len=0
   4 16.921826661  192.168.100.20  192.168.100.10  UDP        42 64337 → 28369 Len=0
   5 16.921826781  192.168.100.20  192.168.100.10  UDP        42 64337 → 19647 Len=0
   6 16.921826882  192.168.100.20  192.168.100.10  UDP        42 64337 → 20848 Len=0
   7 16.921826972  192.168.100.20  192.168.100.10  UDP        82 64337 → 43824 Len=40
   8 16.921827062  192.168.100.20  192.168.100.10  UDP        82 64337 → 59765 Len=40
   9 16.921827142  192.168.100.20  192.168.100.10  UDP        42 64337 → 20003 Len=0
  10 16.921827242  192.168.100.20  192.168.100.10  UDP        42 64337 → 28543 Len=0
  11 16.921941206  192.168.100.10  192.168.100.20  ICMP       70 Destination unreachable (Port unreachable)
  12 16.922695655  192.168.100.10  192.168.100.20  ICMP       70 Destination unreachable (Port unreachable)
  13 16.922850225  192.168.100.10  192.168.100.20  ICMP       70 Destination unreachable (Port unreachable)
  14 16.923026017  192.168.100.10  192.168.100.20  ICMP       70 Destination unreachable (Port unreachable)
  15 16.923048300  192.168.100.10  192.168.100.20  ICMP      110 Destination unreachable (Port unreachable)
  16 16.923513746  192.168.100.10  192.168.100.20  ICMP      110 Destination unreachable (Port unreachable)
  17 16.924128139  192.168.100.20  192.168.100.10  UDP        42 64337 → 16402 Len=0
  18 16.924128440  192.168.100.20  192.168.100.10  UDP        82 64337 → 33459 Len=40
  19 16.928399982  192.168.100.20  192.168.100.10  Portmap    82 V104316 proc-0 Call
  20 16.928400263  192.168.100.20  192.168.100.10  UDP        82 64337 → 32771 Len=40
  21 16.928400373  192.168.100.20  192.168.100.10  UDP        82 64337 → 34577 Len=40
  22 16.928400463  192.168.100.20  192.168.100.10  UDP        42 64337 → 20326 Len=0
  23 16.928400563  192.168.100.20  192.168.100.10  UDP        82 64337 → 61322 Len=40
  24 16.929157107  192.168.100.20  192.168.100.10  UDP        42 64337 → 16498 Len=0
  25 16.929157378  192.168.100.20  192.168.100.10  UDP        42 64337 → 3052 Len=0
  26 16.929157458  192.168.100.20  192.168.100.10  UDP       114 64337 → 1059 Len=72
  27 16.929157538  192.168.100.20  192.168.100.10  UDP        42 64337 → 16786 Len=0
```

```
› Frame 3: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface enp0s3, id 0
› Ethernet II, Src: PcsCompu_ed:9e:a5 (08:00:27:ed:9e:a5), Dst: PcsCompu_5b:97:ae (08:00:27:5b:97:ae)
› Internet Protocol Version 4, Src: 192.168.100.20, Dst: 192.168.100.10
› User Datagram Protocol, Src Port: 64337, Dst Port: 16503
```

