



Universidade do Minho
Escola de Engenharia

TECNOLOGIA DE SEGURANÇA

TP2 – Parte B

PenTest - Scanning

Diogo Araújo A78485; Diogo Nogueira A78957

Conteúdo

1. Contextualização.....	5
2. Parte 1 - NMap e Wireshark.....	6
2.1 Questão 1	6
2.1.1. Modo de <i>Scan</i> -sn (No Port Scan)	6
2.1.2. Modo de <i>Scan</i> -sU (UDP Scan)	8
2.1.3. Modo de <i>Scan</i> -sT (TCP Connect Scan)	13
2.1.4. Modo de <i>Scan</i> -sS (TCP SYN Scan)	17
2.2 Questão 2	20
2.2.1. Modo de <i>Scan</i> -sA (TCP ACK Scan)	20
2.2.2. Modos de <i>Scan</i> -sF, -sN e -sX (TCP FIN, NULL e Xmas Scan)	22
2.3 Questão 3	26
2.4 Questão 4	27
3. Parte 2 – OpenVAS/ <i>Nessus</i> e <i>Snort</i>	30
3.1. Questão 5	30
3.2 Questão 6	31
3.3 Questão 7	33
3.4 Questão 8	35
3.4.1. HTTP TRACK/TRACE Methods Allowed	35
3.4.2. rexecd Service Detection	38
3.4.3. Bind Shell Backdoor Detection	42
3.4.4. VNC Server ‘password’ Password	42
4. Conclusões e Observações Finais.....	46
5. Referências.....	47

Índice de Figuras

Figura 1. Captura Tráfego e Output do modo de scan -sn	7
Figura 2. Output do modo de scan -sU	8
Figura 3. Captura Tráfego de uma porta TPC Aberta do modo de scan -sU	9
Figura 4. Captura Tráfego de uma porta TPC Aberta Filtrada do modo de scan -sU 10	
Figura 5. Output do modo de scan -sU com a flag -V e -T	11
Figura 6. Captura Tráfego de uma porta TPC Fechada do modo de scan -sU	12
Figura 7. Output do modo de scan -sT	13
Figura 8. Captura Tráfego de uma porta TPC Aberta do modo de scan -sT	14
Figura 9. Conexão TCP com a porta 80 da Máquina Virtual Metasploitable 2 no modo de scan -sT	15
Figura 10. Captura Tráfego de uma porta TPC Fechada do modo de scan -sT	16
Figura 11. Conexão TCP com a porta 143 da Máquina Virtual Metasploitable 2 no modo de scan -sT	16
Figura 12. Output do modo de scan -sS	17
Figura 13. Captura Tráfego de uma porta TPC Aberta do modo de scan -sS	18
Figura 14. Conexão TCP com a porta 80 da Máquina Virtual Metasploitable 2 no modo de scan -sS	18
Figura 15. Captura Tráfego de uma porta TPC Fechada do modo de scan -sS	19
Figura 16. Conexão TCP com a porta 143 da Máquina Virtual Metasploitable 2 no modo de scan -sS	19
Figura 17. Ping do servidor com endereço IP 45.33.32.156	20
Figura 18. Captura Tráfego do Servidor com Endereço IP 45.33.32.156 do modo de scan -sA e respetivo output	21
Figura 19. Captura Tráfego da Máquina Virutal Metasploitable 2 do modo de scan -sA e respetivo output	22
Figura 20. Captura Tráfego do Servidor com Endereço IP 45.33.32.156 do modo de scan -sF e respetivo output	23
Figura 21. Output do modo de scan -sF feito à Máquina Virutal Metasploitable 2 ...	24
Figura 22. Output do modo de scan -sF com a flag -V e -T feito à Máquina Virutal Metasploitable 2	25
Figura 23. Scanning Ativo à Máquina Virtual Metasploitable 2	26
Figura 24. Output da deteção de versões dos serviços das portas da Máquina Virutal Metasploitable 2	27
Figura 25. Detalhes da vulnerabilidade de CVE-2019-10164 correspondente ao serviço PostgreSQL	28

Figura 26. Detalhes da vulnerabilidade de CVE-2015-2575 correspondente ao serviço MySQL	28
Figura 27. Detalhes da vulnerabilidade de CVE-2019-10098 correspondente ao serviço Apache HTTP Server	29
Figura 28. Contagem das vulnerabilidades da Máquina Virtual Metasploitable 2 e listagem de algumas delas.....	30
Figura 29. Excerto do output do Snort para a vulnerabilidade Bind Shell Backdoor Detection/Detalhes do Nessus sobre a mesma.....	31
Figura 30. Excerto do output do pacote TCP vindo da Máquina Virtual Metasploitable 2/Tráfego Wireshark.....	32
Figura 31. Detalhes do Nessus para a tentativa de descobrir a password e o root directory do TFTP Server/Tráfego Wireshark.....	33
Figura 32. Excerto do output do Snort correspondente aos pacotes UDP enviados em broadcast	34
Figura 33. Excerto do output do Snort correspondente a um comando ping.....	34
Figura 34. Resultado do OpenVAS da vulnerabilidade HTTP Debugging Methods Enabled	36
Figura 35. Verificação do serviço (e versão) que corre na porta 80 da Máquina Virtual Metasploitable 2	36
Figura 36. Gráficos do scan inicial e do scan após correção da vulnerabilidade correspondentes aos resultados das vulnerabilidades por nível de gravidade.....	37
Figura 37. Lista das vulnerabilidades do scan inicial e do scan após correção da vulnerabilidade	38
Figura 38. Resultado do OpenVAS da vulnerabilidade rexecd Service Detection	39
Figura 39. Verificação do serviço (e versão) que corre na porta 512 da Máquina Virtual Metasploitable 2	40
Figura 40. Gráficos do scan inicial e do scan após correção da vulnerabilidade correspondentes aos resultados das vulnerabilidades por nível de gravidade.....	41
Figura 41. Lista das vulnerabilidades do scan inicial e do scan após correção da vulnerabilidade	41
Figura 42. Resultado do OpenVAS da vulnerabilidade VNC Brute Force Login	43
Figura 43. Comandos no terminal que demonstram a resolução da vulnerabilidade.....	44
Figura 44. Gráficos do scan inicial e do scan após correção da vulnerabilidade correspondentes aos resultados das vulnerabilidades por nível de gravidade.....	44
Figura 45. Lista das vulnerabilidades do scan inicial e do scan após correção da vulnerabilidade	45

1. Contextualização

A ideia base deste trabalho prático passa por agrupar toda a aprendizagem obtida até então, na tentativa de compreender a importância da mesma na globalidade da segurança da informação.

Começamos por entender como eram classificados os vários tipos de vulnerabilidades, a sua gravidade e o impacto que causam. Após isso, criamos todo um modelo de ameaça, compreendo melhor a importância de se estudar o funcionamento de um sistema e de que forma podemos idealizar uma boa segurança para o mesmo. Passamos à parte de Coleta Passiva de Informações, pela análise de detalhes divulgados publicamente, vendo até que ponto isso poderia ser útil para um possível início de ataque e terminamos agora na forma mais crua e agressiva de executar/estudar uma invasão com a Coleta Ativa de Informações.

Esta Coleta Ativa de Informações distingue-se do restante, na medida em que envolve o “contacto” direto com o *target* em causa, algo que vamos comprovar no desenvolver deste relatório.

Para este último trabalho prático em específico foram usadas ferramentas mundialmente conhecidas como o NMap, o *Snort* e o *Nessus/OpenVAS*:

- **NMap:** Pelo NMap (*Network Mapper*) foi possível explorar a rede, determinar os *hosts* disponíveis, quais os serviços que esses oferecem, entre outras características;
- ***Snort* e *Nessus/OpenVAS*:** Pelo *Snort* e pelo *Nessus/OpenVAS* conseguimos obter todo um resultado de vulnerabilidades do sistema *target*, o que nos permitiu analisar o tráfego existente durante o *scan* e resolver algumas das vulnerabilidades apresentadas por estes.

Além destas ferramentas, usou-se como auxílio o **Wireshark**, que permitiu estudar os pacotes que circulavam pela rede no decorrer dos *scans*, possibilitando assim um estudo mais detalhado e fruto de melhores conclusões.

2. Parte 1 - NMap e Wireshark

Através de um ambiente de testes devidamente preparado, pretende-se efetuar um *scanning* ativo através de uma Máquina Virtual Kali Linux tendo como *target* de *scan* a Máquina Virtual Metasploitable 2.

A ferramenta de segurança NMap a usar nesta primeira parte surge essencialmente com o objetivo de reconhecer sistemas e serviços na rede de dados. Com um vasto leque de funcionalidades, esta ferramenta vai-nos permitir analisar um conjunto de informações que serão importantes na análise do tráfego que circula entre as duas Máquinas Virtuais e aquilo que em si o envolve. Esse tráfego será capturado pelo Wireshark.

Assim, através do uso simultâneo de ambas as ferramentas, será possível estudar melhor os métodos de *scan* que o NMap oferece, tentando compreender a sua importância na área da segurança informática, concretamente ao nível da rede.

2.1 Questão 1

2.1.1. Modo de *Scan -sn* (No Port Scan)

Através da pesquisa na documentação oficial da ferramenta NMap, ficamos a saber que este modo de *scan* significa “sem varredura de porta”, ou seja, não é efetuado um *scan* ao nível das portas existentes e das suas respetivas informações.

