

SEMANA 06

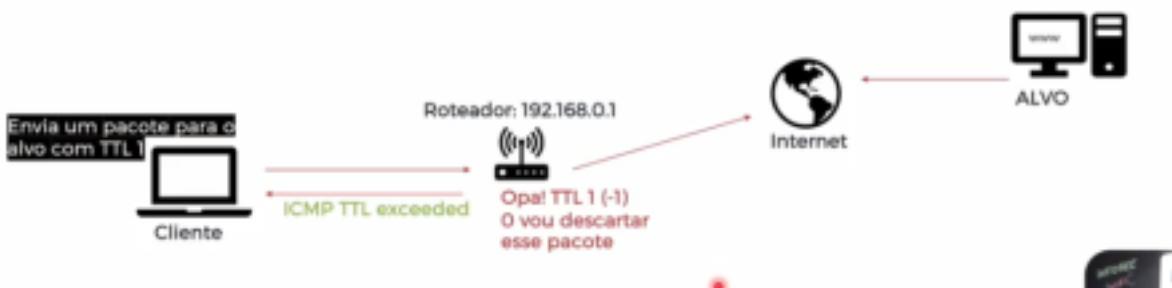
Scanning

Introdução - Scanning

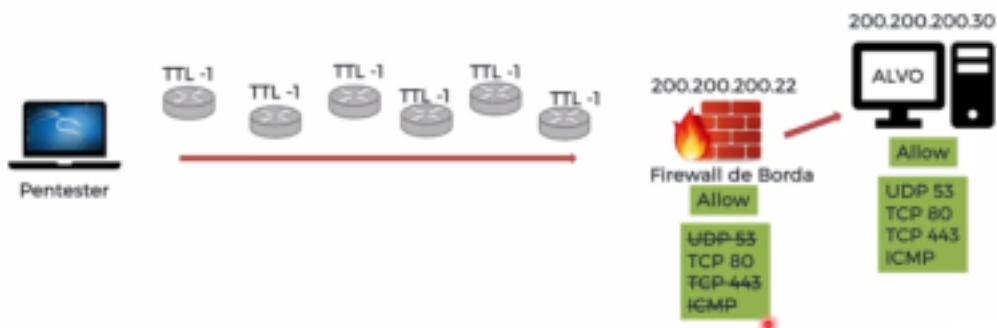
- + Identificar o host/serviço/sistema operacional que está em execução no nosso alvo
- + Com os dados da aula passado encontramos informações como hosts, endereços de IP, mas não sabemos o que esses endereços representam, se é um computador, uma impressora, uma câmera de vídeo, etc.
- + Nessa fase vamos descobrir isso. Técnicas para crackear a rota do pacote, se existe algum filtro. Se tiver portas abertas no host, vamos descobrir, o que passa por elas, qual o sistema operacional.

Tracking the route - Aula Teórica

- + Rastreamento de rota
- + Identificar a rota que os pacotes fazem até chegar no alvo
- + Identificar possíveis filtros e bloqueios
- + Identificar quais pacotes são aceitos
 - o host aceita UDP?
 - o host aceita TCP?
 - o host aceita ICMP?
 - Existe algum filtro por determinadas rotas?
- + Como funciona?
Quando o TTL se torna 0 normalmente o roteador utiliza o protocolo ICMP para avisar, então ele utiliza o ICMP de tipo 11 e código 0.
- Sendo assim, podemos manipular o TTL afim de descobrir o caminho e a rota que o pacote percorre.



Rastreando a rota até o alvo



Tracking the route - Aula Prática

+ Quando fazemos o ping em uma rede, a diferença entre o ttl default e o ttl apresentado representa a quantidade de hosts na rota

+ Veja o seguinte default para os sistemas operacionais:

- Windows - 128
- Linux - 64
- Unix - 255

+ Enviaremos 1 pacote ping para a businesscorp:

```
ping -c 1 businesscorp.com.br
```

```
[root@DESKTOP-NJHHNK6]# ./ping -c 1 businesscorp.com.br
PING businesscorp.com.br (37.59.174.225) 56(84) bytes of data.
64 bytes from ip225.ip-37-59-174.eu (37.59.174.225): icmp_seq=1 ttl=50 time=189 ms
--- businesscorp.com.br ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 188.660/188.660/188.660/0.000 ms
```

Veja que o ttl encontrado foi de 50, o que nos mostra que, como está próximo do 64, possivelmente o sistema alvo é um Linux, e nossa distância à ele é de $64 - 50 = 14$ hosts

+ Para descobrir cada um dos hosts desse caminho, faremos pings consecutivos incrementando uma unidade ao ttl

+ O parâmetro que nos permite controlar o valor do ttl é o **-t**

```

└─(root@DESKTOP-NJHHNK6)-[/home/kali]
└─# ping -c 1 -t 1 businesscorp.com.br
PING businesscorp.com.br (37.59.174.225) 56(84) bytes of data.
From 192.168.1.1 (192.168.1.1) icmp_seq=1 Time to live exceeded

--- businesscorp.com.br ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

└─(root@DESKTOP-NJHHNK6)-[/home/kali]
└─# ping -c 1 -t 2 businesscorp.com.br
PING businesscorp.com.br (37.59.174.225) 56(84) bytes of data.
From 100.90.0.1 (100.90.0.1) icmp_seq=1 Time to live exceeded

--- businesscorp.com.br ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

└─(root@DESKTOP-NJHHNK6)-[/home/kali]
└─# ping -c 1 -t 3 businesscorp.com.br
PING businesscorp.com.br (37.59.174.225) 56(84) bytes of data.
From 172.31.250.6 (172.31.250.6) icmp_seq=1 Time to live exceeded

--- businesscorp.com.br ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

```

~~~~~ Aprendendo a usar o traceroute ~~~~

+ É uma ferramenta que realiza pings e nos devolve a rota e os tempos de resposta de cada roteador

+ Principais parâmetros:

→ traceroute businesscorp.com.br

Modo comum de uso, que por padrão, aguarda resposta de até 3s e utiliza o protocolo UDP

Quando apresenta mais de um endereço de IP em uma linha, significa que houve um balanceamento de carga

Quando aparece um ***, significa que o host local não aceitou o protocolo UDP, e quando vemos apenas 1 *, significa que o tempo de espera foi excedido (>3s)

→ traceroute businesscorp.com.br -w 1

Indica que mudamos o tempo padrão de espera que era 3s, e agora será de apenas 1s

→ traceroute businesscorp.com.br -m 1

Indica que mudamos o ttl, que era 30 e passou a ser apenas 1

→ traceroute businesscorp.com.br -m 20 -f 15

Agora teremos 20 saltos (hops) e só serão exibidos a partir do 15

→ traceroute businesscorp.com.br -A

Com esse parâmetro, serão exibidos os ASN's

→

```
traceroute businesscorp.com.br -n
```

Essa opção não diz o host. Traz uma saída mais limpa mostrando apenas os IP's.

+ Para mudar os protocolos enviados, basta executar os seguintes parâmetros

- I → ICMP

- T → TCP

- U → UDP (vem como padrão, mas quando executamos assim, ele mira na porta 53.)

+ Para mudar a porta de ataque, bas executar o parâmtro -p 43
[nesse caso muda a porta para 43]

Overview sobre Firewall

Firewall - Iptables na Prática

+ iptables é uma ferramenta de Firewall para Linux que, por default, já vem desabilitada (com todas as condições ACCEPT). Ele é como um “guarda de fronteira” enrtre o pc e a internet.

+ `iptables -nL`

→ mostra como estão as regras do firewall na máquina

```
root@firewall:~# iptables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source                               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source                               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source                               destination
root@firewall:~#
```

INPUT → entradas (aceitando qualquer coisa na entrada)

FORWARD → encaminhamento de portas

OUTPUT → saída

todas estão no default que não impede nada.

```
+ iptables -P INPUT DROP
```

- o -P indica a política que vamos implementar.
- nesse caso, mudamos a política de ACCEPT para DROP
- com essa configuração, ele nega tudo
- O host não poderá mais ser pingado, scaneado ou acessado, mesmo com os serviços ativos.

+ Queremos agora liberar algumas entradas (no caso, vamos liberar a porta 80)

```
+ iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

- -p indica o protocolo (que escolhemos o tcp), o 80 é a porta e o -j é a ação que queremos tomar

```
root@firewall:~# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
root@firewall:~# iptables -nL
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:80

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@firewall:~# _
```



- nesse caso, conseguimos acessar o endereço na internet na porta 80, mas ainda n conseguimos fazer um portscanning ou sequer um ping.

+ Agora faremos a liberação do ping, lembrando que o ping se usa do protocolo ICMP

```
+ iptables -A INPUT -p icmp -j ACCEPT
```

```
+ iptables -F
```

- Esse último comando faz que o firewall zere as regras (com exceção das políticas já programadas)

+ Existe uma maneira de filtrar não apenas a porta e o protocolo, mas também o ip de origem.

```
+ iptables -A INPUT -p tcp --dport 80 192.168.0.11 -j ACCEPT
```

- aqui o 192.168.0.11 é o ip de origem

+ Filtraremos agora o tipo do protocolo icmp que poderá passar

```
+ iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
```

+ Se quiséssemos resolver o host businesscorp.com.br com o comando host businesscorp.com.br, não conseguiríamos pois o comando host usa, por default, o protocolo udp.

+ Para habilitar a resolução, devemos habilitar o protocolo udp

```
iptables -A INPUT -p udp -j ACCEPT
```

+ Para filtrar a porta de destino, usamo --dport [n da porta]. Se fosse a de origem, --sport [n da porta]. s de source

Descobrindo Hosts Ativos - Ping Sweep

+ Depois da etapa passada de descobrir os ranges de endereços de IP, vamos agora descobrir quais desses endereços estão ativos e respondendo.

Ping Sweep

+ Basicamente vamos usar o utilitário do ping e ver quais endereços mandam respostas

+ Quando n há resposta do ping, n podemos afirmar diretamente que o host está inativo. Pode ser que haja um Firewall impedindo o ping.

+ Conectamos na VPN da desec e fizemos uma varredura interna na rede com o ping

```
for ip in $(seq 1 254); do ping -c 1 172.16.1.$ip -w 1 ; done | grep "64 bytes"
```

```
[root@DESKTOP-NJHHNK6]# ./ping_sweep.sh
# for ip in $(seq 1 254); do ping -c 1 172.16.1.$ip -w 1 ; done | grep "64 bytes"
64 bytes from 172.16.1.1: icmp_seq=1 ttl=64 time=183 ms
64 bytes from 172.16.1.2: icmp_seq=1 ttl=63 time=183 ms
64 bytes from 172.16.1.3: icmp_seq=1 ttl=63 time=217 ms
64 bytes from 172.16.1.4: icmp_seq=1 ttl=127 time=180 ms
64 bytes from 172.16.1.5: icmp_seq=1 ttl=63 time=185 ms
64 bytes from 172.16.1.7: icmp_seq=1 ttl=63 time=183 ms
64 bytes from 172.16.1.10: icmp_seq=1 ttl=63 time=186 ms
64 bytes from 172.16.1.31: icmp_seq=1 ttl=63 time=193 ms
64 bytes from 172.16.1.33: icmp_seq=1 ttl=63 time=183 ms
^C
```

+ O parâmetro -w serve para indicar o tempo de espera, no caso -w 1 indica espera de apenas 1s pela resposta e o -c 1 indica que enviamos apenas um pacote ping.

+ Para simular o ping sweep numa rede externa, vamos executar o seguinte (no ambiente da businesscorp)

```
for ip in $(seq 224 239); do ping -c 1 37.59.174.$ip -w 1 ; done | grep "64 bytes"
```

```

└# for ip in $(seq 224 239); do ping -c 1 37.59.174.$ip -w 1 ; done | grep "6
4 bytes" | grep "from 172.16.1.2: icmp_seq=1 ttl=63 time=183 ms"
64 bytes from 37.59.174.225: icmp_seq=1 ttl=50 time=188 ms
64 bytes from 37.59.174.226: icmp_seq=1 ttl=50 time=190 ms
64 bytes from 37.59.174.227: icmp_seq=1 ttl=50 time=187 ms
64 bytes from 37.59.174.228: icmp_seq=1 ttl=50 time=192 ms
64 bytes from 37.59.174.229: icmp_seq=1 ttl=50 time=184 ms
64 bytes from 37.59.174.231: icmp_seq=1 ttl=50 time=186 ms
64 bytes from 37.59.174.232: icmp_seq=1 ttl=50 time=190 ms
64 bytes from 37.59.174.239: icmp_seq=1 ttl=50 time=185 ms

```

Ele retorna os hosts que identificou como ativos (no caso, que respondem icmp)

- + Existem ferramentas que fazem isso de maneira automatizada, como o fping

```
fping -a -g 172.16.1.0/24
```

- a para verificar apenas os hosts q respondem
- g para passarmos o range

Estudo Técnico: Ping Sweep

- + Alguns hosts podem estar ativos, mas não responder ao ping por ter algum firewall que bloqueie o protocolo icmp.

