

Mapeando as máquinas da rede com NMAP

Agora, vamos iniciar um mapeamento da rede. Dessa forma, vamos usar o nmap para buscar as máquinas na rede e verificar os serviços disponíveis. Além disso, vale ressaltar que o nmap pode buscar por serviços que operam em portas UDP e TCP. Como exemplo vamos usar o comando abaixo para pesquisar hosts na rede 192.168.0.0/24.

nmap 192.168.0.0/24

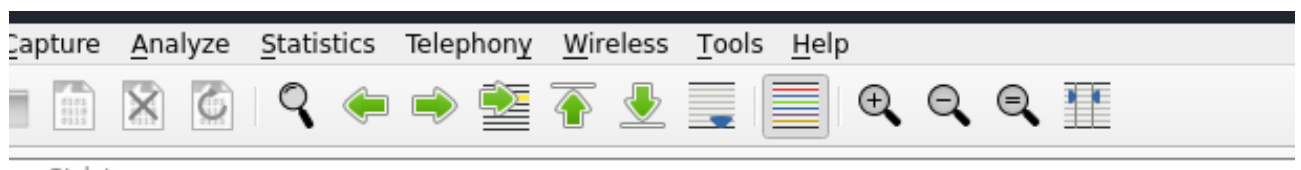
```
(teste@kali)-[~]
└─$ nmap 192.168.0.0/24
Starting Nmap 7.91 ( https://nmap.org ) at
Nmap scan report for 192.168.0.1
Host is up (0.0014s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
80/tcp    open  http
443/tcp   open  https
Nmap scan report for 192.168.0.111
Host is up (0.0014s latency).
All 1000 scanned ports on 192.168.0.111 are
```

nmap 192.168.0.0/24

Analizando o NMAP no Wireshark

Também vamos investigar a forma como o nmap descobre máquinas na rede. Para isso, vamos usar o wireshark.

O Wireshark possibilita a investigação de pacotes coletados a partir do tráfego que está sendo coletado da rede ou de um host específico. Existem outras funcionalidades disponíveis no wireshark, mas vamos falar disso em outro momento.



Source	Destination	Protocol	Length	Info
192.168.0.111	192.168.0.1	TCP	66	38406 → 23 [ACK] Seq=1 Ack=1
192.168.0.111	192.168.0.1	TCP	66	38406 → 23 [RST, ACK] Seq=1
192.168.0.111	192.168.0.1	TCP	74	45438 → 587 [SYN] Seq=0 Win=
192.168.0.1	192.168.0.111	TCP	60	587 → 45438 [RST, ACK] Seq=1
192.168.0.111	192.168.0.1	TCP	74	50252 → 443 [SYN] Seq=0 Win=
192.168.0.1	192.168.0.111	TCP	74	443 → 50252 [SYN, ACK] Seq=0
192.168.0.111	192.168.0.1	TCP	66	50252 → 443 [ACK] Seq=1 Ack=
192.168.0.111	192.168.0.1	TCP	74	53910 → 143 [SYN] Seq=0 Win=
192.168.0.1	192.168.0.111	TCP	60	143 → 53910 [RST, ACK] Seq=1
192.168.0.111	192.168.0.1	TCP	74	55012 → 110 [SYN] Seq=0 Win=
192.168.0.1	192.168.0.111	TCP	60	110 → 55012 [RST, ACK] Seq=1
192.168.0.111	192.168.0.1	TCP	74	49700 → 53 [SYN] Seq=0 Win=6
192.168.0.1	192.168.0.111	TCP	60	53 → 49700 [RST, ACK] Seq=1
192.168.0.111	192.168.0.1	TCP	74	44466 → 80 [SYN] Seq=0 Win=6
192.168.0.1	192.168.0.111	TCP	74	80 → 44466 [SYN, ACK] Seq=0
192.168.0.111	192.168.0.1	TCP	66	44466 → 80 [ACK] Seq=1 Ack=1
192.168.0.111	192.168.0.1	TCP	74	58046 → 1720 [SYN] Seq=0 Win=
192.168.0.1	192.168.0.111	TCP	60	1720 → 58046 [RST, ACK] Seq=
192.168.0.111	192.168.0.1	TCP	74	59850 → 995 [SYN] Seq=0 Win=
192.168.0.1	192.168.0.111	TCP	60	995 → 59850 [RST, ACK] Seq=1
192.168.0.111	192.168.0.1	TCP	74	58318 → 3389 [SYN] Seq=0 Win=
192.168.0.1	192.168.0.111	TCP	60	3389 → 58318 [RST, ACK] Seq=
192.168.0.111	192.168.0.1	TCP	74	50818 → 993 [SYN] Seq=0 Win=