No modo padrão de *scan*, o NMap executa dois tipos de varredura:

1. *Scan* de *Host* – determinação dos *hosts* disponíveis;
2. *Scan* de *Porta* – divulgação do estado das portas nos *hosts* disponíveis.

Ao executarmos este modo de *scan*, estamos apenas a realizar uma *scan* de *hosts*, extraindo unicamente as informações relativas aos *hosts* disponíveis que deram resposta ao teste da descoberta do *scan* efetuado. Basicamente, é o mesmo que fazer-se *ping*, confirmando a existência de *hosts* e os seus respetivos endereços MAC, sem que sejam reveladas as suas portas.

Neste exercício em específico, verifica-se que o resultado do *scan* para o *target* 172.16.3.2 (Máquina Virtual Metasploitable 2) devolve o endereço MAC desta mesma máquina.

Através do Wireshark fica também visível o tipo de interação que está efetivamente a ocorrer por detrás. Existe um pedido ARP (*Address Resolution Protocol*), cuja funcionalidade é solicitar o endereço MAC de uma determinada máquina através do seu endereço IP. Assim, existe um *reply* à Máquina Virtual Kali Linux que contém no corpo deste pacote ARP o endereço MAC que se está a tentar solicitar.

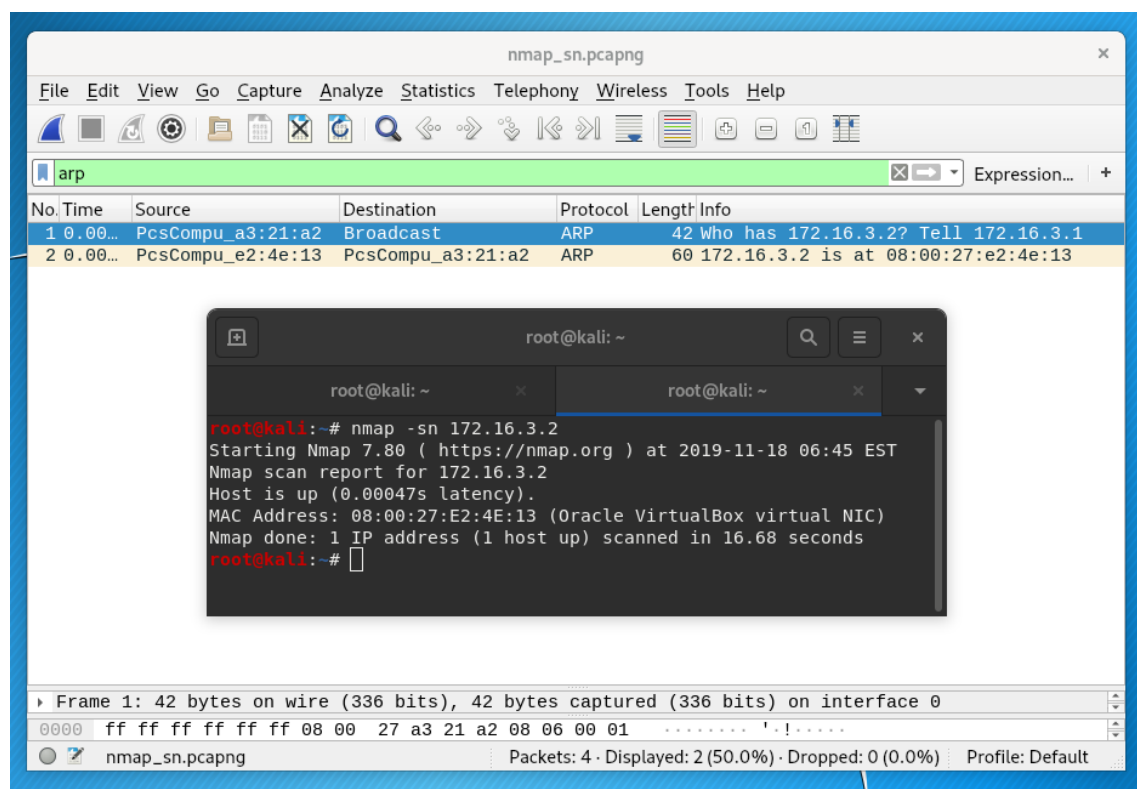
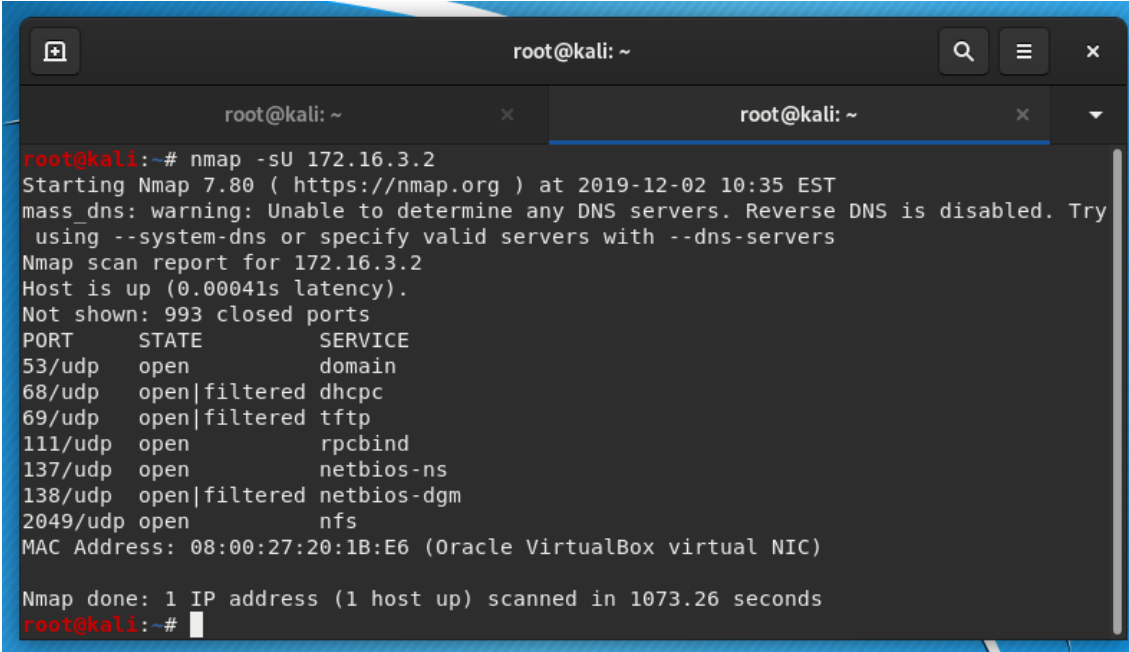


Figura 1. Captura Tráfego e Output do modo de scan -sn

2.1.2. Modo de *Scan -sU* (UDP Scan)

Ao executarmos este comando notamos logo uma grande diferença em termos de tempo de execução. Isso fica ainda mais perceptível quando se realizam os *scans* das alíneas seguintes, e aí compreende-se a diferença na dificuldade de um *scan* UDP comparativamente ao TCP em si.

Aproximadamente 15 minutos após a execução do comando, obtemos assim a lista das portas UDP. Algumas são dadas como *open*, outras como *open/filtered* estando as restantes listadas como *close*.

A terminal window titled 'root@kali: ~' showing the output of the command 'nmap -sU 172.16.3.2'. The output includes the Nmap version (7.80), a warning about DNS, the scan report for 172.16.3.2, a list of open and filtered ports with their corresponding services, and the total scan time (1073.26 seconds).

```
root@kali: ~
root@kali: ~
root@kali: # nmap -sU 172.16.3.2
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-02 10:35 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try
using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 172.16.3.2
Host is up (0.00041s latency).
Not shown: 993 closed ports
PORT      STATE      SERVICE
53/udp    open       domain
68/udp    open|filtered dhcpc
69/udp    open|filtered tftp
111/udp   open       rpcbind
137/udp   open       netbios-ns
138/udp   open|filtered netbios-dgm
2049/udp  open       nfs
MAC Address: 08:00:27:20:1B:E6 (Oracle VirtualBox virtual NIC)
Nmap done: 1 IP address (1 host up) scanned in 1073.26 seconds
root@kali: #
```

Figura 2. Output do modo de *scan -sU*

Para se compreender como funciona o *scan* em si, vai ser analisado todo o tráfego capturado pelo Wireshark, fazendo as distinções em termos de estado de portas. Dessa forma, conseguimos perceber o que distingue cada um deles e como é que o NMap interpreta as respostas dadas pelo UDP.

1. Porta UDP Aberta

O UDP *scan* baseia-se no envio de um pacote UDP para todas as portas de destino existentes. No caso de uma porta UDP Aberta, a resposta que é dada pelo *probe* é algo incomum, podendo corresponder a qualquer tipo de resposta UDP por parte da porta de destino. Através da Figura 3 conseguimos validar essa ideia, dado que se verifica um pacote UDP de resposta que não é comum de uma resposta em UDP padrão.

Com uma pesquisa extra, apuramos a informação de que esta porta número 137 faz parte do protocolo TCP/UDP, mais propriamente do serviço *NetBIOS Name Service* (que consta na Figura 3). Sabe-se ainda que este serviço usa tipicamente o protocolo UDP como o seu protocolo de transporte, ficando assim justificada a resposta dada pelo *probe* para esta porta.