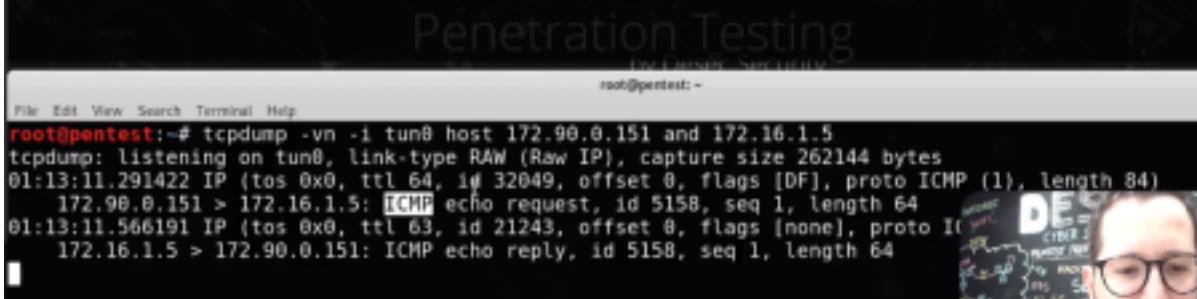
- + O objetivo dessa aula foi fazer a captura [com o tcpdump] e análise do envio de pacotes icmp setados pelo ping numa rede interna.

- + Basicamente, quando o host está com as regras de firewall desligadas, podemos ver a resposta do protocolo icmp

```

root@pentest:~# ping -c1 172.16.1.5
PING 172.16.1.5 (172.16.1.5) 56(84) bytes of data.
64 bytes from 172.16.1.5: icmp_seq=1 ttl=63 time=275 ms
--- 172.16.1.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 274.838/274.838/274.838/0.000 ms
root@pentest:~#

```



- + Quando o host está ativo mas o firewall impede o input de icmp, não conseguiremos ver a ICMP echo reply

```
root@pentest:~# ping -c1 172.16.1.5
PING 172.16.1.5 (172.16.1.5) 56(84) bytes of data.
```

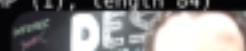


Penetration Testing

by Diogo S. Securidão

File Edit View Search Terminal Help

```
root@pentest:~# tcpdump -vn -i tun0 host 172.90.0.151 and 172.16.1.5
tcpdump: listening on tun0, link-type RAW (Raw IP), capture size 262144 bytes
01:14:44.177151 IP (tos 0x0, ttl 64, id 47700, offset 0, flags [DF], proto ICMP (1), length 84)
    172.90.0.151 > 172.16.1.5: ICMP echo request, id 5165, seq 1, length 64
```



- + Quando o firewall estiver configurado para rejeitar (nem aceitar [accept], nem bloquear [drop]), poderemos ver uma echo reply dizendo que a porta está fechada

```
iptables -A INPUT -p icmp -j REJECT
```

```
root@pentest:~# ping -c1 172.16.1.5
PING 172.16.1.5 (172.16.1.5) 56(84) bytes of data.
From 172.16.1.5 icmp_seq=1 Destination Port Unreachable
...
... 172.16.1.5 ping statistics ...
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
root@pentest:~#
```

I

Penetration Testing

by Diogo S. Securidão

File Edit View Search Terminal Help

```
root@pentest:~# tcpdump -vn -i tun0 host 172.90.0.151 and 172.16.1.5
tcpdump: listening on tun0, link-type RAW (Raw IP), capture size 262144 bytes
01:16:05.053182 IP (tos 0x0, ttl 64, id 61328, offset 0, flags [DF], proto ICMP (1), length 84)
    172.90.0.151 > 172.16.1.5: ICMP echo request, id 5175, seq 1, length 64
01:16:05.297312 IP (tos 0xc0, ttl 63, id 21244, offset 0, flags [none], proto :)
    172.16.1.5 > 172.90.0.151: ICMP 172.16.1.5 protocol 1 port 63344 unreachable
    IP (tos 0x0, ttl 63, id 61328, offset 0, flags [DF], proto ICMP (1), length 84)
    172.90.0.151 > 172.16.1.5: ICMP echo request, id 5175, seq 1, length 64
```



Descobrindo Hosts Ativos - Pentest Interno

- + Encontrando hosts ativos através do

~~~~~  
Protocolo ARP  
~~~~~

- + Esse método só funciona bem quando aplicado presencialmente, foi à distância se usam ferramentas da camada 3 do modelo OSI.

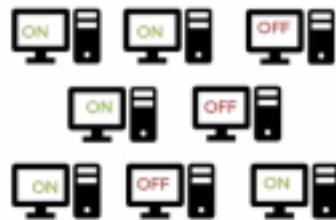
- + De maneira semelhante ao ping, usaremos agr o arping para descobrir quais hosts estão ativos em uma rede

✓ Conectado em um ponto de rede (cabo ou wireless)

Quais hosts estão ativos?



ICMP Bloqueado



+ Veja a captura pelo tcpdump do arping

```
root@pentest:~# arping -c 1 192.168.0.13
ARPING 192.168.0.13
60 bytes from 00:50:56:27:9f:f5 (192.168.0.13): index=0 time=17.904 usec
```

```
... 192.168.0.13 statistics ...
```

```
root@pentest:~#
```

```
root@pentest:~# tcpdump -vn -i eth0 host 192.168.0.11 and 192.168.0.13 -e
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
02:43:57.668187 00:0c:29:13:01:fa > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 64: Ethernet (len 64), IPv4 (len 4), Request who-has 192.168.0.13 tell 192.168.0.11, length 44
02:43:57.669056 00:50:56:27:9f:f5 > 00:0c:29:13:01:fa, ethertype ARP (0x0806), length 60: Ethernet (len 64), IPv4 (len 4), Reply 192.168.0.13 is-at 00:50:56:27:9f:f5, length 46
```

+ Para fazer um script que seja semelhante ao anterior no sentido de fazer a varredura em um range, basta trocar ping por arping e onde tinham 64 bytes que era a resposta padrão do ping, colocaremos agora 60 bytes, como se segue:

```
for ip in $(seq 10 14);do arping -c 1 192.168.0.$ip;done | grep "60 bytes"
```

+ Há uma ferramenta que executa uma varredura automática na rede interna que é o arp-scan. Veja abaixo o seu uso local

```
arp-scan -l
```

```
root@DESKTOP-NJHHNK6:~/kali# arp-scan -l
Interface: wlan0, type: EN10MB, MAC: da:0c:03:87:86:aa, IPv4: 192.168.1.138
WARNING: Cannot open MAC/Vendor file ieeeoui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1      00:6d:b5:d2:20      (Unknown) by kernel
192.168.1.39     24:0a:64:05:63:72      (Unknown) 25 seconds (120.47 hosts/sec), 6 respond
192.168.1.33     28:9c:6e:6a:e2:bc      (Unknown)
192.168.1.33     28:9c:6e:6a:e2:bc      (Unknown) (DUP: 2)

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.868 seconds (137.04 hosts/sec).
```

Descobrindo Hosts Ativos - NMAP

+ Indicado para pentests externos quando o icmp for bloqueado

+ O nmap realiza outros testes que não somente os do protocolo icmp para verificar se um host está ativo ou não. Veja a seguinte captura de tráfego:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.28.1.182	172.16.1.216	ICMP	28	Echo (ping) request id=0x836b, seq=0/0, ttl=52 (reply in 5)
2	0.000026512	172.28.1.182	172.16.1.216	TCP	44	33948 → 443 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3	0.000035232	172.28.1.182	172.16.1.216	TCP	48	33948 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
4	0.000043133	172.28.1.182	172.16.1.216	ICMP	40	Timestamp request id=0xab35, seq=0/0, ttl=55
5	0.170407064	172.16.1.216	172.28.1.182	ICMP	28	Echo (ping) reply id=0x836b, seq=0/0, ttl=63 (request in 1)
6	0.170407068	172.16.1.216	172.28.1.182	TCP	40	443 → 33948 [RST, ACK] Seq=1 ACK=1 Win=0 Len=0
7	0.170514056	172.16.1.216	172.28.1.182	ICMP	40	Timestamp reply id=0xab35, seq=0/0, ttl=63

+ Fizemos um filtro para que o wireshark capturasse apenas pela tun0, pois estamos usando a VPN.

+ Nessa captura, veja que o nmap tentou o ICMP, TCP na porta 443, TCP na porta 80, ICMP novamente e outros. Veja que o protocolo TCP na porta 443 não obteve sucesso, mas o ICMP sim, pois teve uma reply mostrando que o host está ativo.

```
+ nmap -sn 172.16.1.216
```

```
[root@DESKTOP-NJHHNK6 ~]# nmap -sn 172.16.1.216
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-22 18:20 -03
Nmap scan report for 172.16.1.216
Host is up (0.17s latency).
Nmap done: 1 IP address (1 host up) scanned in 0.30 seconds
```

```
+ nmap -sn 172.16.1.0/24 -oG hostsativos
```

-o indica o output

N → normal

X → xml

G → grepable (em que se pode aplicar o grep)

hostsativos é o nome do diretório em que a resposta será guardada

```
cat hostsativos | cut -d " " -f 2 > ips
```

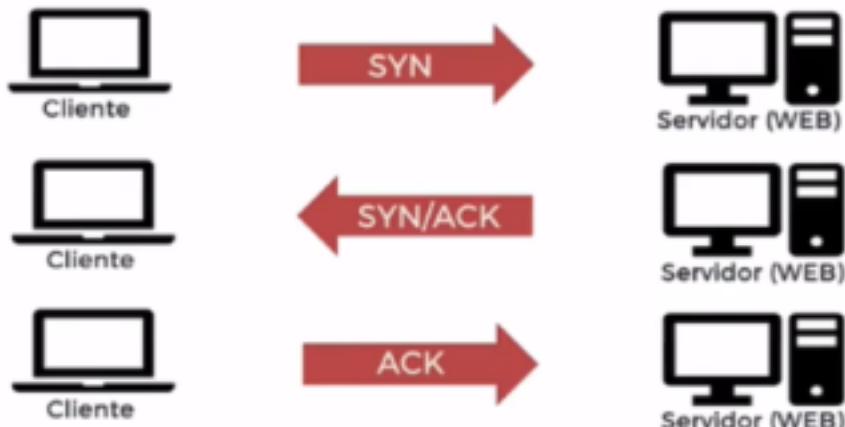
A indicação de que o host estava ativo é a saída “host up”

Introdução ao Port Scanning

+ Depois de identificar quais os hosts ativos, devemos fazer o mapeamento das portas abertas.

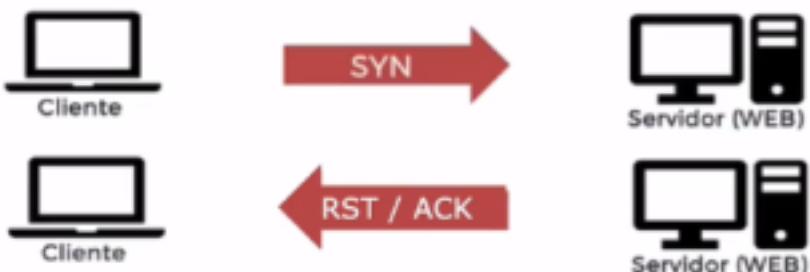
+ Faremos uma breve revisão do 3WHS

3WHS



→ Essa imagem acima representa um serviço ativo pois completou o 3WHS.