```
0000  08 00 27 5b 97 ae 08 00  27 ed 9e a5 08 00 45 00   ··'[···· '·····E·
0010  00 1c 39 00 00 00 36 11  02 62 c0 a8 64 14 c0 a8   ··9···6· ·b··d···
0020  64 0a fb 51 40 77 00 08  7a a5                     d··Q@w·· z·
```

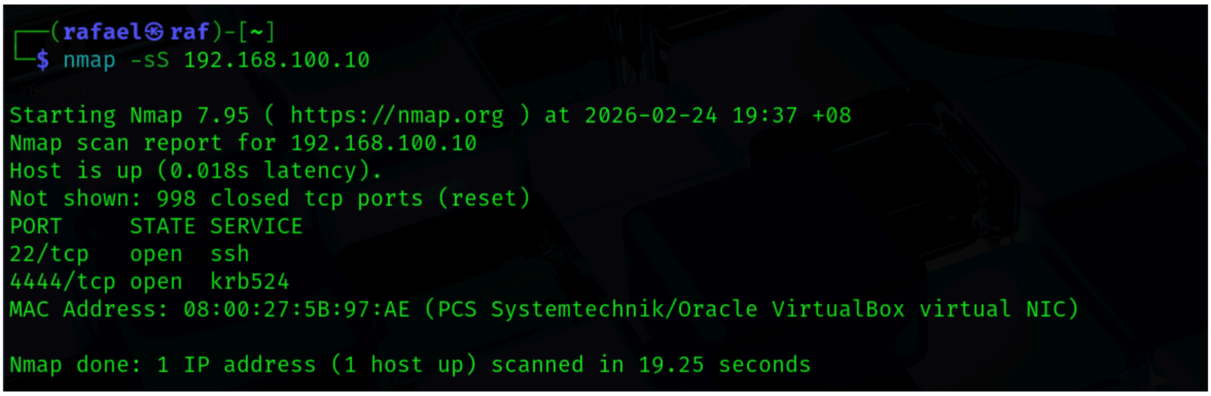
```
Internet Control Message Protocol: Protocol          Packets: 2707 · Displayed: 2582 (95.4%) · Dropped: 0 (0.0%)          Profile: Default
```

From the screenshot above, we can conclude that there are a lot of ICMP Destination unreachable or port unreachable. So from this we get the insight that, if there are a high ICMP port unreachable response, it is possible to be a UDP scan.

Experiment 5 — Capture with tcpdump (CLI Monitoring)

On Kali, I will just do a SYN scan, like the command shown below:

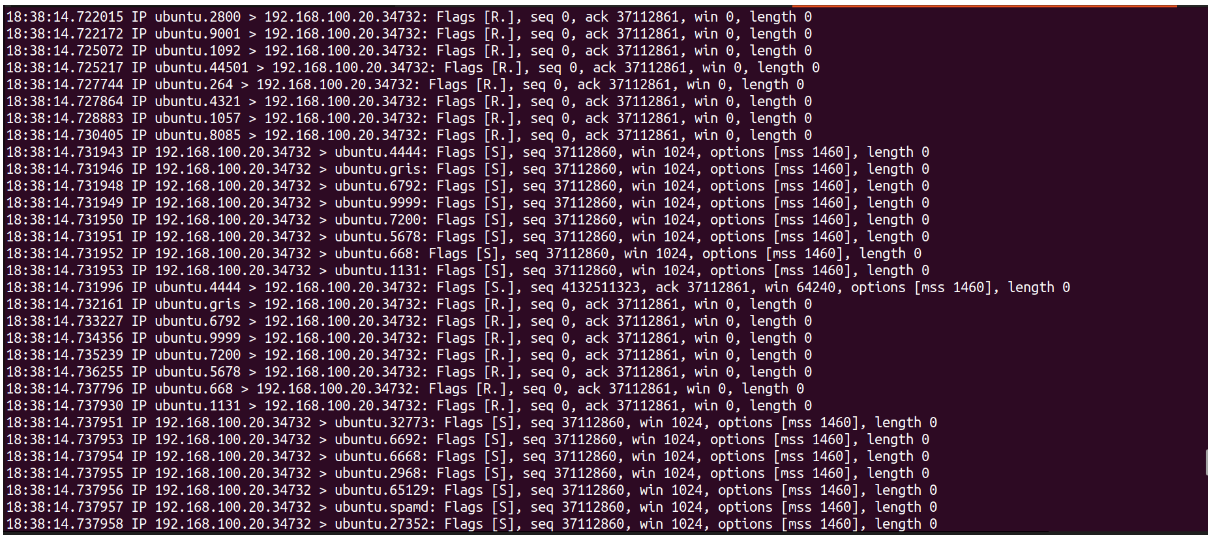
nmap -sS 192.168.100.10