nmap wireshark

NMAP descobrindo portas TCP abertas

O nmap usa pacotes SYN para a descoberta das portas TCP abertas, nessa modalidade que estamos usando. Dessa forma, são enviados SYN e aguarda-se a resposta SYN+ ACK e depois é enviado um ACK. Essa modalidade padrão de escaneamento do nmap faz a conexão inteira utilizando as 3 fases da conexão de 3 vias.

Quando o nmap envia um SYN e obtém uma resposta que dá continuidade a conexão, a ferramenta interpreta que a porta está aberta. No entanto, quando recebe um RST, o nmap interpreta que a porta está fechada. Isso porque, em ambientes normais um host vai responder com RST quando houver uma solicitação a uma porta que está fechada.

No entanto, vale ressaltar que podemos ter um equipamento ou um software filtrando as respostas de um determinado host. Dessa forma, ao invés de responder com um RST o host pode apenas descartar o pedido de

conexão. Nesse caso, a porta pode aparecer como filtrada.

Utilizaremos o Wireshark para investigar comportamentos relacionados ao NMAP em mapeamento de portas TCP e UDP.

Observando os pacotes SYN, SYN+ACK e RST

Podemos observar no Wireshark os pacotes SYN, SYN+ACK, ACK e RST. Aqui podemos ver o comportamento da ferramenta ao tentar verificar os serviços que são executados nas portas investigadas.

Escaneando um host coma opção -sT

Essa opção segue o escaneamento padrão do nmap, executando as 3 vias do TCP para detectar se uma porta está aberta.

nmap -sT 192.168.0.1

```

(teste@kali)-[~]
$ nmap -sT 192.168.0.1
Starting Nmap 7.91 ( https://nmap.org ) at
Nmap scan report for 192.168.0.1
Host is up (0.011s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
80/tcp    open  http
443/tcp   open  https
Nmap done: 1 IP address (1 host up) scanned

```

nmap -sT

NMAP: scan com a opção -sS

Outra forma de escaneamento de portas TCP é com a opção -sT. Nessa opção, o nmap envia um SYN e aguarda por um SYN+ACK. Depois disso o nmap fecha a conexão enviando um RST para o alvo. Dessa forma, o nmap não completa as 3 vias da conexão TCP. O Objetivo é fazer um scan mais rápido e até mais silencioso do que o utilizando a opção -sT.

sudo nmap -sS 192.168.0.1

```

(teste@kali)-[~]
$ sudo nmap -sS 192.168.0.1
[sudo] password for teste:
Starting Nmap 7.91 ( https://nmap.org ) at 2021-
Nmap scan report for 192.168.0.1
Host is up (0.0073s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
80/tcp    open  http
443/tcp   open  https
MAC Address: 08:00:27:4B:32:42 (Oracle VirtualBox)

```

nmap -sS

Capture	Analyze	Statistics	Telephony	Wireless	Tools	Help
... <Ctrl-/>						
Source	Destination	Protocol	Length	Info		
PcsCompu_19:d5:47	Broadcast	ARP	42	Who has 192.168.0.1? Tel		
PcsCompu_4b:32:42	PcsCompu_19:d5:47	ARP	60	192.168.0.1 is at 08:00:27:4b:32:42		
192.168.0.111	192.168.0.1	TCP	58	42990 → 21 [SYN] Seq=0 Win=0 Len=0		
192.168.0.1	192.168.0.111	TCP	60	21 → 42990 [SYN, ACK] Seq=42990 Win=65535 Len=0		
192.168.0.111	192.168.0.1	TCP	54	42990 → 21 [RST] Seq=1 Win=0 Len=0		
PcsCompu_4b:32:42	PcsCompu_19:d5:47	ARP	60	Who has 192.168.0.111? Tel		
PcsCompu_19:d5:47	PcsCompu_4b:32:42	ARP	42	192.168.0.111 is at 08:00:27:4b:32:42		