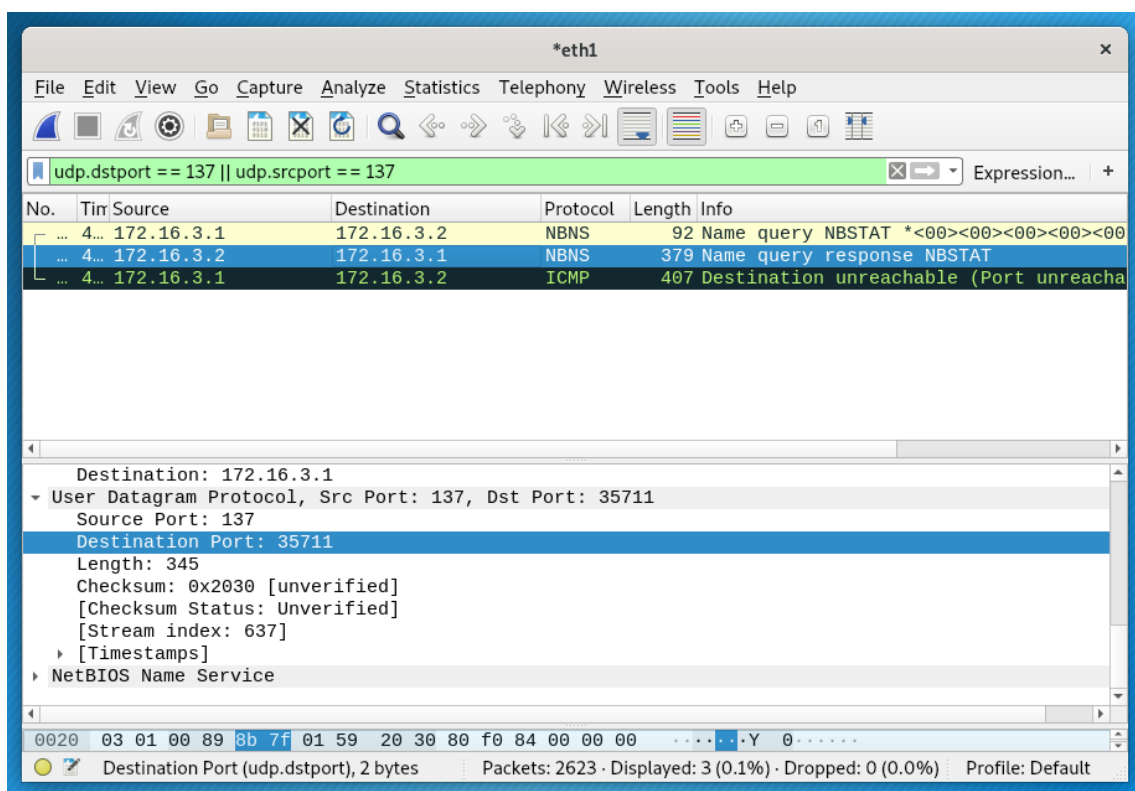


Figura 3. Captura Tráfego de uma porta TPC Aberta do modo de scan-sU

2. Porta UDP Aberta | Filtrada

O grande problema de um UDP *scan* é precisamente o facto de se criar uma nuance relativamente ao estado da porta e o tempo que demora a fazer todo o *scan*. Fala-se das portas que são dadas como *open/filtered* e que criam uma ambiguidade, dado que acaba por não se saber qual o verdadeiro estados dessas mesmas portas.

Pelo *output* da Figura 2 vimos que 3 (porta número 68, 69 e 1138) das 1000 portas analisadas não foram bem concluídas. Isto acontece quando não existe qualquer tipo de resposta por parte da porta de destino, mesmo após várias retransmissões do pacote UDP. A Figura 4 retrata exatamente essa ideia de retransmitir várias vezes o mesmo pacote e a não existência de uma resposta mesmo após todas essas tentativas.

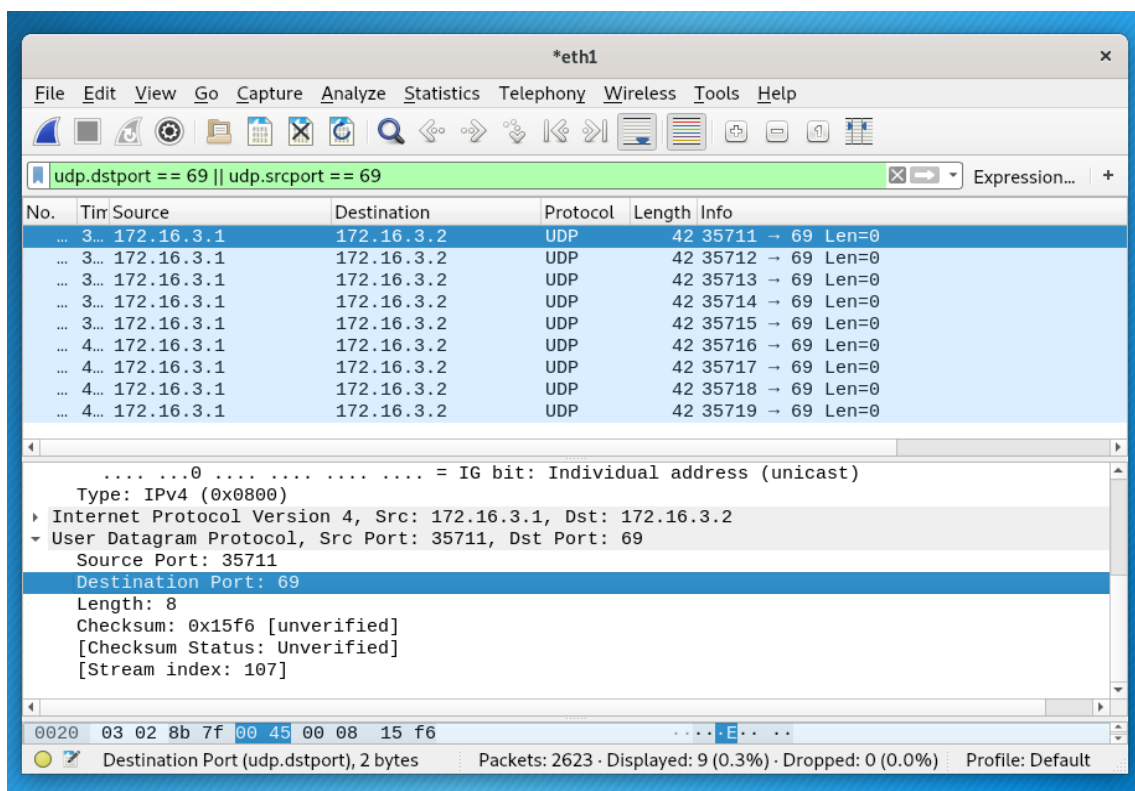


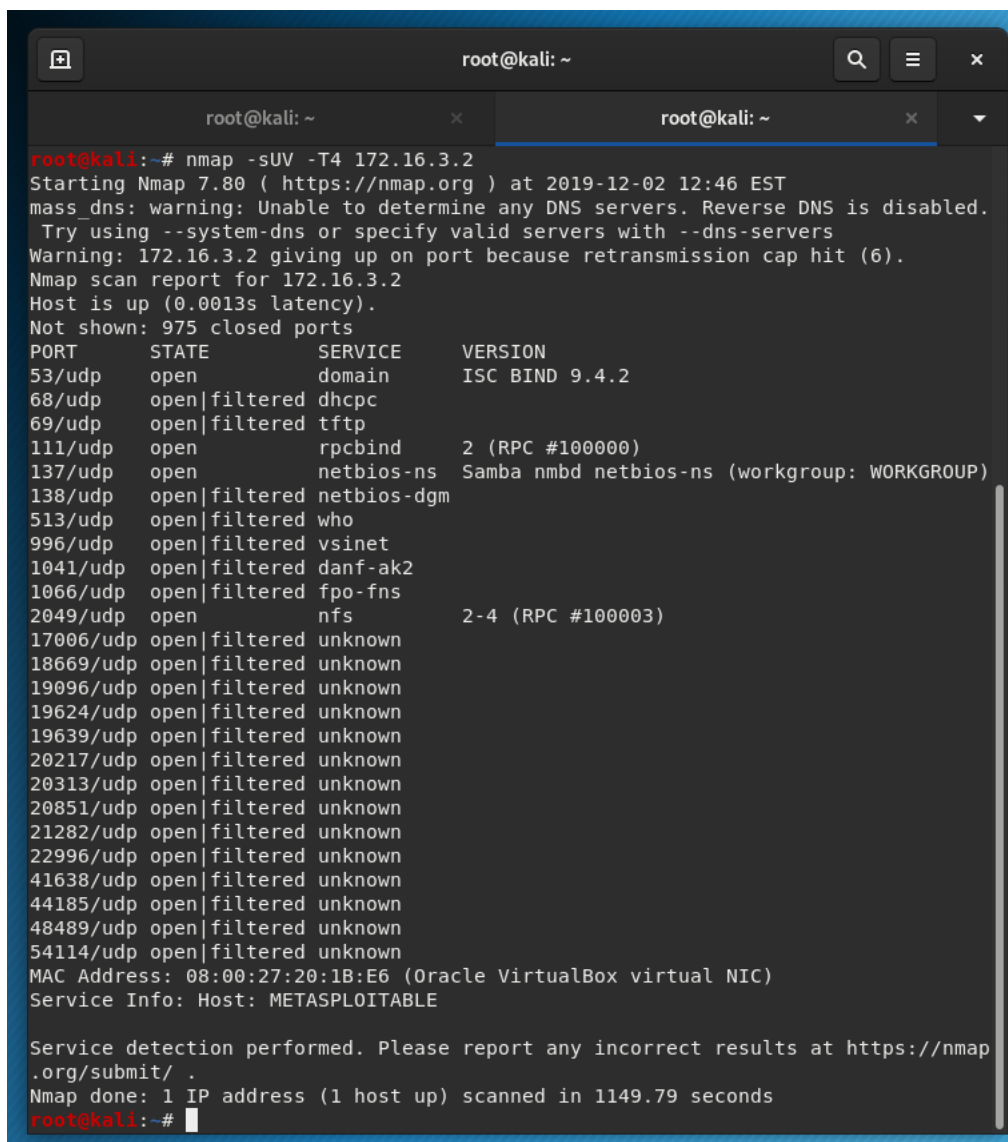
Figura 4. Captura Tráfego de uma porta TPC Aberta | Filtrada do modo de *scan -sU*

Tendo em conta que não conseguimos obter uma certeza em relação às portas número 68, 69 e 138, o suposto agora é tentar arranjar uma solução que consiga desmitificar estas três.