Serviço Inativo



PORTE 80 FECHADA

→ Essa próxima imagem já revela um serviço inativo, pois a flag Reset foi enviada como resposta

Estudo Técnico: Port Scanning

+ Vamos usar o **hping3**, que é um utilitário que nos permite enviar pacotes personalizados.

```
hping3 -c 1 --syn -p 80 businesscorp.com.br
```

→ o --syn é pra enviar uma flag SYN

```
[root@DESKTOP-NJHHNK6] [/home/kali]
# hping3 -c 1 --syn -p 80 businesscorp.com.br
HPING businesscorp.com.br (wlan0 37.59.174.225): S set, 40 headers + 0 data bytes
len=44 ip=37.59.174.225 ttl=50 DF id=0 sport=80 flags=SA seq=0 win=14600 rtt=179.9 ms
...
--- businesscorp.com.br hping statistic ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 179.9/179.9/179.9 ms
```

→ A resposta no campo flag=SA significa SYN / ACK, o que indica que completou o 3WHS e, portanto, o serviço está ativo e a porta aberta

→ Caso a resposta fosse RA (de RESET / ACK), a porta estaria fechada

+ Podemos executar isso também usando o **nmap**, que é um scan conhecido de portas

```
nmap -sS -p 80 -Pn businesscorp.com.br
```

→ -s é de scan e -sS é de syn-scan (enviando pacotes SYN)

→ -Pn serve para que ele ignore se o host está ativo ou não

```
[root@DESKTOP-NJHHNK6] [/home/kali]
# nmap -sS -p 80 -Pn businesscorp.com.br
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-22 19:35 -03
Nmap scan report for businesscorp.com.br (37.59.174.225)
Host is up (0.17s latency).
rDNS record for 37.59.174.225: ip225.ip-37-59-174.eu

PORT      STATE SERVICE
80/tcp    open  http
          packet loss
          327.3 ms

Nmap done: 1 IP address (1 host up) scanned in 0.63 seconds
```

→ Ele identificou a porta 80 como aberta

→ Caso executássemos na 81, ela seria dada como fechada

+ Há o caso de que a porta seja identificada como filtrada, conforme mostramos a seguir

```
File Edit View Search Terminal Help
root@pentest:~/Desktop# nmap -sS -p 23,80,8055 -Pn 172.16.1.5
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-19 19:57 -03
Nmap scan report for 172.16.1.5
Host is up (0.24s latency).

PORT      STATE     SERVICE
23/tcp    filtered telnet
80/tcp    open      http
8055/tcp  closed   senomix04
```

+ Geralmente, as portas são dadas como fechadas se a resposta padrão for RST, mas isso pode ocorrer de duas formas:

◇ A porta realmente está fechada

◇ Há um firewall agindo na porta que envia essa resposta

+ A porta é dada como filtrada (existe um firewall protegendo) se não vier resposta da flag SYN

Apply a display filter ... <Ctrl-/>						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	172.90.0.151	172.16.1.5	TCP	44	33545 → 23 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
2	0.000221203	172.90.0.151	172.16.1.5	TCP	44	33545 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3	0.000369512	172.90.0.151	172.16.1.5	TCP	44	33545 → 8055 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
4	0.242477596	172.16.1.5	172.90.0.151	TCP	44	80 → 33545 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1358
5	0.242504779	172.90.0.151	172.16.1.5	TCP	40	33545 → 80 [RST] Seq=1 Win=0 Len=0
6	0.242519079	172.16.1.5	172.90.0.151	TCP	40	8055 → 33545 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
7	1.971891767	172.90.0.151	172.16.1.5	TCP	44	33546 → 23 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

→ Nesse caso acima, veja que um pacote TCP SYN foi enviado duas vezes à porta 23, mas não houve resposta. Isso indica a ação de um firewall

+ Há um parâmetro no nmap que nos permite saber a razão da resposta que ele deu sobre o resultado do escaneamento, é o --reason

```
root@pentest:~/Desktop# nmap -sS -p 80 -Pn 172.16.1.5 --reason
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-19 20:02 -03
Nmap scan report for 172.16.1.5
Host is up, received user-set (0.27s latency).

PORT      STATE      SERVICE REASON
80/tcp    filtered  http      port-unreach ttl 63
```

+ Como resumo:

RESPOSTAS	
PORTE ABERTA	RESponde com SYN / ACK (SA)
PORTE FECHADA	RESponde com RST / ACK (RA)
PORTE COM FILTRO DE FIREWALL EM DROP	SEM RESPOSTA
PORTE COM FILTRO DE FIREWALL EM REJECT	RESponde com um ICMP PORT UNREACHABLE
PORTE COM FILTRO DE FIREWALL EM REJECT COM RST	RESponde como porta fechada (RST)

Diferenças entre os tipos de Scan

~~~~~

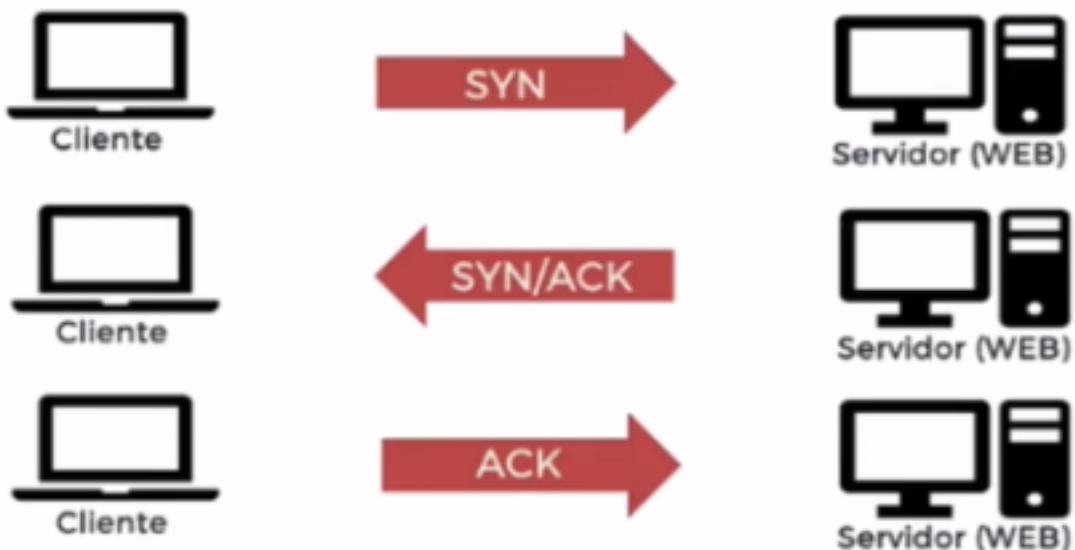
Diferenças entre o TCP Connect e o SYN Scan

~~~~~

+ TCP Connect

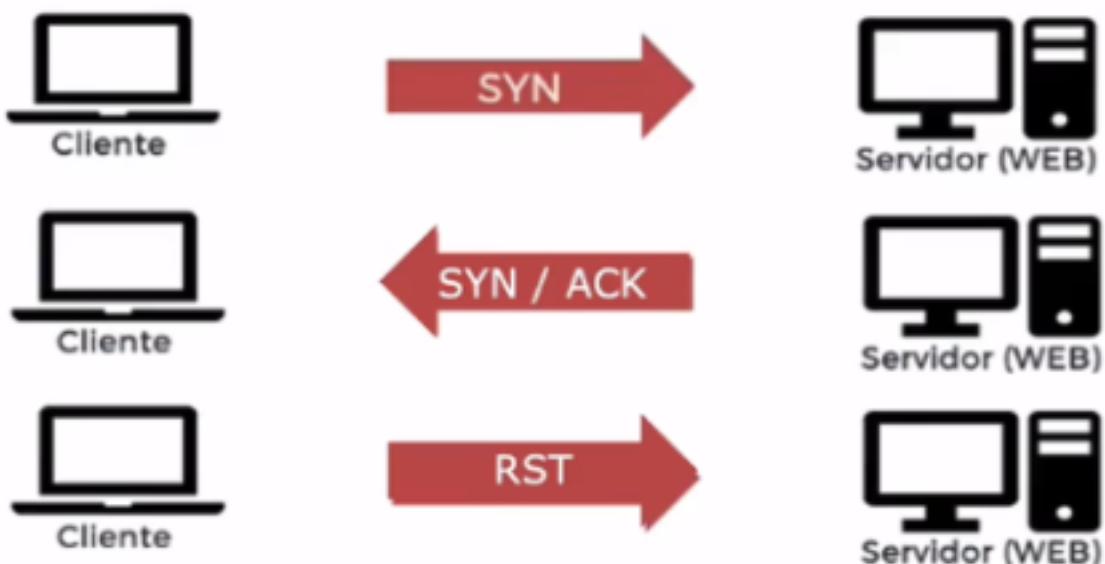
- ◊ Completa o #WHS
- ◊ Facilmente detectável e “barulhento”
- ◊ Consumo maior de tráfego na rede

- ◊ Após estabelecer o 3WHS, envia um RESET para encerrar a conexão



+ Half Open / SYN Scan

- ◊ Não completa o three-way handshake
- ◊ É mais furtivo que o TCP Connect
- ◊ Consome menos tráfego na rede



- + Veja que ambos os testes fornecem o mesmo resultado, que é o de indicar que a porta 80 está aberta

```
root@pentest:~/Desktop# nmap -sT -p 80 -Pn 172.16.1.5
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-19 23:58 -03
Nmap scan report for 172.16.1.5
Host is up (0.33s latency).

PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 0.46 seconds
root@pentest:~/Desktop#
```

Pénétration Testi

root@pentest: ~/Desktop

File Edit View Search Terminal Help

```
root@pentest:~/Desktop# nmap -sS -p 80 -Pn 172.16.1.5
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-19 23:58 -03
Nmap scan report for 172.16.1.5
Host is up (0.29s latency).

PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 0.47 seconds
root@pentest:~/Desktop#
```

→ Mas a diferença repousa sobre o lado do servidor atacado, que não detecta tão facilmente o log via netcat

+ Segue abaixo o resultado do listening no servidor:

```
root@pentest:~/Desktop# nc -vnlp 5555
listening on [any] 5555 ...
connect to [192.168.0.11] from (UNKNOWN) [192.168.0.12] 38892
root@pentest:~/Desktop# nc -vnlp 5555
listening on [any] 5555 ...
```

→ Repare que no primeiro método (TCP Connect), pudemos visualizar quem logou na rede. Já no segundo, não.

+ Existe também uma maneira de fazer o teste com a flag FYN. Pode não ser tão efetivo

```
nmap -sF -p 80 -Pn 172.16.1.5
```

Analisando o Consumo de um Scan

+ Aqui usaremos o `iptables` para fazer a leitura dos tamanhos dos pacotes enviados e recebidos durante a realização de um scan.

+ Faremos testes apenas locais (entre eu e eu) para fazer essa leitura

+ Iniciaremos os serviços do apache2 e do ssh para abrir portas

+ O teste `iptables -nL` mostra quais são as configurações iniciais

```
iptables -nL
```

```
[root@DESKTOP-NJHHNK6 ~]# iptables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

+ Meu IP é 192.168.1.138. Faremos então a abertura de comunicação para esse IP, tanto entrada quanto saída.