```
┌──(rafael㉿raf)-[~]
└─$ nmap -sS 192.168.100.10

Starting Nmap 7.95 ( https://nmap.org ) at 2026-02-24 19:37 +08
Nmap scan report for 192.168.100.10
Host is up (0.018s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
4444/tcp open  krb524
MAC Address: 08:00:27:5B:97:AE (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 19.25 seconds
```

This time, instead of GUI applications like Wireshark, this time we will use tcpdump, a CLI tool also used to collect packets.

sudo tcpdump -i enp0s3 host 192.168.100.20



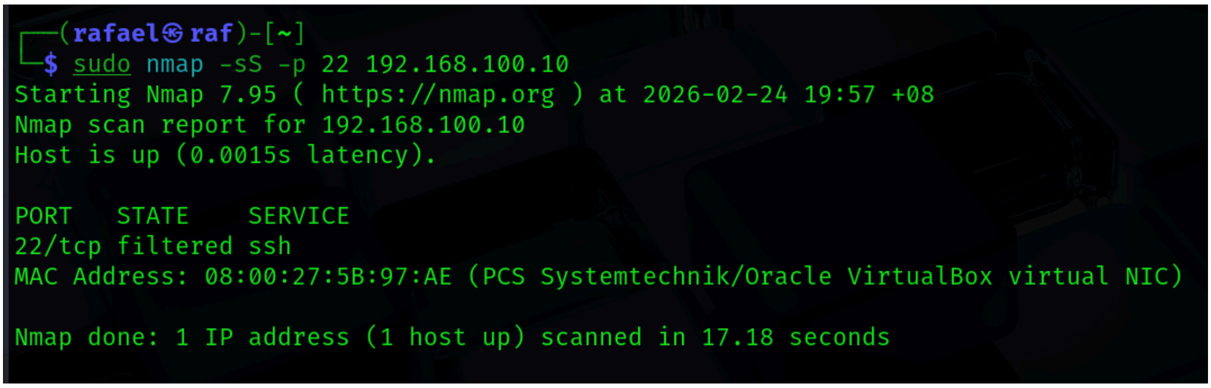
```
18:38:14.722015 IP ubuntu.2800 > 192.168.100.20.34732: Flags [R.], seq 0, ack 37112861, win 0, length 0
18:38:14.722172 IP ubuntu.9001 > 192.168.100.20.34732: Flags [R.], seq 0, ack 37112861, win 0, length 0
18:38:14.725072 IP ubuntu.1092 > 192.168.100.20.34732: Flags [R.], seq 0, ack 37112861, win 0, length 0
18:38:14.725217 IP ubuntu.44501 > 192.168.100.20.34732: Flags [R.], seq 0, ack 37112861, win 0, length 0
18:38:14.727744 IP ubuntu.264 > 192.168.100.20.34732: Flags [R.], seq 0, ack 37112861, win 0, length 0
18:38:14.727864 IP ubuntu.4321 > 192.168.100.20.34732: Flags [R.], seq 0, ack 37112861, win 0, length 0
18:38:14.728883 IP ubuntu.1057 > 192.168.100.20.34732: Flags [R.], seq 0, ack 37112861, win 0, length 0
18:38:14.730405 IP ubuntu.8085 > 192.168.100.20.34732: Flags [R.], seq 0, ack 37112861, win 0, length 0
18:38:14.731943 IP 192.168.100.20.34732 > ubuntu.4444: Flags [S], seq 37112860, win 1024, options [mss 1460], length 0
18:38:14.731946 IP 192.168.100.20.34732 > ubuntu.gris: Flags [S], seq 37112860, win 1024, options [mss 1460], length 0
18:38:14.731948 IP 192.168.100.20.34732 > ubuntu.6792: Flags [S], seq 37112860, win 1024, options [mss 1460], length 0
18:38:14.731949 IP 192.168.100.20.34732 > ubuntu.9999: Flags [S], seq 37112860, win 1024, options [mss 1460], length 0
18:38:14.731950 IP 192.168.100.20.34732 > ubuntu.7200: Flags [S], seq 37112860, win 1024, options [mss 1460], length 0
18:38:14.731951 IP 192.168.100.20.34732 > ubuntu.5678: Flags [S], seq 37112860, win 1024, options [mss 1460], length 0
18:38:14.731952 IP 192.168.100.20.34732 > ubuntu.668: Flags [S], seq 37112860, win 1024, options [mss 1460], length 0
18:38:14.731953 IP 192.168.100.20.34732 > ubuntu.1131: Flags [S], seq 37112860, win 1024, options [mss 1460], length 0
18:38:14.731996 IP ubuntu.4444 > 192.168.100.20.34732: Flags [S.], seq 4132511323, ack 37112861, win 64240, options [mss 1460], length 0
18:38:14.732161 IP ubuntu.gris > 192.168.100.20.34732: Flags [R.], seq 0, ack 37112861, win 0, length 0
18:38:14.733227 IP ubuntu.6792 > 192.168.100.20.34732: Flags [R.], seq 0, ack 37112861, win 0, length 0
18:38:14.734356 IP ubuntu.9999 > 192.168.100.20.34732: Flags [R.], seq 0, ack 37112861, win 0, length 0
18:38:14.735239 IP ubuntu.7200 > 192.168.100.20.34732: Flags [R.], seq 0, ack 37112861, win 0, length 0
18:38:14.736255 IP ubuntu.5678 > 192.168.100.20.34732: Flags [R.], seq 0, ack 37112861, win 0, length 0
18:38:14.737796 IP ubuntu.668 > 192.168.100.20.34732: Flags [R.], seq 0, ack 37112861, win 0, length 0
18:38:14.737930 IP ubuntu.1131 > 192.168.100.20.34732: Flags [R.], seq 0, ack 37112861, win 0, length 0
18:38:14.737951 IP 192.168.100.20.34732 > ubuntu.32773: Flags [S], seq 37112860, win 1024, options [mss 1460], length 0
18:38:14.737953 IP 192.168.100.20.34732 > ubuntu.6692: Flags [S], seq 37112860, win 1024, options [mss 1460], length 0
18:38:14.737954 IP 192.168.100.20.34732 > ubuntu.6668: Flags [S], seq 37112860, win 1024, options [mss 1460], length 0
18:38:14.737955 IP 192.168.100.20.34732 > ubuntu.2968: Flags [S], seq 37112860, win 1024, options [mss 1460], length 0
18:38:14.737956 IP 192.168.100.20.34732 > ubuntu.65129: Flags [S], seq 37112860, win 1024, options [mss 1460], length 0
18:38:14.737957 IP 192.168.100.20.34732 > ubuntu.spamd: Flags [S], seq 37112860, win 1024, options [mss 1460], length 0
18:38:14.737958 IP 192.168.100.20.34732 > ubuntu.27352: Flags [S], seq 37112860, win 1024, options [mss 1460], length 0
```

From what we can see if the ports are closed, after the SYN requests (the [S]), directly it will reset the connection (the [R]). However, we see that there is one line ubuntu.4444, where port 4444 is open in this case, it returns a "S." response, which is a SYN-ACK response, signifying that this is an open port.

EXPERIMENT 6 — Firewall Behavior (UFW)

This time wI will experiment a bit differently, since I will be using Firewalls this time UFW, which stands for uncomplicated firewall accessible through the Firewall. So I am going to test this on port 22. Therefore I will specifically only scan for port 22.

sudo nmap -sS -p 22 192.168.100.10



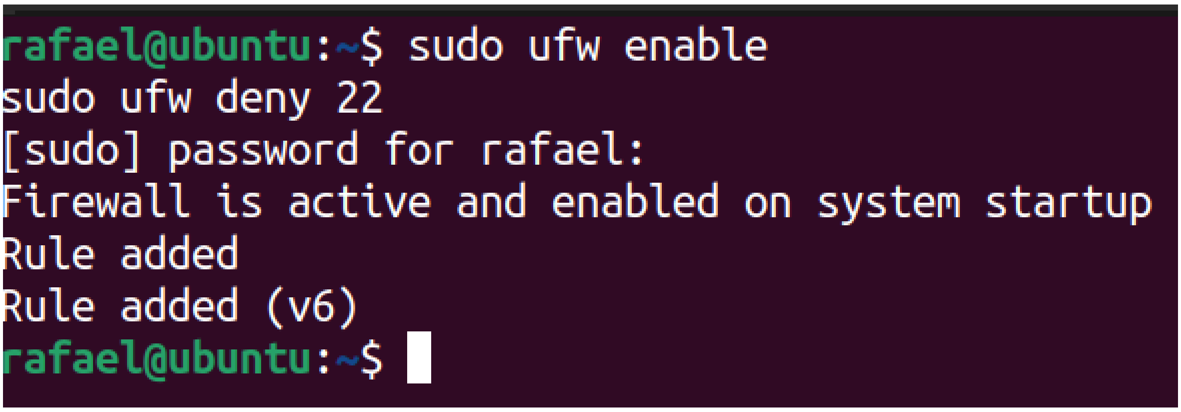
```
┌──(rafael㉿raf)-[~]
└─$ sudo nmap -sS -p 22 192.168.100.10
Starting Nmap 7.95 ( https://nmap.org ) at 2026-02-24 19:57 +08
Nmap scan report for 192.168.100.10
Host is up (0.0015s latency).