Pela pesquisa feita no manual oficial do comando NMap, constatamos a existência de métodos que podem diminuir esta ambiguidade, mostrando assim um novo estado para estas portas.

Dessa forma, adicionamos as seguintes *flags*:

- *Flag-sUV*: permite que seja exibida a informação acerca do serviço da porta e da versão ativa para o mesmo;
- *Flag-T*: permite definir o modelo de tempo (que quando mais alto for, melhor será o resultado).



```
root@kali: ~  
root@kali:~# nmap -sUV -T4 172.16.3.2  
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-02 12:46 EST  
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.  
Try using --system-dns or specify valid servers with --dns-servers  
Warning: 172.16.3.2 giving up on port because retransmission cap hit (6).  
Nmap scan report for 172.16.3.2  
Host is up (0.0013s latency).  
Not shown: 975 closed ports  
PORT      STATE      SERVICE      VERSION  
53/udp    open      domain      ISC BIND 9.4.2  
68/udp    open|filtered dhcpc  
69/udp    open|filtered tftp  
111/udp   open      rpcbind      2 (RPC #100000)  
137/udp   open      netbios-ns   Samba nmbd netbios-ns (workgroup: WORKGROUP)  
138/udp   open|filtered netbios-dgm  
513/udp   open|filtered who  
996/udp   open|filtered vsinet  
1041/udp  open|filtered danf-ak2  
1066/udp  open|filtered fpo-fns  
2049/udp  open      nfs          2-4 (RPC #100003)  
17006/udp open|filtered unknown  
18669/udp open|filtered unknown  
19096/udp open|filtered unknown  
19624/udp open|filtered unknown  
19639/udp open|filtered unknown  
20217/udp open|filtered unknown  
20313/udp open|filtered unknown  
20851/udp open|filtered unknown  
21282/udp open|filtered unknown  
22996/udp open|filtered unknown  
41638/udp open|filtered unknown  
44185/udp open|filtered unknown  
48489/udp open|filtered unknown  
54114/udp open|filtered unknown  
MAC Address: 08:00:27:20:1B:E6 (Oracle VirtualBox virtual NIC)  
Service Info: Host: METASPLOITABLE  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 1149.79 seconds  
root@kali:~#
```

Figura 5. Output do modo de scan *-sU* com a flag *-Ve-T*

Esta nova execução visa assim melhorar os resultados anteriores, devolvendo um novo *output* no que toca ao estados das portas no geral. Como se sabe que o modelo de tempo pode variar de 0 até 5, fizemos apenas teste para um modelo de tempo de 4, o que adiciona um *timeout* de 5 minutos por cada *host* e nunca espera mais que 1.25 segundos para testar as respostas.

Contrariamente ao que se esperava, não conseguimos obter um estado exato das três portas mencionadas. Ainda assim, existiu uma redução de cerca de 18 portas desde o estado *close* até ao estado *open/filtered*, o que é uma mudança bastante significativa ainda que continue incerta.

3. Porta UDP Fechada

Quando a porta UDP se encontra no estado *close*, a resposta dada é sempre um erro ICMP do Tipo 3. Na Figura 6 vimos o envio do pacote UDP inicial e a resposta em modo pacote ICMP que determina logo que a porta em si se encontra fechada.

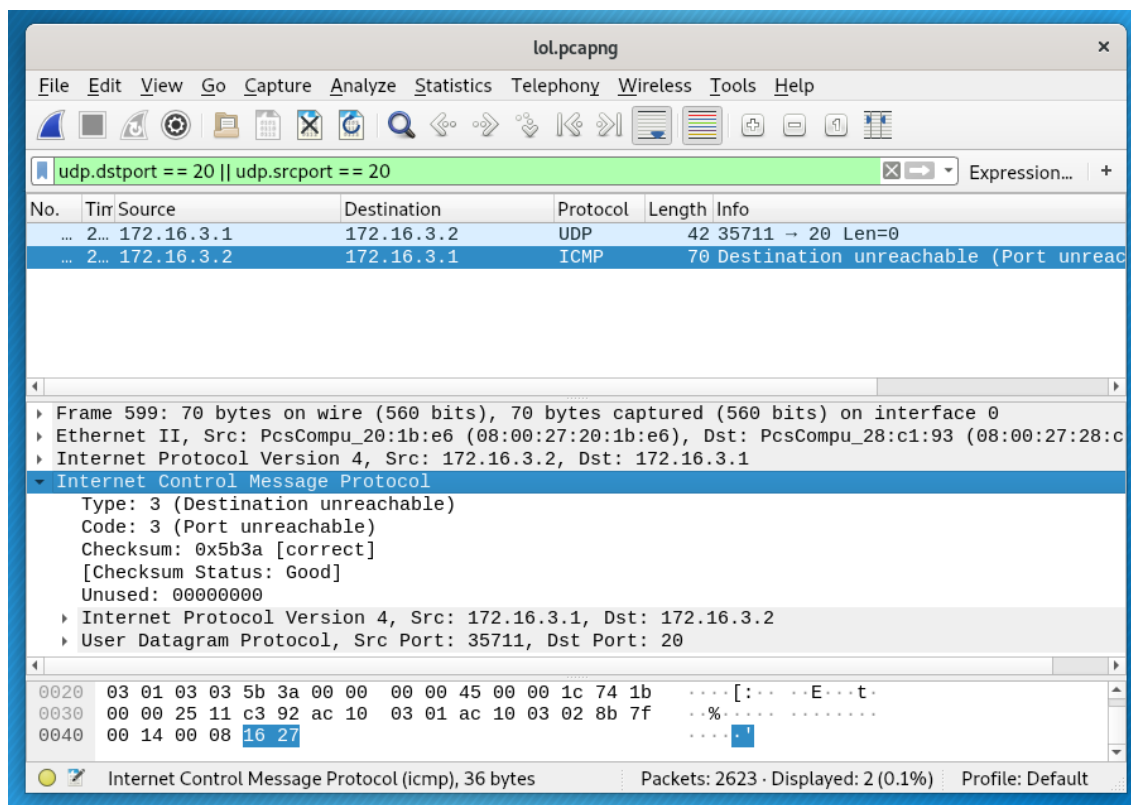
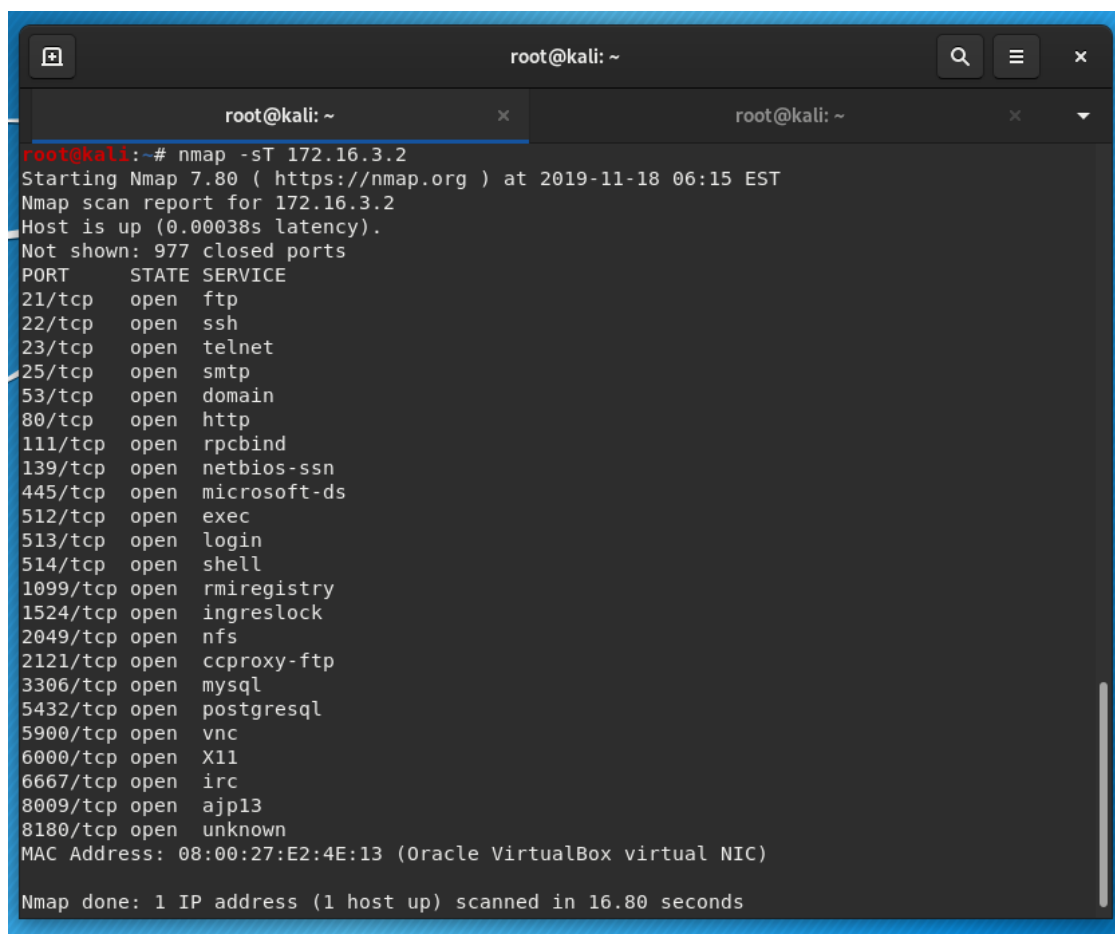