```
iptables -A INPUT -s 192.168.1.138 -j ACCEPT
```

-s de source

```
iptables -A OUTPUT -d 192.168.1.138 -j ACCEPT
```

-d de destination

```
iptables -nL
```

```
[root@DESKTOP-NJHHNK6 ~]# iptables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT    0   --  192.168.1.138      0.0.0.0/0
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT    0   --  0.0.0.0/0          192.168.1.138
```

+ Agora faremos o scanneamento com o nmap e analisaremos o tamanho e quantidade dos pacotes por meio do comando iptables -nvL

+ Primeiramente, faremos um scan com TCP Connect

```
nmap -sT -p 80 -Pn 192.168.1.138
```

-Pn para que não seja verificado se o host está ativo ou não (já sabemos que ele está)

```
iptables -nvL
```

```
[root@DESKTOP-NJHHNK6] [/home/kali]
# iptables -nvL
Chain INPUT (policy ACCEPT 60501 packets, 77M bytes)
  pkts bytes target     prot opt in     out     source               destination
  4    224 ACCEPT      0      --  *      *      192.168.1.138      0.0.0.0/0
+ Agora faremos o scaneamento com o nmap e analisaremos o tamanho
  rod que os pacotes por meio do comando
  cu
  Pn para que não seja verificado se o host está ativo ou não (já sabemos
  que ele está)
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
  4    224 ACCEPT      0      --  *      *      0.0.0.0/0
Chain OUTPUT (policy ACCEPT 15946 packets, 2929K bytes)
  pkts bytes target     prot opt in     out     source               destination
  4    224 ACCEPT      0      --  *      *      0.0.0.0/0
  38 can report for businesscorp.com.br (37.59.174.225)
```

→ Veja que temos 4 pacotes entrando e 4 saindo com o TCP Connect

+ Para continuar nossa análise, devemos zerar esses valores, para que
não fiquem sendo somados com os resultados novos. Isso se dá por meio
do comando

```
iptables -Z
```

+ Agora faremos um scaneamento com o SYN Scan

```
nmap -sS -p 80 -Pn 192.168.1.138
```

```
[root@DESKTOP-NJHHNK6] [/home/kali]
# iptables -nvL
Chain INPUT (policy ACCEPT 983 packets, 1355K bytes)
  pkts bytes target     prot opt in     out     source               destination
  3    128 ACCEPT      0      --  *      *      192.168.1.138      0.0.0.0/0
+ Veja que temos 4 pacotes entrando e 4 saindo com o TCP Connect
  rod + Para continuar nossa análise, devemos zerar esses valores, para que
  cu não fiquem sendo somados com os resultados novos. Isso se dá por meio
  Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
  3    128 ACCEPT      0      --  *      *      0.0.0.0/0
Chain OUTPUT (policy ACCEPT 123 packets, 8680 bytes)
  pkts bytes target     prot opt in     out     source               destination
  3    128 ACCEPT      0      --  *      *      0.0.0.0/0
  38 can report for businesscorp.com.br (37.59.174.225)
```

→ Veja que agora foram mandados apenas 3 pacotes, que foram
também recebidos

+ Podemos realizar um scan em todas as portas executando apenas
o comando

```
nmap -sS -p- -Pn 192.168.1.138
```

```
iptables -nvL
```

```

└─[root@DESKTOP-NJHHNK6]─[~/home/kali]
# iptables -nvL
Chain INPUT (policy ACCEPT 4772 packets, 6170K bytes)
  pkts bytes target     prot opt in     out      source          destination
  131K  5505K ACCEPT     0   --  *       *         192.168.1.138    0.0.0.0/0
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out      source          destination
Chain OUTPUT (policy ACCEPT 984 packets, 167K bytes)
  pkts bytes target     prot opt in     out      source          destination
  131K  5505K ACCEPT     0   --  *       *         0.0.0.0/0          192.168.1.1

```

38

+ Fazer a verificação só em portas mais usuais pode ser uma boa ideia pelo motivo de fazer “menos barulho” durante o mapeamento.

Network Mapper - NMAP

NMAP

+ Uma das ferramentas mais conhecidas para realizar scan

+ Já existe a mais de 20 anos

+ Sugestão de livro: Exame de redes com o **nmap**

+ Quando executamos somente nmap no terminal, podemos visualizar as ferramentas dele que podemos usar, como

- Especificação de portas
- Endereços IP
- Redes
- Técnicas de scanneamento
- Detecção de Sistemas Operacionais
- Tempo e performance de execução do mapeamento
- Evasão de Firewalls
- E muitas outras

+ Exemplo:

```
nmap -v -p 21-25,80,8080 -Pn 172.16.1.5
```

-v de verbose (que dá a saída que possamos ler)

-p especifica as portas, no caso da 21 à 25, a 80 e a 8080

-Pn para que não haja verificação de se o host está ativo ou não

- no final, o endereço de ip

```
nmap -v -sS --top-port=5 -Pn 172.16.1.5
```

-sS de SYN Scan

--top-port=5 para escanear as 5 principais portas. Caso n fosse declarado esse parâmetro, ele faria o scan nas 1000 principais (melhor do que fazer nas 65535)
→ Se quisesse varrer toda a rede, poderíamos declarar o ip como 172.16.1.0/24

+ Há também um modo de ter acesso aos scripts de scan do nmap, que é chegar no diretório onde eles estão guardados:

```
cd /usr/share/nmap/scripts
```

Metodologia Scanning

• ~~~~~ ~~~~~ PROCESSO

~~~~~  
→ TCP HOST SCAN - Identificar todas as portas abertas (Protocolo TCP)  
→ UDP HOST SCAN - Identificar todas as portas abertas (Protocolo UDP)  
→ NETWORK SWEEPING - Identificar portas abertas na rede otimizando a busca  
→→→ Esse último é mais indicado quando estamos em uma rede com muitos computadores (tipo uns 5000). A análise de cada porta demoraria demais

### ~~~~~ ~~~~~ DESAFIOS

~~~~~  
→ Tempo de scan e consumo do tráfego na rede
→ Firewall Filtrando/ Rejeitando Pacotes
→ Bloqueio de Portscan
→ Sistemas de Detecção e Prevenção de Intrusos (IDS/IPS)

TCP Host Scan

+ Da aula passada, aprendemos que o nmap realiza a varredura padrão nas 1000 portas mais comuns com o seguinte comando

```
nmap -v -sS -Pn 192.168.0.11
```

+ Porém, podemos realizar a mudança da porta de um serviço
→ Exemplo disso é o ssh, cujas configurações se encontram em /etc/ssh/sshd_config
→ Ao abrirmos elas com o nano, podemos mudar sua porta de funcionamento de 22 para outra qualquer, digamos 22999.

```
GNU nano 7.2                                     /etc/ssh/sshd_config

# This sshd was compiled with PATH=/usr/local/bin:/usr/bin:/bin:/usr/games
# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options override the
# default value.
Include /etc/ssh/sshd_config.d/*.conf
#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

→ Essa porta não seria mais capturada pelo nmap na configuração setada inicialmente

→ Para que a captura dessa porta possa ser feita mesmo depois de modificada, executamos

```
nmap -v -sS -p- -Pn 192.168.0.11
```

UDP Host Scan

+ O UDP Host Scan pode ser mais difícil de executar, dadas as ambiguidades oferecidas pelas respostas de mapeamento, como se segue:

RESPOSTAS	
PORTE ABERTA	SEM RESPOSTA
PORTE FECHADA	PORT UNREACHABLE
PORTE COM FILTRO DE FIREWALL EM DROP	SEM RESPOSTA
PORTE COM FILTRO DE FIREWALL EM REJECT	PORT UNREACHABLE

+ Resposta apresentada pelo escaneamento de uma porta aberta

```
root@pentest:~/Desktop# hping3 --udp -p 69 -c 1 172.16.1.5
HPING 172.16.1.5 (tun0 172.16.1.5): udp mode set, 28 headers + 0 data bytes
... 172.16.1.5 hping statistic ...
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

+ Resposta apresentada pelo escaneamento de uma porta fechada

```
root@pentest:~/Desktop# hping3 --udp -p 161 -c 1 172.16.1.5
HPING 172.16.1.5 (tun0 172.16.1.5): udp mode set, 28 headers + 0 data bytes
ICMP Port Unreachable from ip=172.16.1.5 name=UNKNOWN
status=0 port=2804 seq=0

... 172.16.1.5 hping statistic ...
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 359.6/359.6/359.6 ms
```

→ Port Unreachable

+ Agora, com o iptables, vamos por a configuração de DROP na porta 69:

```
iptables -A INPUT -p udp --dport 69 -j DROP
```

```
root@pentest:~/Desktop# hping3 --udp -p 69 -c 1 172.16.1.5
HPING 172.16.1.5 (tun0 172.16.1.5): udp mode set, 28 headers + 0 data bytes
... 172.16.1.5 hping statistic ...
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

→ Agora não houve resposta ao nosso ping, o que traz a dúvida se ela

→ está aberta ou protegida por um Firewall. Isso é justamente o que conclui

→ o nmap:

```
root@pentest:~/Desktop# nmap -sU -p 69 -Pn 172.16.1.5
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-20 17:40 -03
Nmap scan report for 172.16.1.5
Host is up.

PORT      STATE            SERVICE
69/udp    open|filtered  tftp

Nmap done: 1 IP address (1 host up) scanned in 2.22 seconds
```

→ A resposta dele é que ou está aberta, ou filtrada

+ Colocaremos agora a configuração de REJECT

```
iptables -F
iptables -A INPUT -p udp --dport 69 -j DROP
```

→ a porta não está fechada, apenas com a configuração reject,
→ mas a resposta ao ping é igual à de quando está fechada

```
root@pentest:~/Desktop# hping3 --udp -p 69 -c 1 172.16.1.5
HPING 172.16.1.5 (tun0 172.16.1.5): udp mode set, 28 headers + 0 data bytes
ICMP Port Unreachable from ip=172.16.1.5 name=UNKNOWN
status=0 port=1153 seq=0

... 172.16.1.5 hping statistic ...
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 326.0/326.0/326.0 ms
```

+ Para ajudar nessa situação, vamos executar o

```
nmap -v -sUV -p 161 172.16.1.4
```

→ serão executados testes de conexão com a porta (banner grabbing, por exemplo)
para se verificar o caso de ela estar aberta ou não

```
root@pentest:~/Desktop# nmap -v -sUV -p 161 172.16.1.4
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-20 17:42 -03
NSE: Loaded 45 scripts for scanning.
Initiating Ping Scan at 17:42
Scanning 172.16.1.4 [4 ports]
Completed Ping Scan at 17:42, 0.35s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 17:42
Completed Parallel DNS resolution of 1 host. at 17:42, 0.02s elapsed
Initiating UDP Scan at 17:42
Scanning 172.16.1.4 [1 port]
Completed UDP Scan at 17:42, 3.24s elapsed (1 total ports)
Initiating Service scan at 17:42
Scanning 1 service on 172.16.1.4
Discovered open port 161/udp on 172.16.1.4
Discovered open|filtered port 161/udp on 172.16.1.4 is actually open
Completed Service scan at 17:42, 0.28s elapsed (1 service on 1 host)
NSE: Script scanning 172.16.1.4.
Initiating NSE at 17:42
```

→ Veja que ele executará 45 scripts para fazer esse teste

Network Sweeping

+ O objetivo aqui é fazer um mapeamento mais eficiente de uma rede grande escolhendo bem as portas que serão testadas pelo nmap

+ Primeiro, devemos ter a noção de quais hosts estão ou não ativos
→ Isso pode ser feito pelo seguinte comando

```
nmap -v -sn 172.16.1.0/24 -oG ativos.txt
```

-v de verbose

-sn para que ele verifique se estão ativos

-oG para que a saída (output) seja grepável kkk

ativos.txt será o arquivo em que as respostas serão guardadas

+ Quando analisarmos a varredura do comando passado com o

```
cat ativos.txt
```

veremos que ela exibe o estado de cada host como “up” ou “down”.
Obviamente, faremos um filtro pelos “up”

```
grep "Up" ativos.txt
```

Em seguida, faremos um filtro só pelos endereços de IP

```
grep "Up" ativos.txt | cut -d " " -f 2 > hosts
```

+ Faremos agora uma varredura nas principais portas desses hosts

```
nmap -sSV -p 80 --open -Pn -iL hosts -oG web.txt
```

→ Esse filtro varre apenas a porta 80

-sSV para que seja capturado o Banner Grab e possamos identificar a tecnologia em que o serviço está funcionando

-Pn para não verificar se está ativo ou não (já sabemos que está)

-iL para carregar uma lista, no caso a hosts

+ Em seguida, se queremos, por exemplo, os hosts que rodam Ubuntu, podemos executar

```
grep "Ubuntu" web.txt
```

+ Se quisermos fazer testes em outras portas, podemos ver qual o serviço de nosso interesse e fazer a pesquisa no kali mesmo

→ Exemplo, queremos as portas que rodam serviço ftp para que possamos passar como parâmetro de pesquisa no nmap

Executamos então

```
cat /etc/services | grep "ftp"
```

```

└─[root@DESKTOP-NJHHNK6]─[/home/kali]
# cat /etc/services | grep "ftp"
ftp-data          20/tcp          # TCP port service multiplexer
ftp               21/tcp          # 
tftp              69/udp         sink null
ftps-data         989/tcp        users          # FTP over SSL (data)
ftps              990/tcp        # 
venus-setstat    2431/udp      # udp sftp side effect
codasrv-se        2433/udp      quote          # udp sftp side effect
gsiftp            2811/tcp      ttyst source
zope-ftp          8021/tcp      ttyst source   # zope management by ftp

```

+ Ou, na hora da pesquisa com o nmap, podemos declarar direto o serviço desejado:

```
nmap -sS -p http* --open -Pn -iL hosts
```

→ Ele fará os testes nas portas mais comuns para o http

Identificando Serviços

+ Agora que identificamos as portas abertas, podemos identificar também os serviços que estão ativos

+ Quando fazemos um SYN Scan normal e são identificadas portas abertas, o nmap responde quais eram os serviços baseado na lista de porta-serviço padrão

```
nmap -v -sS -Pn 172.16.1.2
```

PORT	STATE	SERVICE
21/tcp	open	ftp
22/tcp	open	ssh
53/tcp	open	domain
80/tcp	open	http
111/tcp	open	rpcbind

+ Para ter maior certeza dessas informações, é necessário que haja interação com o serviço para que se possa capturar sua resposta.

```
nmap -v -sV -Pn 172.16.1.2
```

```

PORT      STATE SERVICE VERSION
21/tcp    open  ftp    ProFTPD 1.3.4a
22/tcp    open  ssh    -o OpenSSH 6.0p1 Debian 4+deb7u2 (protocol 2.0)
53/tcp    open  domain ISC BIND 9.8.4-rpz2+r105.12-P1
80/tcp    open  http   Apache httpd 2.2.22 ((Debian))
111/tcp   open  rpcbind 2-4 (RPC #100000)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

```

+ Basicamente, ele envia requisição no mesmo padrão da requerida pelo serviço
Podemos fazer também manualmente.