wireshark nmap -sS

NMAP direcionado a uma porta -p

O nmap também permite que seja especificado uma porta para ser investigada. Para isso, usamos a opção -p e a porta que desejamos verificar. Essa opção é interessante quando desejamos verificar um serviço específico em um host. Nesse

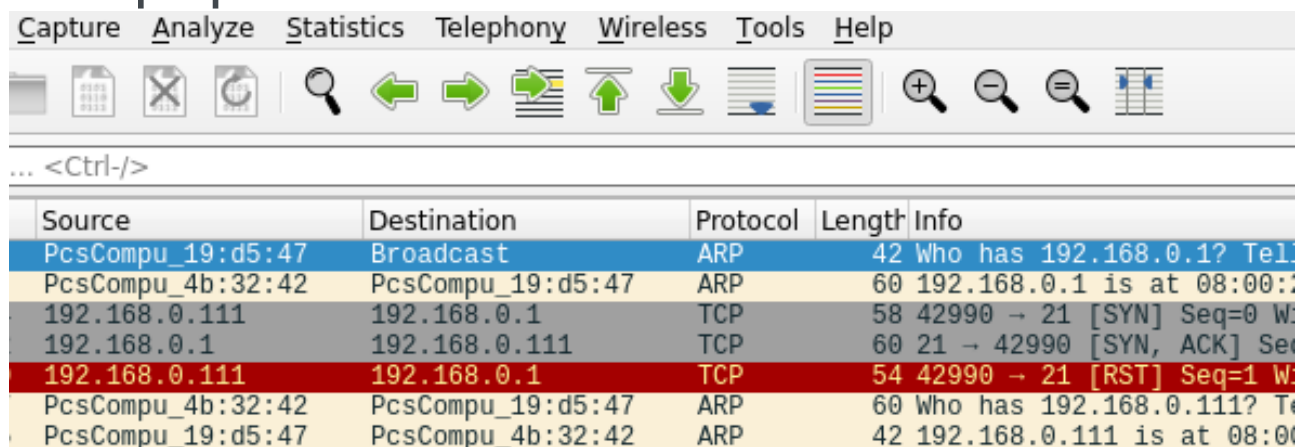
exemplo vamos escanear a porta 21 relacionada ao protocolo FTP.

sudo nmap -sS 192.168.0.1 -p 21

```
(teste@kali)-[~]
$ sudo nmap -sS 192.168.0.1 -p 21
Starting Nmap 7.91 ( https://nmap.org )
Nmap scan report for 192.168.0.1
Host is up (0.00065s latency).

PORT      STATE SERVICE
21/tcp    open  ftp
MAC Address: 08:00:27:4B:32:42 (Oracle VM VirtualBox)
Nmap done: 1 IP address (1 host up) scanned
```

nmap -p



Source	Destination	Protocol	Length	Info
PcsCompu_19:d5:47	Broadcast	ARP	42	Who has 192.168.0.1? Tell
PcsCompu_4b:32:42	PcsCompu_19:d5:47	ARP	60	192.168.0.1 is at 08:00:27:4B:32:42
192.168.0.111	192.168.0.1	TCP	58	42990 → 21 [SYN] Seq=0 Win=0 Len=0
192.168.0.1	192.168.0.111	TCP	60	21 → 42990 [SYN, ACK] Seq=1000000000 Win=0 Len=0
192.168.0.111	192.168.0.1	TCP	54	42990 → 21 [RST] Seq=1000000000 Win=0 Len=0
PcsCompu_4b:32:42	PcsCompu_19:d5:47	ARP	60	Who has 192.168.0.111? Tell
PcsCompu_19:d5:47	PcsCompu_4b:32:42	ARP	42	192.168.0.111 is at 08:00:27:4B:32:42

wireshark nmap -p 21

Escanear uma lista de portas

Outra opção interessante é especificar uma lista de portas como por exemplo da porta 20

até a 30. Para isso usamos a opção -p seguida da primeira porta e “-” seguido da última porta.