Figura 6. Captura Tráfego de uma porta TPC Fechada do modo de scan -sU

2.1.3. Modo de *Scan -sT* (TCP Connect Scan)

Este comando faz uma espécie de descoberta de portas TCP existentes, pelo uso da verificação padrão de conexão do protocolo TCP. Mediante o *output* dado pelo NMap, é possível herdar uma lista de todas as portas TCP abertas com a referência ao número de cada uma e o serviço que usam. É ainda informada a existência de mais 977 portas, mas que se encontram fechadas para uso na rede.



```
root@kali: ~  
root@kali:~# nmap -sT 172.16.3.2  
Starting Nmap 7.80 ( https://nmap.org ) at 2019-11-18 06:15 EST  
Nmap scan report for 172.16.3.2  
Host is up (0.00038s latency).  
Not shown: 977 closed ports  
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  
2121/tcp  open  ccproxy-ftp  
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  
MAC Address: 08:00:27:E2:4E:13 (Oracle VirtualBox virtual NIC)  
Nmap done: 1 IP address (1 host up) scanned in 16.80 seconds
```

Figura 7. Output do modo de scan -sT

Tal como foi feito na alínea anterior, é através do tráfego Wireshark que vamos apreender o processo de troca de pacotes/mensagens, validando assim o estado de cada porte em análise.

1. Porta TCP Aberta

A Figura 8 mostra a captura de tráfego referente à verificação de conexão da porta 80. Identifica-se imediatamente o processo comum do estabelecimento de uma conexão TCP e o seu posterior encerramento, dado que existe toda a troca de pacotes padrão que dizem respeito ao protocolo TCP estudado mais a fundo no âmbito de outras Unidades Curriculares.

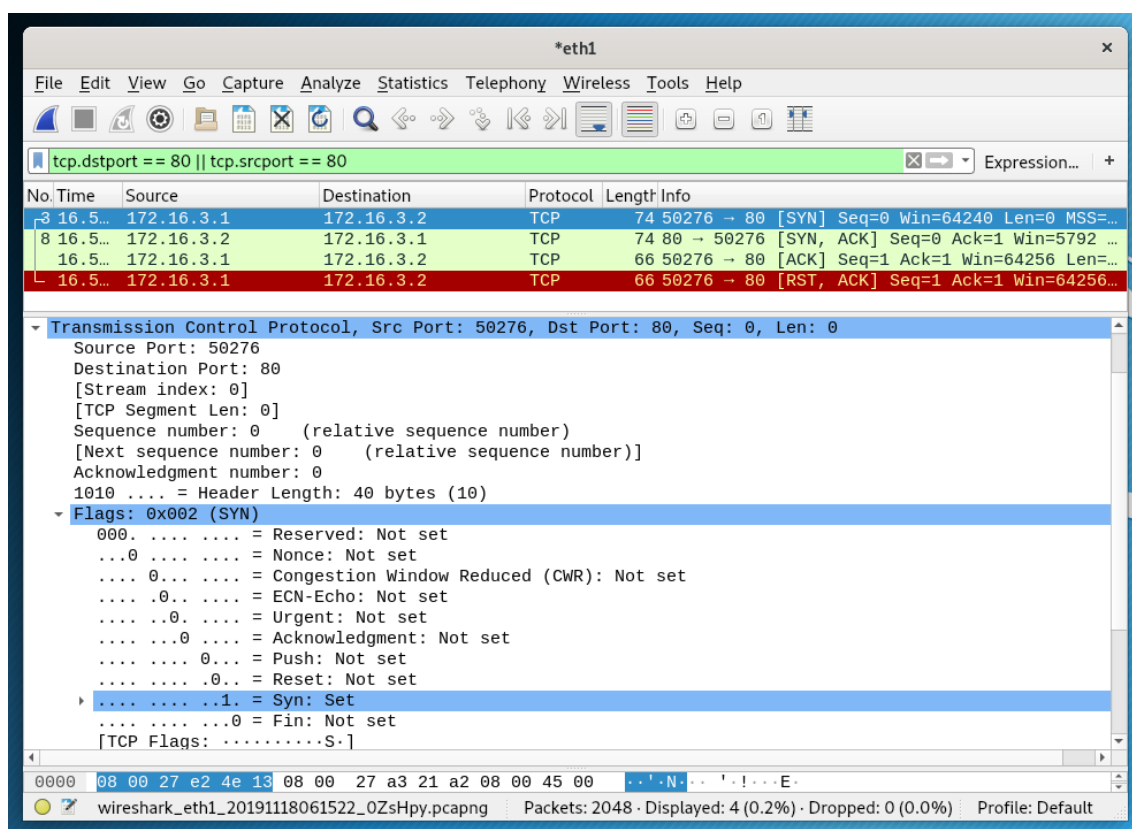


Figura 8. Captura Tráfego de uma porta TPC Aberta do modo de scan -sT

A Figura 9 serve assim para facilitar a compreensão das mensagens/pacotes que são trocados quando é estabelecida uma conexão TPC no caso de uma porta se encontrar aberta.

Note-se que assim que é enviado um pacote **SYN** para a porta 80, tentando-se estabelecer a conexão, existe uma resposta por parte desta porta TCP, com o retorno de um **SYN/ACK** que indica que a mesma se encontra aberta e que por isso foi possível

iniciar a conexão corretamente. Dado que já se descobriu aquilo que o comando em causa pretende, é enviado um **RST** que permite que a conexão seja assim interrompida.

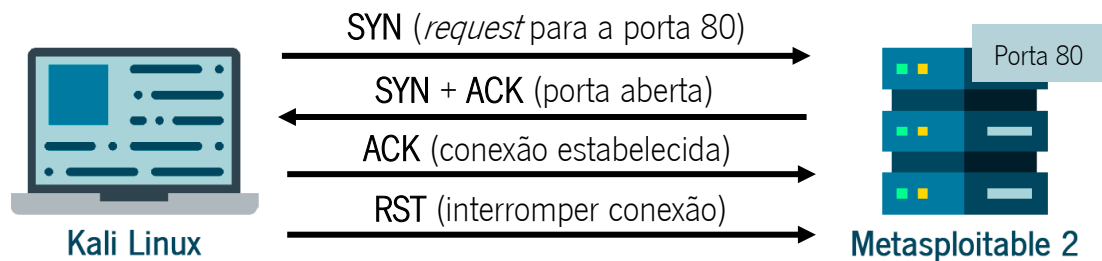


Figura 9. Conexão TCP com a porta 80 da Máquina Virtual Metasploitable 2 no modo de scan -sT

Este pacote **RST** é uma forma diferente de “terminar” a conexão em curso. O padrão de encerramento de uma conexão via TCP exige um envio de pacotes que em si acaba por ser desnecessário para a informação que se pretende obter. Para se evitar esta perda de tempo e aumento de tráfego, manda-se um pacote **RST** no meio da transferência de dados, rejeitando-se assim a comunicação adicional que teria de existir para encerrar a ligação.

2. Porta TCP Fechada

Contrariamente ao que acontece no caso de uma porta se encontrar aberta, o tráfego neste caso é visivelmente inferior. No exemplo anterior, assim que a ferramenta NMap verificava que a conexão tinha sido estabelecida, era enviado um pacote **RST** para se interromper a ligação, tendo em conta que não ia existir uma transferência de dados entre as duas portas.

Para uma porta fechada a ideia de enviar um pacote **RST** mantém-se, mudando apenas o facto de que este pacote é mandado a meio do *handshake*. Assim, quando se verifica que a conexão não está disponível, o **RST** é enviado e mais nenhuma mensagem é trocada entre as duas portas.