+ Exemplo, pro http, ele envia uma requisição do tipo HEAD:
HEAD / HTTP/1.0
(enviando via nc)

Estudo Técnico: Enganando o Atacante

Serviços Falsos

+ Já vimos em módulos anteriores que se mudarmos a porta padrão de acesso de um serviço qualquer em suas configurações originais e executarmos um scan normal nas portas, o nmap retornará o nome de serviço padrão para aquela porta.

+ Como exemplo, podemos mudar a porta do ssh de 22 (padrão) para 21 (porta padrão do ftp) com o `nano /etc/ssh/sshd_config` e o nmap, caso faça uma varredura comum, vai identificar o serviço ftp como ativo

```
nmap -v -sS -Pn 192.168.1.138
```

```

PORT      STATE SERVICE
21/tcp    open  ftp
80/tcp    open  http

```

+ Mas caso executemos uma condição de interação com o serviço, poderemos capturar o banner e descobrir que serviço está rodando ali.

```
nmap -v -sV -Pn 192.168.1.138
```

```

PORT      STATE SERVICE VERSION
21/tcp    open  ssh    OpenSSH 9.4p1 Debian 1 (protocol 2.0)
80/tcp    open  http   Apache httpd 2.4.58 ((Debian))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

```

+ O objetivo dessa aula é modificar esse banner nos bytes do protocolo de execução desse serviço

+ Faremos isso por intermédio de uma ferramenta chamada `bless`

File Edit View Search Tools Help

ssh x

This file has been changed on disk. You may choose to ignore the changes but reloading is the only safe option.

00080f49	72 73 69 6F 6E 20 6D 69 73 6D 61 74 63 68 00 4B 69 6C 6C 69 6E 67 20 70 72 69 76	rsion mismatch.Killing priv
00080f64	73 65 70 20 63 68 69 6C 64 20 25 64 00 25 73 3A 20 6B 69 6C 6C 28 25 64 29 3A 20	sep child %d.%s: kill(%d):
00080f7f	25 73 00 72 65 78 65 63 00 70 72 69 76 61 74 65 00 20 72 64 6F 6D 61 69 6E 20 00	%s.rexec.private. rdomain .
00080f9a	20 72 64 6F 6D 61 69 6E 20 22 00 73 65 74 67 72 6F 75 70 73 28 29 3A 20 25 2E 32	rdomain ".setgroups(): %2
00080fb5	30 30 73 00 5B 63 6F 6D 6D 61 6E 64 2D 6C 69 6E 65 5D 00 74 6F 6F 20 6D 61 6E 79	00s.[command-line].too many
00080fd0	20 70 6F 72 74 73 2E 0A 00 42 61 64 20 70 6F 72 74 20 6E 75 6D 62 65 72 2E 0A 00	ports...Bad port number...
00080feb	49 6E 76 61 6C 69 64 20 6C 6F 67 69 6E 20 67 72 61 63 65 20 74 69 6D 65 2E 0A 00	Invalid login grace time...
00081006	49 6E 76 61 6C 69 64 20 75 74 6D 70 20 6C 65 6E 67 74 68 2E 0A 00 63 6F 6D 6D 61	Invalid utmp length...comma
00081021	6E 64 2D 6C 69 6E 65 00 4F 70 65 6E 53 53 48 5F 38 2E 30 70 31 20 44 65 62 69 61	nd-line.OpenSSH_8.0p1 Debian
0008103c	6E 2D 34 00 25 73 2C 20 25 73 0A 00 4B 52 42 35 43 43 4E 41 4D 45 00 6E 6F 6E 65	n-4.%s, %s..KRB5CCNAME.none
00081057	00 45 78 74 72 61 20 61 72 67 75 6D 65 6E 74 20 25 73 2E 0A 00 4F 70 65 6E 53 53	.Extra argument %s...OpenSS

Search for: OpenSSH

Signed 8 bit: 95 Unsigned 8 bit: 95 Signed 16 bit: 24376 Unsigned 16 bit: 24376

Signed 32 bit: 1597517360 Unsigned 32 bit: 1597517360

Float 32 bit: 1.32716E+19 Float 64 bit: 4.94699747220254E+150

Hexadecimal: 5F 38 2E 30 Decimal: 095 056 046 048 Octal: 137 070 056 060 Binary: 0101111 00111000 00101110 00110000

Show little endian decoding Show unsigned as hexadecimal

ASCII Text: _8.0 Offset: 0x81073 / 0xc9d87 Selection: 0x810

+ Aparentemente ela n existe mais. Mas basicamente abrimos o arquivo sshd que estava localizado na /usr/sbin/sshd e modificávamos as linhas de texto correspondentes ao banner do protocolo. A ferramenta fazia a modificação dos bytes (funciona desde que tenhamos o cuidado de fazer arquivos de mesmo tamanho)

bless sshd

OS Fingerprinting

Identificando o Sistema Operacional

+ Existem várias técnicas para se identificar o sistema operacional, como:

→ TTL: Cada sistema operacional utiliza um padrão de TTL

- Windows = 128
- Linux = 64

- FreeBSD = 64
- Solaris = 255
- Cisco = 254

→ Análise de Serviços (Remote Desktop [3389] / SSH [22] / Webserver)
→ Implementação da Pilha TCP/IP
→ Enumeração de Serviços Identificados
→ NMAP -O / -A

~~~~~

+ Os testes de TTL podem ser feitos por meio do seguinte comando

```
for i in $(seq 1 254); do ping -c1 -w1 172.16.1.$i; done  
| grep "64 bytes"
```

→ Ao visualizarmos a resposta, o normal é que os TTLs estejam próximos dos esperados para os sistemas operacionais padrão

+ A análise de serviços pode se dar pelos filtros dos arquivos que aprendemos a montar nas aulas passadas

Ex: IIS indica serviço Windows

→ Um grep “Terminal” pode indicar em quais portas está rodando um Microsoft Terminal

+ Para tentar fazer um acesso remoto ao sistema, podemos utilizar a ferramenta `rdesktop`:

```
rdesktop 172.16.1.140
```

+ Uma outra maneira de procurar identificar o OS (Operational System) é por meio do nmap:

```
nmap -v -O 172.16.1.140 -Pn
```

+ Vale lembrar que todas essas saídas podem ser manipuladas para enganar um atacante.

→ Para editar o ttl, podemos executar:

```
echo "128" > /proc/sys/net/ipv4/ip_default_ttl
```

## LAB - SEM 06 - OSINT

LAB01: 21,80,110,143

executamos a varredura do nmap q passa por todas as portas

```
nmap -v -sT -p- mail.businesscorp.com.br
```

```
PORT      STATE SERVICE
19/tcp    filtered chargen
21/tcp    open   ftp
25/tcp    filtered smtp
80/tcp    open   http
110/tcp   open   pop3
137/tcp   filtered netbios-ns
143/tcp   open   imap
1900/tcp  filtered upnp
2378/tcp  filtered dali
10571/tcp filtered unknown
10601/tcp filtered unknown
11211/tcp filtered memcache
14324/tcp filtered unknown
14523/tcp filtered unknown
42726/tcp filtered unknown
47541/tcp filtered unknown
51819/tcp filtered unknown
63859/tcp filtered unknown
```

## LAB02: 53

Executamos uma busca mais rápida do nmap

```
nmap -sU -Pn mail.businesscorp.com.br --open
```

```
└# nmap -sU -Pn mail.businesscorp.com.br --open
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-23 16:09 -03
Nmap scan report for mail.businesscorp.com.br (37.59.174.227)
Host is up (0.17s latency).
rDNS record for 37.59.174.227: ip227.ip-37-59-174.eu
Not shown: 954 closed udp ports (port-unreach), 45 open|filtered udp ports (no
-response) | Edit View Help
PORT      STATE SERVICE
53/udp    open   domain
Nmap done: 1 IP address (1 host up) scanned in 1307.69 seconds
```

Que só retornou a porta 53 como udp aberta

## LAB03: 901238127-1281321-d192919

Uma vez que a porta estava aberta, fizemos a conexão via nc

```
[root@DESKTOP-NJHHNK6]~[/home/kali]
# nc -vu mail.businesscorp.com.br 53
DNS fwd/rev mismatch: mail.businesscorp.com.br ≠ ip227.ip-37-59-174.eu
mail.businesscorp.com.br [37.59.174.227] 53 (domain) open
ola
DESEC DNS SERVER 901238127-1281321-d192919
key
DESEC DNS SERVER 901238127-1281321-d192919
pwd
DESEC DNS SERVER 901238127-1281321-d192919
cd
DESEC DNS SERVER 901238127-1281321-d192919
^C
```

#### LAB04: camila,ca123456

Fazendo o filtro pelo pastebin no google:  
businesscorp “pastebin”

#### LAB05: postfix

Primeiro fizemos um brute force de diretórios com o dirb, que usou sua common word list:

```
dirb mail.businesscorp.com.br
```

Com isso, obtivemos um diretório chamado squirrelmail  
Ao logar com o usuário e senha encontrados, pudemos analisar o header da mensagem recebida pelo dev  
[vale lembrar que antes de resetar a máquina, a pesquisa não tinha dado certo pois a caixa de entrada estava vazia]

```
Return-Path: <dev@businesscorp.com.br>
X-Original-To: camila@businesscorp.com.br
Delivered-To: camila@businesscorp.com.br
Received: by businesscorp.com.br (Postfix, from userid 33)
          id 978964300F; Tue, 9 Oct 2018 21:32:16 -0400 (EDT)
Received: from 189.29.146.62
          (SquirrelMail authenticated user dev)
          by 37.59.174.227 with HTTP;
          Tue, 9 Oct 2018 21:32:16 -0400
Message-ID: <47c63b7b2c06622bec4d8706a2a9b4f4.squirrel@37.59.174.227>
Date: Tue, 9 Oct 2018 21:32:16 -0400
Subject: Atualizacao
From: dev@businesscorp.com.br
To: jotafsantos@businesscorp.com.br
Cc: camila@businesscorp.com.br
User-Agent: SquirrelMail/1.4.23 [SVN]
MIME-Version: 1.0
Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: 8bit
X-Priority: 3 (Normal)
Importance: Normal
```

#### LAB06: squirrelmail

Segue do lab passado

**LAB07:** squirrelmail1.4.23

Basta ler o campo do User-Agent

**LAB08:** [jotafsantos@businesscorp.com.br](mailto:jotafsantos@businesscorp.com.br)

Basta olhar a caixa de entrada

**LAB09:** 09/10/2018

Segue do lab passado

**LAB10:** 189.29.146.62

Veja a imagem printada