PORT   STATE    SERVICE
22/tcp filtered ssh
MAC Address: 08:00:27:5B:97:AE (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 17.18 seconds
```

However, as we can see from the results of the nmap scan, the port is filtered due to the firewall.

On Ubuntu we have to enable the UFW and deny any traffic coming to port 22

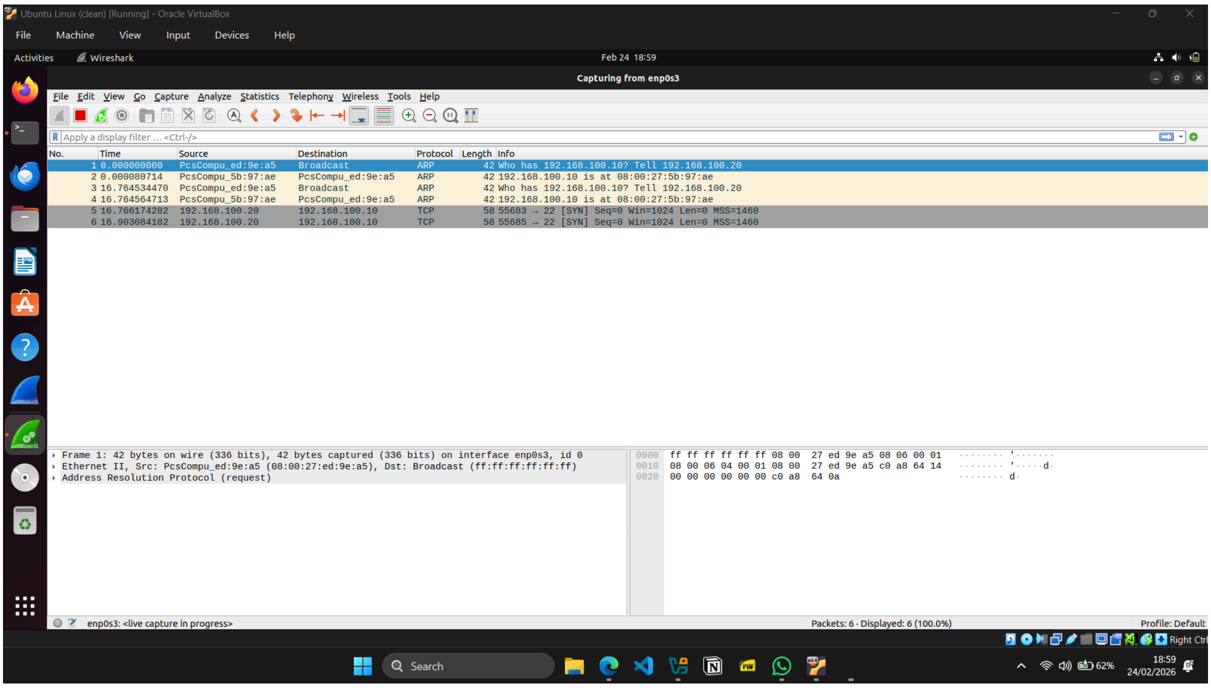
sudo ufw enable

sudo ufw deny 22