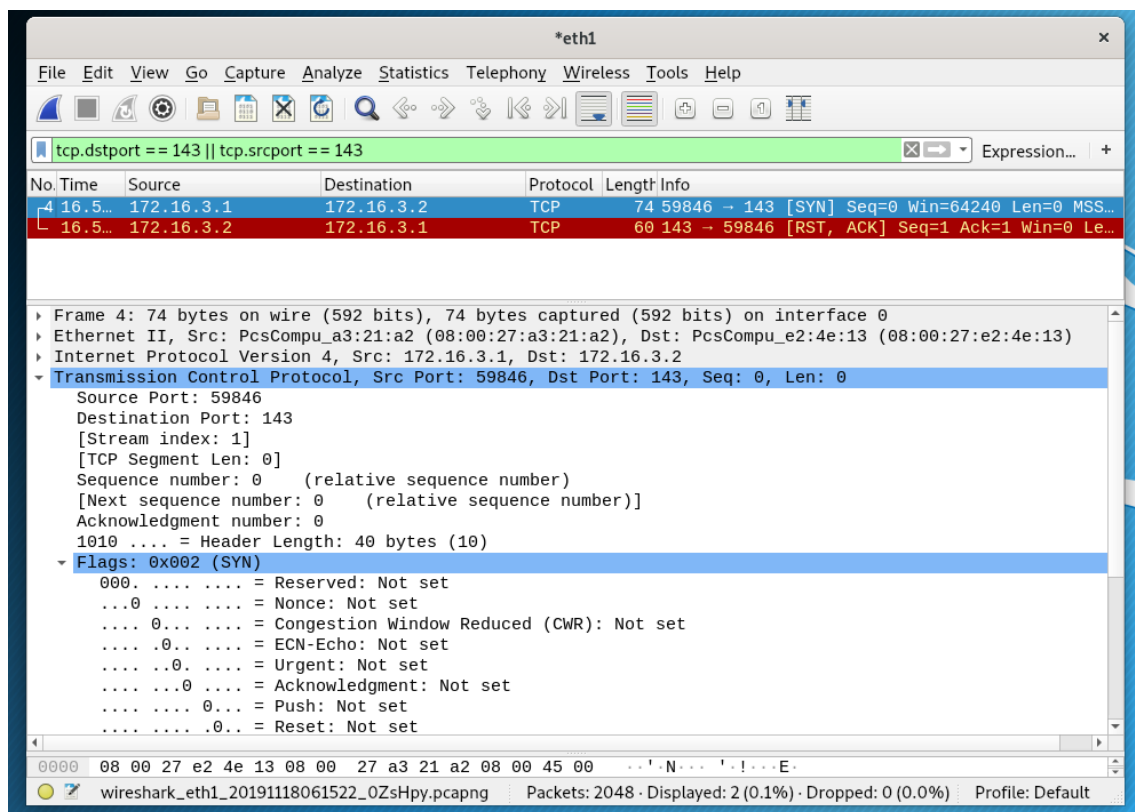


Figura 10. Captura Tráfego de uma porta TPC Fechada do modo de scan -sT

Isto simplifica o processo de compreensão de troca de pacotes, passando apenas a existir o *request* para a porta 143 e o posterior envio do **RST** por parte dessa mesma porta.

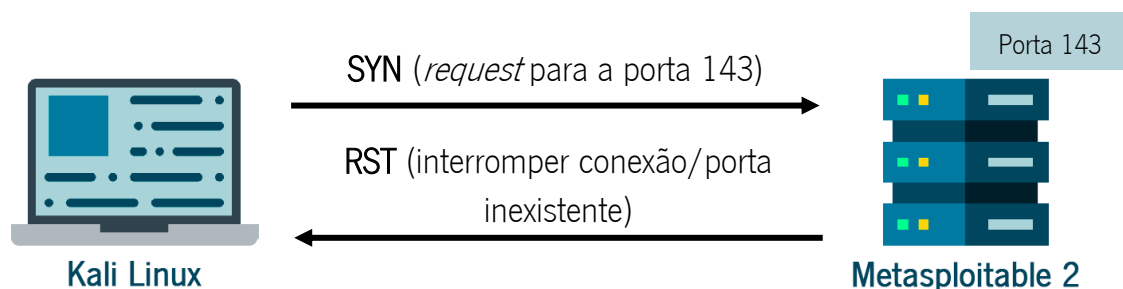
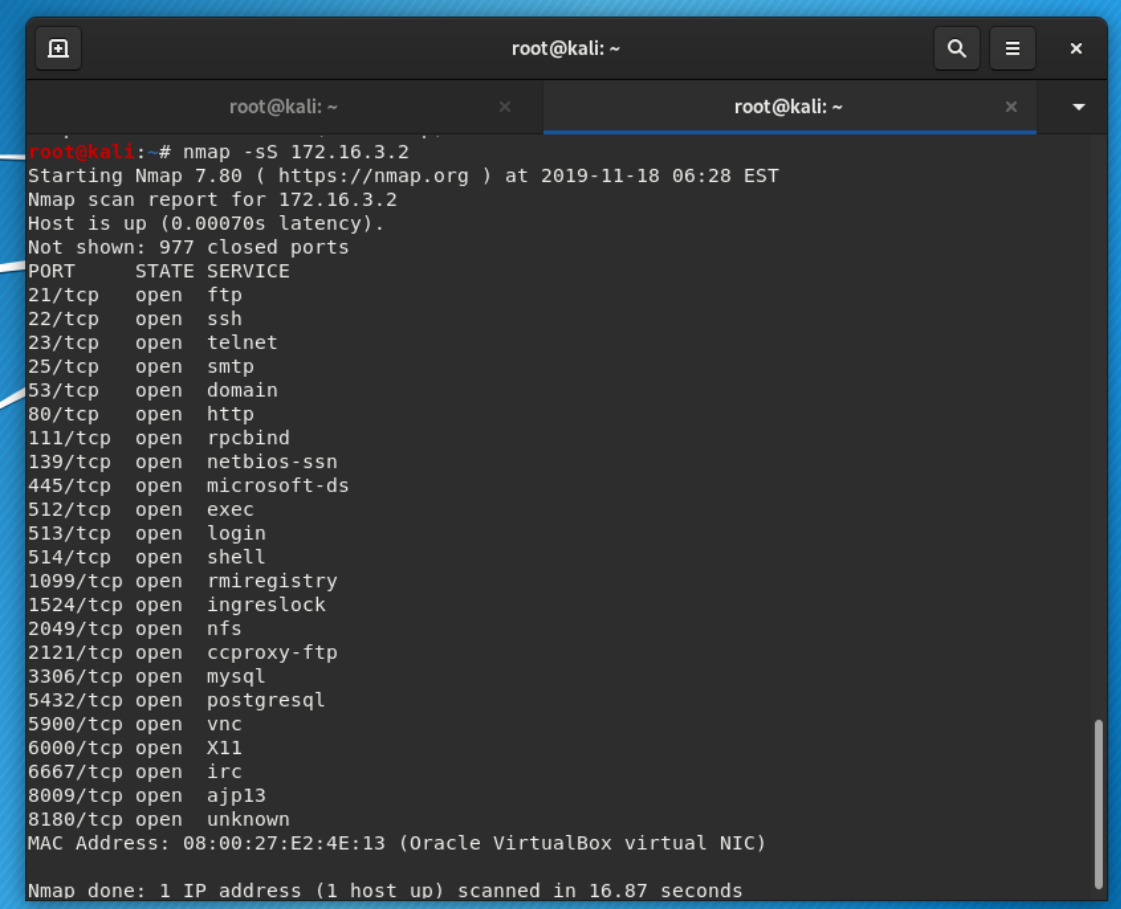


Figura 11. Conexão TCP com a porta 143 da Máquina Virtual Metasploitable 2 no modo de scan -sT

2.1.4. Modo de *Scan -sS* (TCP SYN Scan)

Este comando é similar ao comando anterior, no sentido em que faz a verificação de todas as portas TCP existentes. Contrariamente, o seu processo de descoberta dessas mesmas portas não segue o mesmo princípio, o que torna a execução do comando muito mais rápida, eficiente e adaptada a *firewalls* mais intrusivos.



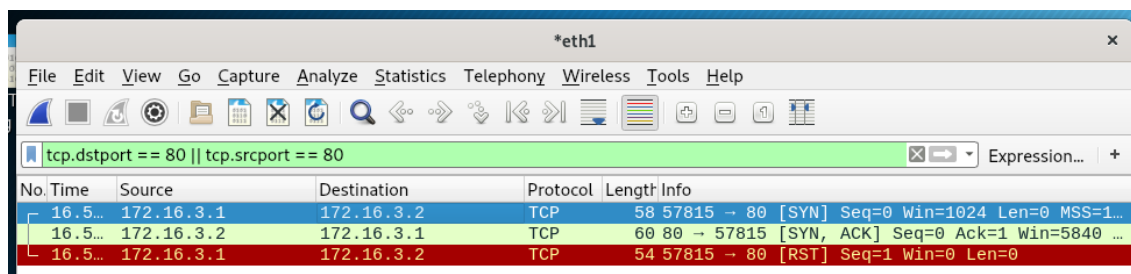
```
root@kali: ~
root@kali: ~
root@kali:~# nmap -sS 172.16.3.2
Starting Nmap 7.80 ( https://nmap.org ) at 2019-11-18 06:28 EST
Nmap scan report for 172.16.3.2
Host is up (0.00070s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
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
2121/tcp  open  ccproxy-ftp
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
MAC Address: 08:00:27:E2:4E:13 (Oracle VirtualBox virtual NIC)
Nmap done: 1 IP address (1 host up) scanned in 16.87 seconds
```

Figura 12. Output do modo de scan *-sS*

A eficácia deste modo de *scan* deve-se à quantidade de pacotes que são enviados durante a conexão TCP e a forma como é estabelecida a ideia de uma porta aberta e fechada. Em termos de tempo, pode-se dizer que o resultado apresentado em terminal demora praticamente o mesmo em ambos os *scans*, no entanto, neste modo existe uma pesquisa mais discreta/clandestina, dado que nunca é concluída a conexão TCP em ti. Esta transferência de pacotes vai ser debatida já de seguida, tanto para quando estamos perante uma porta aberta como para uma porta fechada.