```
Return-Path: <jotafsantos@businesscorp.com.br>
X-Original-To: camila@businesscorp.com.br
Delivered-To: camila@businesscorp.com.br
Received: by businesscorp.com.br (Postfix, from userid 33)
          id 6AB7D4300F; Tue, 9 Oct 2018 21:28:08 -0400 (EDT)
Received: from 189.29.146.62
          (SquirrelMail authenticated user jotafsantos)
          by 37.59.174.227 with HTTP;
          Tue, 9 Oct 2018 21:28:08 -0400
Message-ID: <f46df0fea060ca62f4b7eb4832d99c76.squirrel@37.59.174.227>
Date: Tue, 9 Oct 2018 21:28:08 -0400
Subject: Dados
From: jotafsantos@businesscorp.com.br
To: camila@businesscorp.com.br
Cc: camila@businesscorp.com.br
User-Agent: SquirrelMail/1.4.23 [SVN]
MIME-Version: 1.0
Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: 8bit
X-Priority: 3 (Normal)
Importance: Normal
```

**LAB11:** bj9384221-

Basta ler o campo de mensagens enviadas: sents

**LAB12:** [dev@businesscorp.com.br](mailto:dev@businesscorp.com.br)

Veja na caixa dos emails recebidos, só tem 2 emails

**LAB13:** camila,bj9384221-

O conteúdo da mensagem trocada mostra que a senha enviada pela camila é dela msm

**Subject:** Dados**From:** [jotafsantos@businesscorp.com.br](mailto:jotafsantos@businesscorp.com.br)**Date:** Tue, October 9, 2018 9:28 pm**To:** [camila@businesscorp.com.br](mailto:camila@businesscorp.com.br)**Cc:** [camila@businesscorp.com.br](mailto:camila@businesscorp.com.br)**Priority:** Normal**Options:** [View Full Header](#) | [View Printable Version](#) | [Download this as a file](#)

Ca,

Estou precisando da sua senha para cadastro, a minha esta bloqueada.

Pode me passar?

Bjs

**LAB14:** [327-9931231-48848d3](#)

Basta entrar no site da businesscorp e ir para o servidor de email que será apresentada uma página para logar na intranet. Fazemos o login com o usuário e senha do lab passado

**Bem vindo a INTRANET CAMILA!**

Sua key: [327-9931231-48848d3](#)

## **LAB - SEM 06 - SCANNING**

**LAB01:** [21,22,53,80,111](#)

Executamos o nmap varrendo em todas as portas com pacotes TCP

```
nmap -v -sT -p- businesscorp.com.br --open
```

| PORT    | STATE | SERVICE |
|---------|-------|---------|
| 21/tcp  | open  | ftp     |
| 22/tcp  | open  | ssh     |
| 53/tcp  | open  | domain  |
| 80/tcp  | open  | http    |
| 111/tcp | open  | rpcbind |

**LAB02:** [bind9.8.4](#)

Agora pedimos para o nmap interagir com os serviços

```
nmap -v -sTV -p 21,22,53,80,111 businesscorp.com.br
```

```
PORT      STATE SERVICE VERSION
21/tcp    open  ftp     ProFTPD 1.3.4a
22/tcp    open  ssh     OpenSSH 6.0p1 Debian 4+deb7u2 (protocol 2.0)
53/tcp    open  domain ISC BIND 9.8.4-rpz2+rl005.12-P1
80/tcp    open  http   Apache httpd 2.2.22 ((Debian))
111/tcp   open  rpcbind 2-4 (RPC #100000)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

### LAB03: proftpd1.3.4a

Segue do lab passado

### LAB04: vinte e dois

Primeiramente usamos a ferramenta fping, que já automatiza o exato processo solicitado: verificar quais host respondem ping

```
fping -a -g 172.30.0/24 > ativos_na172_30_0.txt
```

-a para verificar apenas os hosts que respondem

-g para designar o range

Em seguida, contamos as linhas do arquivo com o comando wc

```
wc ativos_na172_30_0.txt
```

### LAB05: vinte e tres

Usamos o nmap para fazer essa investigação com os seguintes parâmetros

```
nmap -sn 172.30.0.0/24 -oG hostativos_172_30
```

-o de output

-G de grepable

o último argumento é o nome do diretório que queremos que ele salve

Em seguida, fizemos um cut para limpar a resposta e encaminhamos para o arquivo hostativos\_172\_30.txt

```
cat hostativos_172_30 | cut -d " " -f 2 > hostativos_172_30.txt
```

Por default, o arquivo vinha com os nomes “nmap” no topo e no fim

Retiramos com o editor nano e por fim contamos as linhas como feito anteriormente com o comando wc

### LAB06: 172.30.0.103

Tentamos o filtro do ttl, mas todos tinham ttl de 63 ou 64

Tentamos o filtro do nmap mas tb n tivemos sucesso

Por fim, com o rdesktop obtivemos sucesso

```
for ip in $(cat hostativos_172_30.txt); do echo "tentando no $ip"; timeout 2s rdesktop $ip ; done
```

Verificamos que apenas o ip 130 ofereceu uma conexão disponível, o que indica a presença do windows

O teste do nmap, embora demorado, também acusou o OS

```
for ip in $(cat hostativos_172_30.txt); do nmap -v -O $ip -Pn; done
```

```
Nmap scan report for 172.30.0.103 (timeout 2s) | rdesktop $ip | done  
Host is up (0.17s latency).  
Not shown: 996 filtered tcp ports (no-response)  
PORT      STATE SERVICE  
135/tcp    open  msrpc  
139/tcp    open  netbios-ssn  
445/tcp    open  microsoft-ds [SN] | timeout 2s rdesktop $ip | done  
8080/tcp   open  http-proxy  
Warning: OSScan results may be unreliable because we could not find at least 1  
open and 1 closed port  
Device type: general purpose  
Running (JUST GUESSING): Microsoft Windows 2012 (86%)
```

### LAB07: 135,139,445,2810,5985,8080

Executamos o teste mais demorado possível (pq eu ia lavar a louça)

```
nmap -v -sV 172.30.0.103 --open
```

```
PORT      STATE SERVICE      VERSION  
135/tcp    open  msrpc      Microsoft Windows RPC  
139/tcp    open  netbios-ssn Microsoft Windows netbios-ssn  
445/tcp    open  microsoft-ds Microsoft Windows Server 2008 R2  
- 2012 microsoft-ds (workgroup: DHC17)  
2810/tcp   open  netsteward?  
5985/tcp   open  http       Microsoft HTTPAPI httpd 2.0 (SSD  
P/UPnP)  
8080/tcp   open  http       Apache Tomcat/Coyote JSP engine  
1.1  
Service Info: Host: SRV01; OS: Windows; CPE: cpe:/o:microsoft:windows
```

### LAB08: 172.30.0.128

Pesquisamos no chatgpt quais as portas mais comuns para o serviço de email. A partir disso, fizemos a varredura usando o nmap:

```
nmap -p 25,587,110,143,465,993,995 172.30.0.0/24 --open
```

```
Nmap scan report for 172.30.0.128  
Host is up (0.17s latency). [correct result]  
Not shown: 6 closed tcp ports (reset)  
PORT      STATE SERVICE  
25/tcp    open  smtp     26 smtp 0.346KB | Rcvd: 1140
```

Apenas esse host foi setado.

### LAB09: postfix

Identificamos o serviço utilizando o nmap na porta especificada no host achado no lab passado

```
nmap -v -sV -p 25 172.30.0.128
```

-v de verbose

-s de scan

-V de para saber o serviço/versão

```
PORT      STATE SERVICE VERSION
25/tcp    open  smtp      Postfix smtpd
Service Info: Host: anubuntu.blois.com.br
```

### LAB10: ubuntu

Segue do lab passado

### LAB11: 172.30.0.20,172.30.0.104

Pesquisamos no chatgpt as portas mais comuns para execução do MySql e descobrimos que é a 3306 e a 1186. A partir disso, fizemos a varredura com o nmap:

```
nmap -v -p 3306,1186 172.30.0.0/24 --open
```

```
Nmap scan report for 172.30.0.20
Host is up (0.17s latency).
Not shown: 1 closed tcp port (reset)
PORT      STATE SERVICE
3306/tcp  open  mysql
Nmap scan report for 172.30.0.104
Host is up (0.17s latency). Rcvd: 1140 bytes
Not shown: 1 closed tcp port (reset)
PORT      STATE SERVICE
3306/tcp  open  mysql
```

### LAB12: 172.30.0.15

A porta padrão para execução do webmin é a 10000

Proseguimos da mesma maneira do lab passado e apenas um host foi setado

```
nmap -v -p 10000 172.30.0.0/24
```

# Burlando Mecanismos de Defesa

## Bypass de Firewall

- + As vezes, quando vamos fazer um mapeamento em um host, o firewall existente pode estar mal configurado ao ponto de permitir o acesso a partir de algumas portas de origem
- + Para verificar isso na prática, acessamos o 172.16.1.5 da rede interna por meio do comando

```
ssh root@172.16.1.5 -o HostKeyAlgorithms=+ssh-dss -o PubkeyAcceptedAlgorithms=+ssh-rsa
```

usando **root** como USER e PASSWORD.

- + Antes de ativar a ./rules4.sh em /home/msfadmin/firewall fizemos a varredura de portas com o kali

```
nmap -v -sS -Pn 53 172.16.1.5
```

```
PORT      STATE SERVICE  
21/tcp    open  ftp  
22/tcp    open  ssh  
23/tcp    open  telnet  
25/tcp    open  smtp  
53/tcp    open  domain  
80/tcp    open  http  
111/tcp   open  rpcbind  
139/tcp   open  netbios-ssn  
445/tcp   open  microsoft-ds  
512/tcp   open  exec  
513/tcp   open  login  
514/tcp   open  shell  
1099/tcp  open  rmiregistry  
1524/tcp  open  ingreslock  
2049/tcp  open  nfs  
3306/tcp  open  mysql  
5432/tcp  open  postgresql  
5900/tcp  open  vnc  
6000/tcp  open  X11  
6667/tcp  open  irc  
8009/tcp  open  ajp13  
8180/tcp  open  unknown
```

- + Em seguida, ativamos a ./rules4.sh e a mesma varredura mostrou-se ineficiente

```
nmap -v -sS -Pn 172.16.1.5
```

```
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
```

+ Para dar o bypass nesse firewall, usamos como porta de origem a 53, que ele deixava passar

```
nmap -v -sS -Pn -g 53 172.16.1.5
```

-g indica a porta de origem

```
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown
```

+ Para conseguir conectar ao serviço, fizemos uma conexão via netcat especificando a porta de origem e mandando o resultado da interação para a mesma pasta do apache em um arquivo chamado recon.html

```
nc -vn -p 53 172.16.1.5 > var/www/html/recon.html
```

o qual pudemos acessar depois de fazer uma requisição GET (GET / HTTP/1.0) com nosso endereço de ip no navegador: 192.168.1.138/recon.html

**OBS:** Acesso ao ssh

Comando SSH:

```
ssh root@172.16.1.5 -o HostKeyAlgorithms=+ssh-dss -o
PubkeyAcceptedAlgorithms=+ssh-rsa
```

Credenciais:

- user: root
- pass: root

## ***IDS - Sistema de Detecção de Intrusos***

+ O conceito de IDS, que pode estar funcionando em um host ou até mesmo em uma rede, é que ele acusa estar acontecendo um scanneamento de portas,

exploração ou sequer interação com um determinado serviço.  
Ele é basicamente um dedo-duro, pois alerta o admin.

+ Mas ele realiza esse aviso baseado nas regras que estão pré configuradas

+ Existe também o IPS, que é um sistema de prevenção

A diferença é que o IPS realiza alguma ação preventiva, ele não apenas avisa.