```
rafael@ubuntu:~$ sudo ufw enable
sudo ufw deny 22
[sudo] password for rafael:
Firewall is active and enabled on system startup
Rule added
Rule added (v6)
rafael@ubuntu:~$
```

And here is how it looks on Wireshark:


Ubuntu Linux (clean) [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Activities
Wireshark
Feb 24 18:59
Capturing from enp0s3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Profile: Default
Search
Right Ctrl
18:59
24/02/2026

```
No.  Time          Source              Destination         Protocol  Length Info
   1 0.000000000   PcsCompu_ed:9e:a5   Broadcast           ARP       42 Who has 192.168.100.10? Tell 192.168.100.20
   2 0.000080714   PcsCompu_5b:97:ae   PcsCompu_ed:9e:a5   ARP       42 192.168.100.10 is at 08:00:27:5b:97:ae
   3 16.764534470  PcsCompu_ed:9e:a5   Broadcast           ARP       42 Who has 192.168.100.10? Tell 192.168.100.20
   4 16.764564713  PcsCompu_5b:97:ae   PcsCompu_ed:9e:a5   ARP       42 192.168.100.10 is at 08:00:27:5b:97:ae
   5 16.766174282  192.168.100.20      192.168.100.10      TCP       58 55683 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
   6 16.903084182  192.168.100.20      192.168.100.10      TCP       58 55685 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
```

```
› Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface enp0s3, id 0
› Ethernet II, Src: PcsCompu_ed:9e:a5 (08:00:27:ed:9e:a5), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
› Address Resolution Protocol (request)
```

```
0000  ff ff ff ff ff ff 08 00  27 ed 9e a5 08 06 00 01   ········ '·······
0010  08 00 06 04 00 01 08 00  27 ed 9e a5 c0 a8 64 14   ········ '·····d·
0020  00 00 00 00 00 00 c0 a8  64 0a                     ········ d·
```

```
enp0s3: <live capture in progress>      Packets: 6 · Displayed: 6 (100.0%)
```

Kali just sent the SYN packets, but it got detected by the firewall, so firewall will just drop the packets.

Conclusion

Behavior	Packet Pattern
Port is open (-sS)	SYN > SYN-ACK > RST
Port is closed	SYN > RST
Filtered by Firewalll	SYN > no response
Full 3 way handshake(-sT)	SYN > SYN-ACK > ACK
UDP port closed (-sU)	ICMP/port unreachable

So therefore putting into statement, here is how to detect if traffic is suspicious:

- Many SYN packets coming to many ports and not getting SYN ACK response
- ICMP/Port unreachable packets