1. Porta TCP Aberta

A Figura 13 demonstro logo a diminuição de troca de pacotes em relação ao modo de *scan* abordado na alínea anterior.



The image shows a Wireshark packet capture on the eth1 interface. The filter is 'tcp.dstport == 80 || tcp.srcport == 80'. The packet list shows three packets:

No.	Time	Source	Destination	Protocol	Length	Info
16.5...		172.16.3.1	172.16.3.2	TCP	58	57815 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1...
16.5...		172.16.3.2	172.16.3.1	TCP	60	80 → 57815 [SYN, ACK] Seq=0 Ack=1 Win=5840 ...
16.5...		172.16.3.1	172.16.3.2	TCP	54	57815 → 80 [RST] Seq=1 Win=0 Len=0

Figura 13. Captura Tráfego de uma porta TPC Aberta do modo de *scan -sS*

Isso fica visivelmente mais perceptível pelo diagrama da Figura 14. Conforme o diagrama mostra, começa-se com o envio de um pacote **SYN**, dado que em si este é o princípio base do protocolo TCP. De forma a saber-se que a porta está aberta, existe a resposta **SYN/ACK** por parte da porta que se está a tentar conectar.

A grande diferença em relação ao *scan* anterior acontece após esse passo. Ao invés de se terminar o *handshake* de 3 vias comum ao estabelecimento de conexão padrão do protocolo TCP, ignora-se isso. Dessa forma, fica-se na mesma a saber o estado da porta e evita-se que se estabeleça por completo a conexão, não sendo necessário depois encerrá-la.

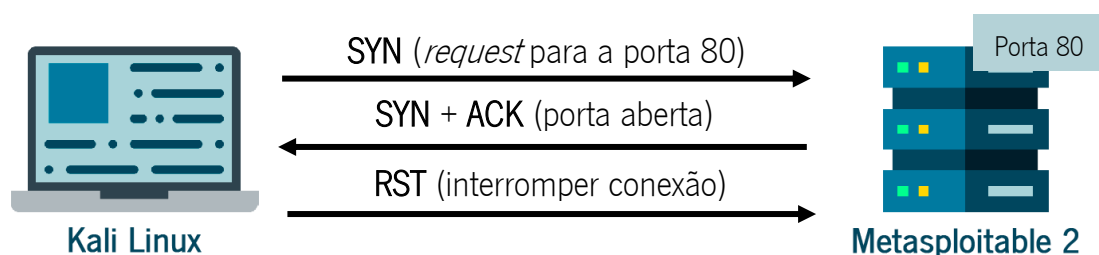
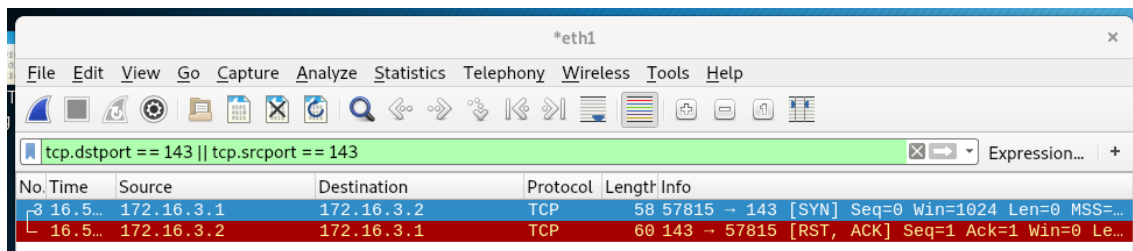


Figura 14. Conexão TCP com a porta 80 da Máquina Virtual Metasploitable 2 no modo de *scan -sS*

Para finalizar toda essa parte, envia-se um pacote **RST**, permitindo que exista o romper da conexão, evitando o envio repetido de um **SYN/ACK** até a normal existência de um pacote **ACK** de resposta.

2. Porta TCP Fechada

Quando se está perante uma porta fechada o processo é igual ao do comando anterior. Se receber de volta um **RST** significa então que a porta está encerrada, não sendo necessário existir qualquer comunicação extra com essa mesma porta.



No.	Time	Source	Destination	Protocol	Length	Info
3	16.5...	172.16.3.1	172.16.3.2	TCP	58	57815 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=...
4	16.5...	172.16.3.2	172.16.3.1	TCP	60	143 → 57815 [RST, ACK] Seq=1 Ack=1 Win=0 Le=...

Figura 15. Captura Tráfego de uma porta TCP Fechada do modo de scan -sS

A troca de pacotes é então exatamente a mesma. Um pacote **SYN** e um pacote **RST** para interromper de imediato a “conexão”.

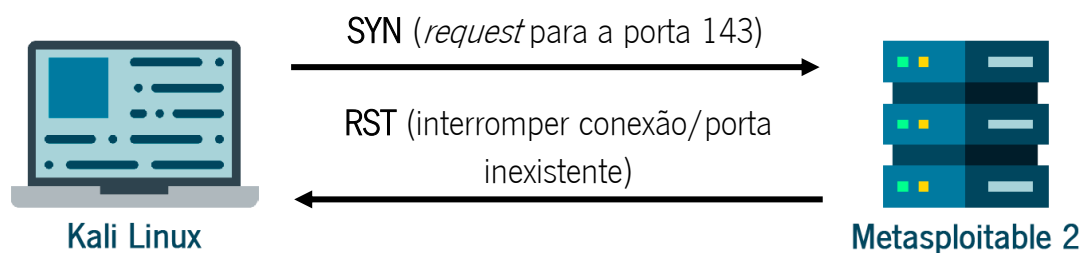
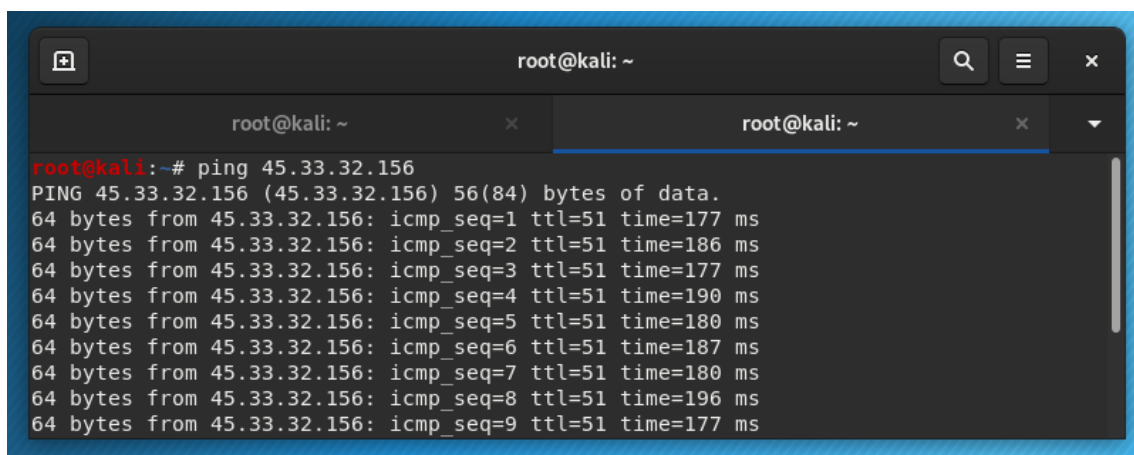


Figura 16. Conexão TCP com a porta 143 da Máquina Virtual Metasploitable 2 no modo de scan -sS

2.2 Questão 2

Tendo em conta que o servidor com endereço IP 45.33.32.156 é agora também um dos *targets*, é necessário que se verifique a sua disponibilidade em termos de sistema. Através do comando *ping* conseguimos verificar a existência de *data flow* pela rede. Desse modo, prova-se que o sistema em si se encontra ativo e pronto para ser feito todo o *scan* posterior.