+ O sistema usado para o estudo no caso foi o **snort**

+ O diretório fica armazenado na máquina em /etc/snort

+ Suas regras ficam guardadas em /snort/rules

+ As configurações estão armazenadas no arquivo snort.conf

+ A maneira de iniciar as regras no terminal é:

```
snort -A fast -q -h 192.168.0.0/24 -c snort.conf
```

-A pelo tipo de arquivo

-h para setar a rede

-q de quiet para não mostrar tanta informação na nossa tela

+ Para verificar o que está ocorrendo, podemos ver no arquivo alert que fica em /var/log/snort

+ Ao executarmos o tail -f alert, poderemos ver em tempo real qualquer ação do atacante

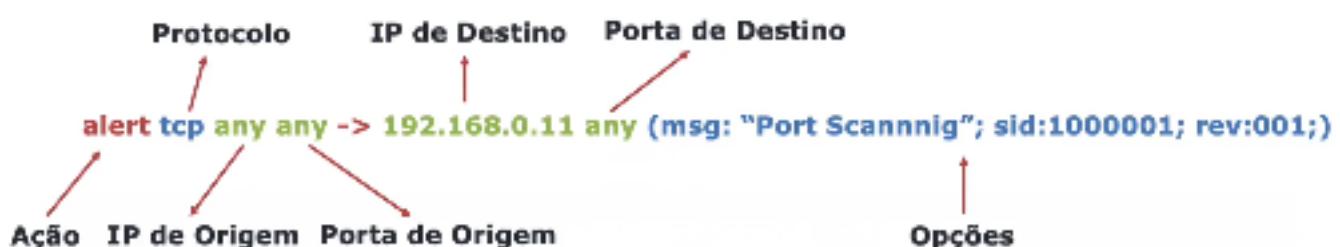
## **Entendendo e Criando Regras de IDS**

+ Mostraremos o exemplo de montagem de regras do snort

+ O comando para verificar os alertas do snort é o seguinte:

```
snort -A console -q -n 192.168.0.0/24 -c snort.conf
```

+ Segue abaixo o modelo padrão de uma regra simples



→ sendo a mensagem uma declaração qualquer, que podemos escolher à vontade

→ sid é um identificador (é necessário que sempre haja pelo menos 5 zeros no meio)

→ rev é o marcador da revisão (quantas vezes modificamos a regra)

+ Na prática, fez-se com o nano o arquivo desec.rules

+ Abriu-se o arquivo padrão de configurações do snort com o nano no diretório /etc/snort e habilitou-se o desec.rules da seguinte maneira:

```

include $RULE_PATH/community-web-client.rules
include $RULE_PATH/community-web-dos.rules
include $RULE_PATH/community-web-iis.rules
include $RULE_PATH/community-web-misc.rules
include $RULE_PATH/desec.rules
#####
# Step #8: Customize your preprocessor and decoder alerts
# For more information, see README.decoder_preproc.rules
#####
# decoder and preprocessor event rules
# include SPREPROC_RULE_PATH/preprocessor.rules

```

+ Exemplo de regras incluídas

```

root@penbook:/etc/snort/rules
GNU nano 4.3 desec.rules
alert tcp any any -> 192.168.0.11 any (msg: "Possível Port Scanning";sid:1000001;rev:001;)
alert tcp any any -> 192.168.0.11 22 (msg: "Pacote SYN enviado ao SSH";flags:S;sid:1000002;rev:001;)
alert tcp any any -> 192.168.0.11 80 (msg: "Acesso ao arquivo robots.txt";content:"robots.txt";sid:1000003;rev:001;)

```

→ Com essas regras, o IDS identificará

- qualquer pacote TCP que saia de qualquer endereço e qualquer porta e chegue no endereço 192.168.0.11 em qualquer porta como um possível port scanning.
- Qualquer pacote TCP que saia de qualquer endereço chegue ao endereço 192.168.0.11 na porta 22 como pacote SYN enviado ao ssh.
- Qualquer pacote TCP que saia de qualquer endereço e de qualquer porta e chegue ao 192.168.0.11 na porta 80 e acesse o robots.txt

## ***Estudo Técnico - Bypass de Regras de IDS***

+ O objetivo é que eu aprenda a fazer uma pesquisa para aprender a burlar as regras existentes de qualquer bypass que me apareça

+ Pelo menos nesse módulo eu entendi a importância de se saber a versão do Firewall usado pela empresa. Para que eu possa estudá-lo e realizar as adaptações adequadas aos meus teste afim de que se chegue ao bypass das regras.

+ O princípio é que se entenda o que os filtros buscam na interação para fazer a declaração ao administrador, para então modificar as interações que fazemos

+ É necessário que eu use o firewall em um ambiente controlado para verificar a ação dessas regras e os requisitos de cada uma

+ Vamos usar o exemplo dado em aula, do snort

[SITUAÇÃO]

+ Enviamos um simples pacote ping para o alvo

**ping -c1 192.168.0.11**

+ 3 Regras nos acusaram:

```
root@pentest:/etc/snort# snort -A console -q -h 192.168.0.0/24 -c snort.conf
12/22-23:11:48.483067 [**] [1:366:7] ICMP PING *NIX [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.12 -> 192.168.0.11
12/22-23:11:48.483067 [**] [1:384:5] ICMP PING [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.12 -> 192.168.0.11
12/22-23:11:48.483124 [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc activity] [Priority: 3] {ICMP} 192.168.0.11 -> 192.168.0.12
^C*** Caught Int-Signal
```

+ Fizemos um grep no diretório das regras para encontrar a primeira regra:

```
root@pentest:/etc/snort/rules# grep -r "ICMP PING *NIX"
root@pentest:/etc/snort/rules# grep -r "ICMP PING \*NIX"
icmp-info.rules:alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING *NIX"; itype:8; content:"|10 11 12 13 14 15 16
17 18 19 1A 1B 1C 1D 1E 1F|"; depth:32; classtype:misc-activity; sid:366; rev:7;)
```

→ Veja que a regra tem 3 requisições:

1 - itype: 8 (essa não dá pra mudar pois indica que é uma request) e

2 - content [conteúdo]: 11 12 13 ... 1F e

3 - depth: 32

+ O objetivo então é mudar o formato do ping para que ele não seja capturado por essa regra

→ Basta mudar uma das características mencionadas acima para que obtenhamos êxito

+ No caso, mudamos apenas o conteúdo do ping com o seguinte comando

```
ping -c1 -p "53461643461" 192.168.0.11
```

→ mudando apenas o corpo da mensagem, foi-se possível evadir a primeira regra

+ A partir disso, devemos sair olhando as demais regras para que se entenda o que elas requerem para que possamos passar por todas sem sermos detectados.

# **IPS / Bloqueio de Port Scanning**

+ O objetivo é mostrar princípios de funcionamento de um IPS que bloqueia scanner de portas

+ O IPS usado foi o [portsentry](#)

+ Basicamente ele fica de olho em portas que não são muito usuais e quando o endereço de IP acessa ela, o IPS registra o endereço e adota uma configuração de drop para aquele domínio

+ Para ajustar essa configuração basta comentar a seguinte sentença no arquivo /etc/portsentry/portsentry.conf

```
# Newer versions of Linux support the reject flag now. This
# is cleaner than the above option.
KILL_ROUTE="/sbin/route add -host $TARGET$ reject"
```

E tirar o comentário da que dá a opção de DROP com o iptables

```
# iptables support for Linux
KILL_ROUTE="/sbin/iptables -I INPUT -s $TARGET$ -j DROP"
#
```

+ Então, reiniciamos o portsentry:

```
service portsentry restart
```

+ Quando verificamos os serviços rodando nas portas abertas, podemos ter noção da existência de um IPS

```
root@pentest:~/Desktop
Host is up (0.0030s latency).
Not shown: 983 closed ports
PORT      STATE SERVICE      VERSION
1/tcp      open  tcpwrapped
22/tcp     open  ssh          OpenSSH 8.0p1 Debian 4 (protocol 2.0)
79/tcp     open  tcpwrapped
80/tcp     open  http         Apache httpd 2.4.41 ((Debian))
111/tcp    open  tcpwrapped
119/tcp    open  tcpwrapped
143/tcp    open  tcpwrapped
1080/tcp   open  tcpwrapped
1524/tcp   open  tcpwrapped
2000/tcp   open  tcpwrapped
6667/tcp   open  tcpwrapped
12345/tcp  open  tcpwrapped
31337/tcp  open  tcpwrapped
32771/tcp  open  tcpwrapped
32772/tcp  open  tcpwrapped
32773/tcp  open  tcpwrapped
32774/tcp  open  tcpwrapped
MAC Address: 00:0C:29:13:01:FA (VMware)
Service Info: OS: Linux CPE: http://cpe.mitre.org
```

+ Para habilitar, no portsentry, a opção de bloqueio, basta que modifiquemos a

seguinte conf:

```
# 1 = Block UDP/TCP scan  
# 2 = Run external command  
  
BLOCK_UDP="1"  
BLOCK_TCP="1"  
  
#####  
# Dropping Routes:  
#
```

+ Para ver os hosts bloqueados, basta acessar o arquivo /etc/hosts.deny

## Bypass IDS/IPS/FW

+ O objetivo aqui é passar apenas a ideia do que ocorrerá no próximo bloco

+ Essencialmente, o bypass se dará quando fizermos uma multidão de endereços IP bater nas portas a ponto de o vigia não conseguir focar no atacante

+ Como se fosse uma multidão de pessoas mechendo na porta e apenas uma delas com as ferramentas certas para abrí-la

## Evadindo Mecanismos de Defesa

+ Dados os princípios apontados no módulo passado, vejamos o modo pelo qual o nmap permite que façamos requisições com ips aleatórios (random)

```
nmap -v -D RND:25 -sS --top-ports=25 --open -Pn 192.168.0.11
```

## LAB - SEM 06 - Mecanismos de Defesa

### LAB01: 443

+ Aqui fizemos uma busca pelas 1000 principais portas no nmap e encontramos um arquivo na internet que tivesse todas elas: o nome era "nmap-top-ports.txt"

+ Executamos então uma varredura com o nmap somente com essas portas especificadas no arquivo (A varredura porta à porta da 1 à 65535) demorou demais. Portanto, foi melhor usar as portas comuns

```
for porta in $(cat nmap-top-ports.txt);  
do echo "TESTANDO A PORTA $porta";  
nmap -v -sS -Pn -g $porta 172.16.1.59 | grep "Discovered";  
done
```

→ Lembrando que a opção -g especifica a porta de origem

```
TESTANDO A PORTA 443 Scanning 172.16.1.5 [1000 ports]
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan time
s may be slower.
Discovered open port 8080/tcp on 172.16.1.59
Discovered open port 22/tcp on 172.16.1.59
```

+ Como disse, o teste porta a porta também funcionou, mas demorou demais

```
for porta in $(seq 1 65535);
do;
nmap -v -sS -Pn -g $porta 172.16.1.59 | grep "Discovered";
echo "TENTATIVA DA PORTA $porta";
done
```

```
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan time
s may be slower.
Discovered open port 8080/tcp on 172.16.1.59
Discovered open port 22/tcp on 172.16.1.59
```

### LAB02: 120537778998

+ Com o resultado do lab passado, fizemos a conexão via netcat enviando uma porta de origem (sim, o netcat tem essa opção), fizemos uma requisição do tipo GET e enviamos para um arquivo xml em /var/www/html/recon.html (recon.html é o nome do arquivo criado)

```
nc -vn -p 443 172.16.1.59 8080 > /var/www/html/recon.html
```

+ Depois, iniciamos nosso serviço do apache

```
service apache2 start
```

+ Por fim, acessamos no navegador o endereço 192.168.1.138/recon.html



### LAB03: 80,2222,10000

Como o objetivo é fazer o mínimo de barulho possível, então usar a opção que manda IP's aleatórios como -D RND:20 não é a mais furtiva.  
A melhor opção é que façamos o teste com as --top-ports e subamos a qtd pouco a pouco

```
nmap -Pn -sS --top-ports=350 intranet.businesscorp.com.br --open
```

```
[root@DESKTOP-NJHHNK6] - [/home/kali]
# nmap -Pn -sS --top-ports=350 intranet.businesscorp.com.br --open
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-25 17:50 -03
Nmap scan report for intranet.businesscorp.com.br (37.59.174.228)
Host is up (0.18s latency).
rDNS record for 37.59.174.228: ip228.ip-37-59-174.eu
Not shown: 344 closed tcp ports (reset), 3 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE
80/tcp    open  http
2222/tcp   open EtherNetIP-1
10000/tcp  open  snet-sensor-mgmt
```