sudo nmap -sS 192.168.0.1 -p 21-23

```
(teste@kali)-[~]  
$ sudo nmap -sS 192.168.0.1 -p 21-23  
Starting Nmap 7.91 ( https://nmap.org )  
Nmap scan report for 192.168.0.1  
Host is up (0.00073s latency).  
  
PORT      STATE SERVICE  
21/tcp    open  ftp  
22/tcp    open  ssh  
23/tcp    open  telnet  
MAC Address: 08:00:27:4B:32:42 (Oracle VM
```

nmap list

Opção do Fast NMAP – F

Para varredura de portas mais rápidas, podemos usar a opção -F. Esta opção -F , verifica as 100 portas comumente usadas ao invés das 1000 portas usadas na varredura padrão do nmap.

sudo nmap -F 192.168.0.1


```
(teste@kali)-[~]
└─$ sudo nmap -sS 192.168.0.1 -F
Starting Nmap 7.91 ( https://nmap.org ) at 2024-08-24 19:21:11
Nmap scan report for 192.168.0.1
Host is up (0.00020s latency).
Not shown: 95 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
80/tcp    open  http
443/tcp   open  https
```

nmap -F

Scan de portas UDP -sU

O nmap também escaneia portas UDP. Para isso, o nmap envia pacotes para as portas UDP do host e aguarda uma mensagem ICMP de destino inalcançado.

sudo nmap -sU 192.168.0.1

Como saber se a porta UDP está aberta?

O nmap vai supor que uma porta está aberta ou filtrada quando não houver resposta icmp de destino inalcançada. O motivo de supor que uma porta está aberta ou filtrada é porque uma porta aberta pode não retornar nenhuma resposta o que

sugere que esteja aberta. No entanto, as respostas icmp podem estar sendo filtradas no host destino ou em um firewall intermediário. Dessa forma, o nmap não tem como ter certeza se a porta está aberta ou filtrada.

```
(teste@kali)-[*]
$ sudo nmap -sU 192.168.0.1 -p 631-635
Starting Nmap 7.91 ( https://nmap.org ) at 2021-09-21 09:24 EDT
Nmap scan report for 192.168.0.1
Host is up (0.00066s latency).

```

PORT	STATE	SERVICE
631/udp	open filtered	ipp
632/udp	closed	bmpp
633/udp	closed	servstat
634/udp	closed	ginad
635/udp	closed	mount

```
MAC Address: 08:00:27:4B:32:42 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 14.50 seconds
```

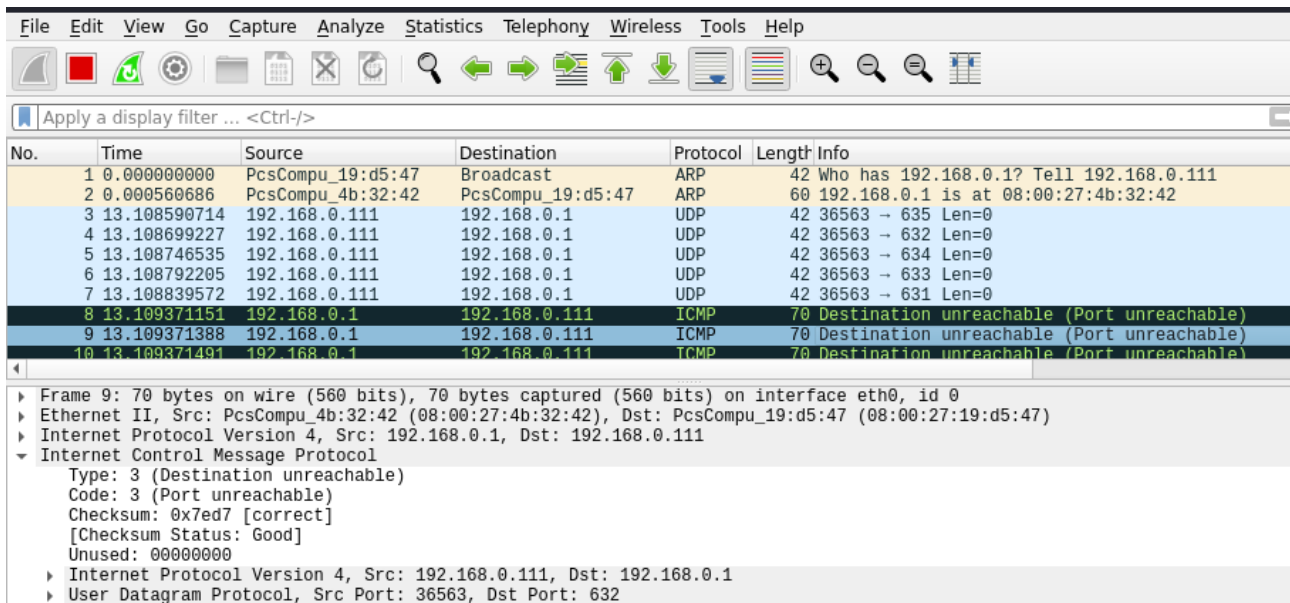
nmap udp scan

Por que escaneamento de portas UDP demora?

O escaneamento de portas UDP pode ficar muito demorado. Isso porque, alguns sistemas operacionais, como é o caso do Linux a partir do kernel 2.4.20, limitam o envio de icmp de destino inalcançável por 1 por segundo. Dessa forma, isso faz com que escanear 65.536 portas leve mais de 18 horas. No entanto existem técnicas para

escanear portas populares, vários hosts em paralelo, dentre outras que podemos falar em outro momento.

Análise de scan UDP no Wireshark



The image shows a Wireshark capture of network traffic. The packet list pane displays several packets. Packets 1-7 are ARP requests and UDP scans. Packets 8-10 are ICMP 'Destination unreachable (Port unreachable)' responses. The packet details pane for packet 9 shows the ICMP response structure.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_19:d5:47	Broadcast	ARP	42	Who has 192.168.0.1? Tell 192.168.0.111
2	0.000560686	PcsCompu_4b:32:42	PcsCompu_19:d5:47	ARP	60	192.168.0.1 is at 08:00:27:4b:32:42
3	13.108590714	192.168.0.111	192.168.0.1	UDP	42	36563 → 635 Len=0
4	13.108699227	192.168.0.111	192.168.0.1	UDP	42	36563 → 632 Len=0
5	13.108746535	192.168.0.111	192.168.0.1	UDP	42	36563 → 634 Len=0
6	13.108792205	192.168.0.111	192.168.0.1	UDP	42	36563 → 633 Len=0
7	13.108839572	192.168.0.111	192.168.0.1	UDP	42	36563 → 631 Len=0
8	13.109371151	192.168.0.1	192.168.0.111	ICMP	70	Destination unreachable (Port unreachable)
9	13.109371388	192.168.0.1	192.168.0.111	ICMP	70	Destination unreachable (Port unreachable)
10	13.109371491	192.168.0.1	192.168.0.111	ICMP	70	Destination unreachable (Port unreachable)

Frame 9: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface eth0, id 0
Ethernet II, Src: PcsCompu_4b:32:42 (08:00:27:4b:32:42), Dst: PcsCompu_19:d5:47 (08:00:27:19:d5:47)
Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.111
Internet Control Message Protocol
Type: 3 (Destination unreachable)
Code: 3 (Port unreachable)
Checksum: 0x7ed7 [correct]
[Checksum Status: Good]
Unused: 00000000
Internet Protocol Version 4, Src: 192.168.0.111, Dst: 192.168.0.1
User Datagram Protocol, Src Port: 36563, Dst Port: 632

wireshark nmap udp scan

Por que o scan UDP repete o envio?

O envio de pacotes UDP podem se perder e a resposta de ICMP mostrando que está inalcançável também pode se perder.

Dessa forma, o nmap repete o envio para garantir que não tenha havido perdas