#### LAB04: debian4

Quando fazemos a varredura dos serviços com a opção `-V` do nmap, vemos que o serviço rodando na porta 2222 é o ssh

```
nmap -v -sSV -p 80,2222,10000 intranet.businesscorp.com.br
```

```
PORT      STATE SERVICE VERSION
80/tcp    open  http      Apache httpd 2.2.22 ((Debian))
2222/tcp   open  ssh       OpenSSH 6.0p1 Debian 4+deb7u6 (protocol 2.0)
10000/tcp  open  http      MiniServ 0.01 (Webmin httpd)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Portanto, se fizermos a interação com o netcat pela porta 2222, vamos capturar o banner do OS:

```
nc -v intranet.businesscorp.com.br 2222
```

De onde recebemos o mesmo banner do acusado pelo nmap e concluimos ser um Debian 4

#### LAB05: Apache/2.2.22

Basta ver o lab4

#### LAB06: openssh6.0p1

Basta ver o lab4  
o serviço de acesso à distância é o ssh

#### LAB07: miniserv/0.01

Basta ver o lab4

#### LAB08: publickey,password

Ao pesquisarmos na internet, achamos a seguinte requisição que se propõe a fazer o que buscamos

```
nmap --script ssh-auth-methods intranet.businesscorp.com.br
```

```
| ssh-auth-methods:
|   Supported authentication methods:
|     publickey
|_    password
```

# Trabalhando com Scapy

## Introdução ao Scapy

- + Ferramenta usada para manipulação de pacotes
- + Ele usa o interpretador do python, por isso consegue ser uma ferramenta que manipula pacotes de maneira interativa

The screenshot shows a terminal window with the following content:

```
(root@DESKTOP-NJHHNK6)-[~/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
          ayp ayyyyyyySCP//Pp      syY//c
          AYAsAYYYYYYYYY///Ps      cY//S
          pCCCCY//p                cSSps y//Y
          SPPPP///a                pp///AC//Y
          A//A                     cyP///C
          p///Ac                  sC///a
          P///YCpc                A//A
          scccccp///pSP///p      p//Y
          sY/////////y caa        S//P
          cayCyayP//Ya            pY/Ya
          sY/PsY///YCc            aC//Yp
          sc  sccaCY//PCypaapycP//YSs
          spCPY//////YPSPs  6.48%
          ccaacs
          Welcome to Scapy
          Version 2.5.0
          https://github.com/secdev/scapy
          Have fun!
          To craft a packet, you have to be a
          packet, and learn how to swim in
          the wires and in the waves.
          -- Jean-Claude Van Damme
          CONTEÚDO EM VÍDEO
using IPython 8.14.0
```

The terminal window has a watermark "Trabalhando com Scapy" across it. The title bar shows "Novo Pentest Profissional". The bottom right corner shows a menu bar with "File Edit View Search Terminal Help" and a progress bar at 6.48%.

- + Para ver a lista de comandos, digitamos `lsc()`
- + Para ter um help sobre algum comando, digitamos `help(comando)`
- + Se digitarmos só `ls()`, ele vai mostrar todos os protocolos suportados
- + Para dar detalhes do protocolo, `ls(protocolo)`

# **Manipulando Pacotes de Rede**

+ Vamos montar o pacote IP, que vamos chamar de pIP (Escolhemos um nome aleatório)

```
>>> pIP = IP(dst="192.168.0.1")
```

→ com isso declaramos o ip de destino. Poderíamos também declarar o ip de origem, bastava fazer

```
>>> pIP = IP(dst="192.168.0.1", src="192.168.1.148")
```

→ Como não digitamos nada anteriormente, ele já admite por padrão o nosso endereço como sendo o de origem.

+ Para mostrar mais informações, temos as seguintes opções:

```
>>> pIP.show()
```

```
>>> pIP.summary()
```

+ Montaremos agora o nosso pacote TCP:

```
>>> pTCP = TCP(dport=80, flags="S")
```

```
>>> pTCP → Para ver informações básicas do pacote
```

```
>>> pTCP.show()
###[ TCP ]###
sport= ftp_data
dport= http
seq= 0
ack= 0
dataofs= None
reserved= 0
flags= S
window= 8192
chksum= None
urgptr= 0
options= []
```

+ Podemos determinar a porta de origem:

```
>>> pTCP.sport=43778
```

+ Para encapsular o pacote TCP dentro do IP, basta:

```
>>> pacote = pIP/pTCP
```

```
>>> pacote = pIP/pTCP
>>> pacote
<IP frag=0 proto=tcp dst=192.168.0.1 |<TCP sport=43778 dport=http flags=S |>
>>> pacote.show()
###[ IP ]###
version= 4
ihl= None
tos= 0x0
len= None
id= 1
flags=
frag= 0
ttl= 64
proto= tcp
chksum= None
src= 192.168.0.11
dst= 192.168.0.1
\options\
###[ TCP ]###
sport= 43778
dport= http
seq= 0
ack= 0
dataofs= None
reserved= 0
flags= S
window= 8192
checksum= None
urgptr= 0
options= []
I
```

→ isso é semelhante ao wireshark

+ Para enviar o pacote:

```
>>> sr1(pacote)
```

+ Para enviar a saída do envio para uma determinada variável chamada “resposta”

```
>>> resposta = sr1(pacote)
```

+ Podemos setar as informações específicas que queremos também

→ Se quisermos, por exemplo, o ip de destino

```
>>> resposta[IP].dst
```

+ Se quisermos fazer mais pacotes (tipo, iguais que ataquem portas diferentes), basta adicionarmos o [] que indica o range que vamos determinar.

→ No exemplo a seguir, vamos fazer o TCP mandar pacotes para as portas 80,443,8080

```
→ >>> pTCP = TCP(dport=[80,443,8080])
```

→ + Depois, encapsulamos novamente o pacote para enviá-lo

```
→ >>> pacote = pIP/pTCP
```

+ Para pegar a resposta do envio do pacote, gravar os resultados em resp e os que não deram resposta em noresp, basta seguir os passos abaixo

```

>>> sr(pacote)
Begin emission:
.*Finished sending 3 packets.
**
Received 4 packets, got 3 answers, remaining 0 packets
(<Results: TCP:3 UDP:0 ICMP:0 Other:0>, <Unanswered: TCP:0 UDP:0 ICMP:0 Other:0>)
>>> resp, noresp = sr(pacote)
Begin emission:
Finished sending 3 packets.
*** Received 3 packets, got 3 answers, remaining 0 packets
>>> resp.summary()
IP / TCP 192.168.0.11:ftp_data > 192.168.0.1:http S ==> IP / TCP 192.168.0.1:http > 192.168.0.11:ftp_data
SA / Padding
IP / TCP 192.168.0.11:ftp_data > 192.168.0.1:https S ==> IP / TCP 192.168.0.1:https > 192.168.0.11:ftp_da
ta SA / Padding
IP / TCP 192.168.0.11:ftp_data > 192.168.0.1:http_alt S ==> IP / TCP 192.168.0.1:http_alt > 192.168.0.11:
ftp_data RA / Padding
>>> resp.show()
0000 IP / TCP 192.168.0.11:ftp_data > 192.168.0.1:http S ==> IP / TCP 192.168.0.1:http > 192.168.0.11:ftp_
_data SA / Padding
0001 IP / TCP 192.168.0.11:ftp_data > 192.168.0.1:https S ==> IP / TCP 192.168.0.1:https > 192.168.0.11:f
tp_data SA / Padding
0002 IP / TCP 192.168.0.11:ftp_data > 192.168.0.1:http_alt S ==> IP / TCP 192.168.0.1:http_alt > 192.168.

```

## Criando Pacotes ICMP e Payloads

+ Para enviar um pacote icmp dentro do pacote ip que criamos na aula passada, basta executarmos:

```

>>> pacote = pIP/ICMP()
>>> resposta = sr1(pacote)
Begin emission:
.Finished sending 1 packets.
*
Received 2 packets, got 1 answers, remaining 0 packets
>>> resposta

```

+ Podemos ver o encapsulamento pelo wireshark:

Screenshot of Wireshark showing two ICMP Echo requests and replies between 192.168.0.11 and 192.168.0.1.

| No. | Time        | Source       | Destination  | Protocol | Length Info                                                     |
|-----|-------------|--------------|--------------|----------|-----------------------------------------------------------------|
| 19  | 4.820118555 | 192.168.0.11 | 192.168.0.1  | ICMP     | 42 Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 20) |
| 20  | 4.825252109 | 192.168.0.1  | 192.168.0.11 | ICMP     | 60 Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 19) |

```

> Frame 20: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
> Ethernet II, Src: ArrisGro_45:c4:8c (d4:ab:82:45:c4:8c), Dst: VMware_13:01:fa (00:0c:29:13:01:fa)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.11
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSSCP: CS0, ECN: Not-ECT)
  Total Length: 28
  Identification: 0xbdc8 (48587)
> Flags: 0x0000
  Time to live: 64
  Protocol: ICMP (1)
  Header checksum: 0x3b69 [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.168.0.1
  Destination: 192.168.0.11
- Internet Control Message Protocol
  Type: 8 (Echo (ping) reply)
  Code: 0
  Checksum: 0xffff [correct]
  [Checksum Status: Good]

```

+ Para fazer um pacote com payload, podemos também

```
>>> pacote = pIP/ICMP()/"DESEC SECURITY"
>>> resposta = srl(pacote)
```

Screenshot of Wireshark showing two ICMP Echo requests and replies between 192.168.0.11 and 192.168.0.1, with a payload "DESEC SECURITY".

| No. | Time        | Source       | Destination  | Protocol | Length Info                                |
|-----|-------------|--------------|--------------|----------|--------------------------------------------|
| 16  | 3.377567073 | 192.168.0.11 | 192.168.0.1  | ICMP     | 56 Echo (ping) request id=0x0000, seq=0/0, |
| 17  | 3.380293372 | 192.168.0.1  | 192.168.0.11 | ICMP     | 60 Echo (ping) reply id=0x0000, seq=0/0,   |

```

> Frame 16: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 0
> Ethernet II, Src: VMware_13:01:fa (00:0c:29:13:01:fa), Dst: ArrisGro_45:c4:8c (d4:ab:82:45:c4:8c)
> Internet Protocol Version 4, Src: 192.168.0.11, Dst: 192.168.0.1
- Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0xe017 [correct]
  [Checksum Status: Good]
  Identifier (BE): 0 (0x0000)
  Identifier (LE): 0 (0x0000)
  Sequence number (BE): 0 (0x0000)
  Sequence number (LE): 0 (0x0000)
  [Response frame: 17]
+ Data (14 bytes)
  Data: 4445534543205345435552495459
  [Length: 14]

0000 d4 ab 82 45 c4 0c 00 0c 29 13 01 fa 08 00 45 00 ...E.... )....E.
0010 00 2a 00 01 00 00 40 01 f9 75 c0 a8 00 0b c0 a8 *...@. 0...
0020 00 01 08 00 e0 17 00 00 00 00 44 45 53 45 43 20 ..... .DESEC
0030 53 45 43 55 52 49 54 59 SECURITY

```

## **Criando um Portscan com o Scapy**

+ Objetivo: Aprender a montar scripts em python com o scapy

```
#!/usr/bin/python3
import sys
from scapy.all import *

conf.verb = 0

portas = [21,22,23,25,80,443,8080]

pIP = IP(dst=sys.argv[1])
pTCP = TCP(dport=portas,flags="S")
pacote = pIP/pTCP
resp, noresp = sr(pacote)
for resposta in resp:
    porta = resposta[1][TCP].sport
    flag = resposta[1][TCP].flags
    if (flag == "SA"):
        print (f"Porta {porta} aberta")
```

+ Esse conf.verb = 0 serve somente para tornar mais limpa a saída do script

+ No resposta[1], o [1] indica que é o pacote recebido em que estamos fazendo a análise

Se quiséssemos pegar o enviado, bastaria trocar por 0

## **Ping Scan com Scapy**

```
#!/usr/share/python3

from scapy.all import *
conf.verb = 0
print ("Hosts Vivos")
for ip in range(1, 255):
    iprange = "192.168.0." + str(ip)
    pIP = IP(dst=iprange)
    pacote = pIP/ICMP()
    resp, noresp = sr(pacote,timeout=1)
    for resposta in resp:
        catulo = resposta[1][IP].src
        print (catulo)
```