A terminal window titled 'root@kali: ~' showing the execution of a ping command. The command is 'root@kali:~# ping 45.33.32.156'. The output shows 'PING 45.33.32.156 (45.33.32.156) 56(84) bytes of data.' followed by nine lines of ping results, each showing '64 bytes from 45.33.32.156: icmp_seq=X ttl=51 time=Y ms' where X ranges from 1 to 9 and Y ranges from 177 to 196.

```
root@kali:~# ping 45.33.32.156
PING 45.33.32.156 (45.33.32.156) 56(84) bytes of data.
64 bytes from 45.33.32.156: icmp_seq=1 ttl=51 time=177 ms
64 bytes from 45.33.32.156: icmp_seq=2 ttl=51 time=186 ms
64 bytes from 45.33.32.156: icmp_seq=3 ttl=51 time=177 ms
64 bytes from 45.33.32.156: icmp_seq=4 ttl=51 time=190 ms
64 bytes from 45.33.32.156: icmp_seq=5 ttl=51 time=180 ms
64 bytes from 45.33.32.156: icmp_seq=6 ttl=51 time=187 ms
64 bytes from 45.33.32.156: icmp_seq=7 ttl=51 time=180 ms
64 bytes from 45.33.32.156: icmp_seq=8 ttl=51 time=196 ms
64 bytes from 45.33.32.156: icmp_seq=9 ttl=51 time=177 ms
```

Figura 17. Ping do servidor com endereço IP 45.33.32.156

2.2.1. Modo de *Scan -sA* (TCP ACK Scan)

Contrariamente aos modos de *scan* que vimos até então, este modo de *scan* não trata de determinar a quantidade de portas abertas num determinado sistema/servidor alvo. O comando serve essencialmente para mapear um conjunto de regras de *firewall*, permitindo descobrir/detetar se um determinado está a ser protegido por uma *firewall*.

Para se testar este comando vamos ter dois *targets*, o que permitirá fazer uma comparação, compreendendo eventuais diferenças que possam existir entre ambos.

1. Servidor com Endereço IP 45.33.32.156

Para este endereço IP verificamos que todas as suas portas se encontram no modo *unfiltered*. Isto leva-nos de imediato a deduzir que este servidor alvo não se encontra protegido por uma *firewall*.

Esta conclusão pode automaticamente ser feita porque se sabe que os *firewalls* que bloqueiam o *probe* geralmente não respondem ao pacote TCP que é enviado inicialmente para cada porta neste modo de *scan*. Em contrapartida, caso exista uma *firewall* a bloquear, é enviado um pacote de erro ICMP. Isso permite que o NMap perceba se os pacotes estão a ser ou não filtrados.

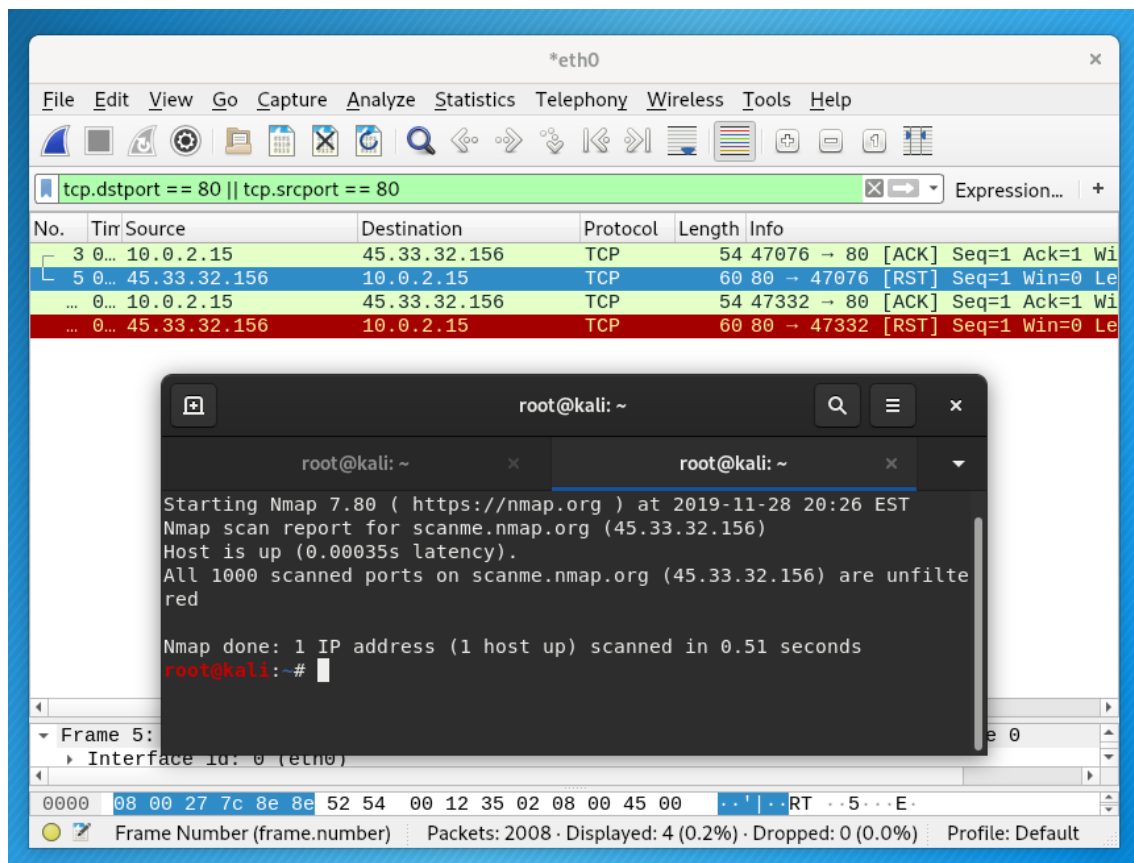


Figura 18. Captura Tráfego do Servidor com Endereço IP 45.33.32.156 do modo de scan *-sA* e respetivo output

2. Máquina Virtual Metasploitable 2

O mesmo acontece para o sistema Metasploitable 2. Das 1000 portas às quais foram feito o *scan*, todas elas estão no estado *unfiltered*.

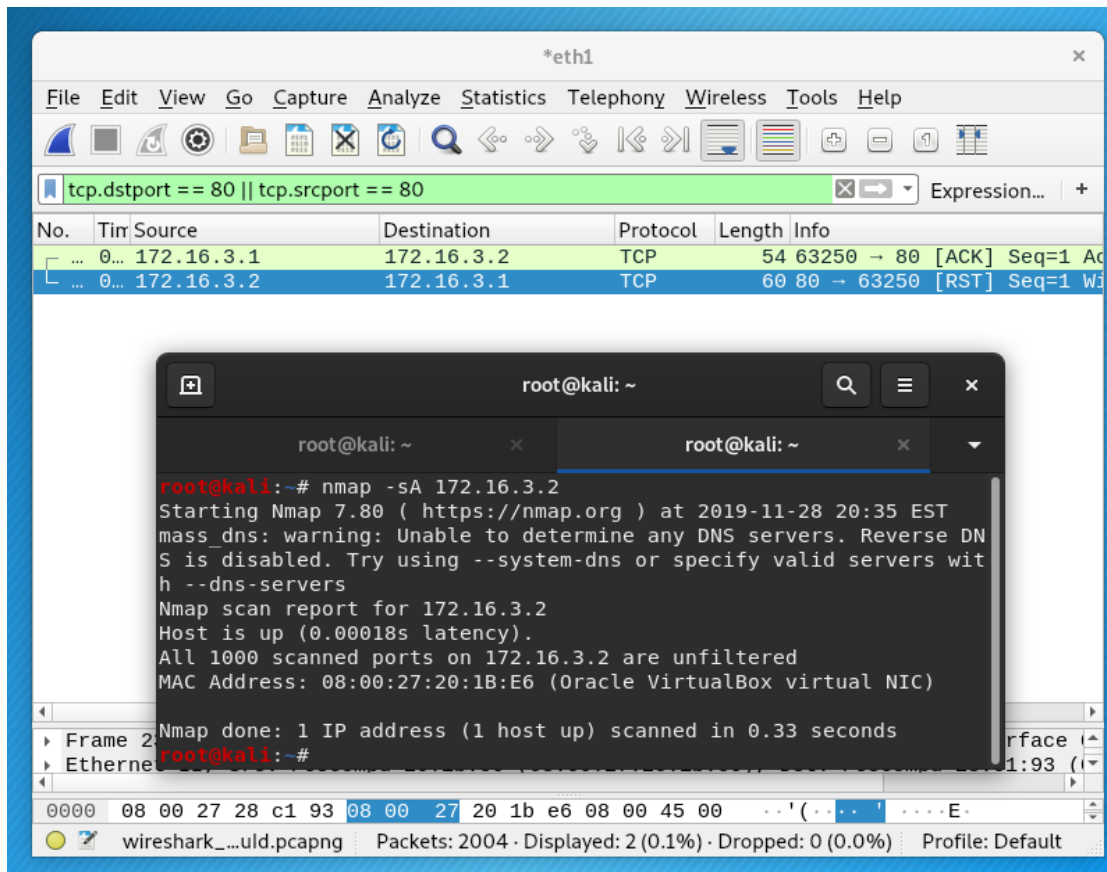


Figura 19. Captura Tráfego da Máquina Virutal Metasploitable 2 do modo de scan *-sA* e respetivo output

2.2.2. Modos de *Scan* **-sF**, **-sN** e **-sX** (TCP FIN, NULL e Xmas Scan)

O comando anterior dava-nos a possibilidade de descobrir se um determinado *target* estava a ser protegido por uma *firewall*. Os três comandos que iremos abordar a partir de agora encaixam-se todos no mesmo, dado que são três formas diferentes de procurar falhas numa *firewall*.

Este primeiro modo trata de fazer uma verificação da *firewall*, definindo o *bit* FIN do pacote TCP que é enviado para cada uma das portas. Ao contrário do comando anterior, aqui importa saber acerca do estado das portas.

1. Servidor com Endereço IP 45.33.32.156

Determina-se que todas as portas em causa se encontram fechadas. Não há muito mais a concluir. A resposta dada pelo *probe* consiste num pacote TPC RST.

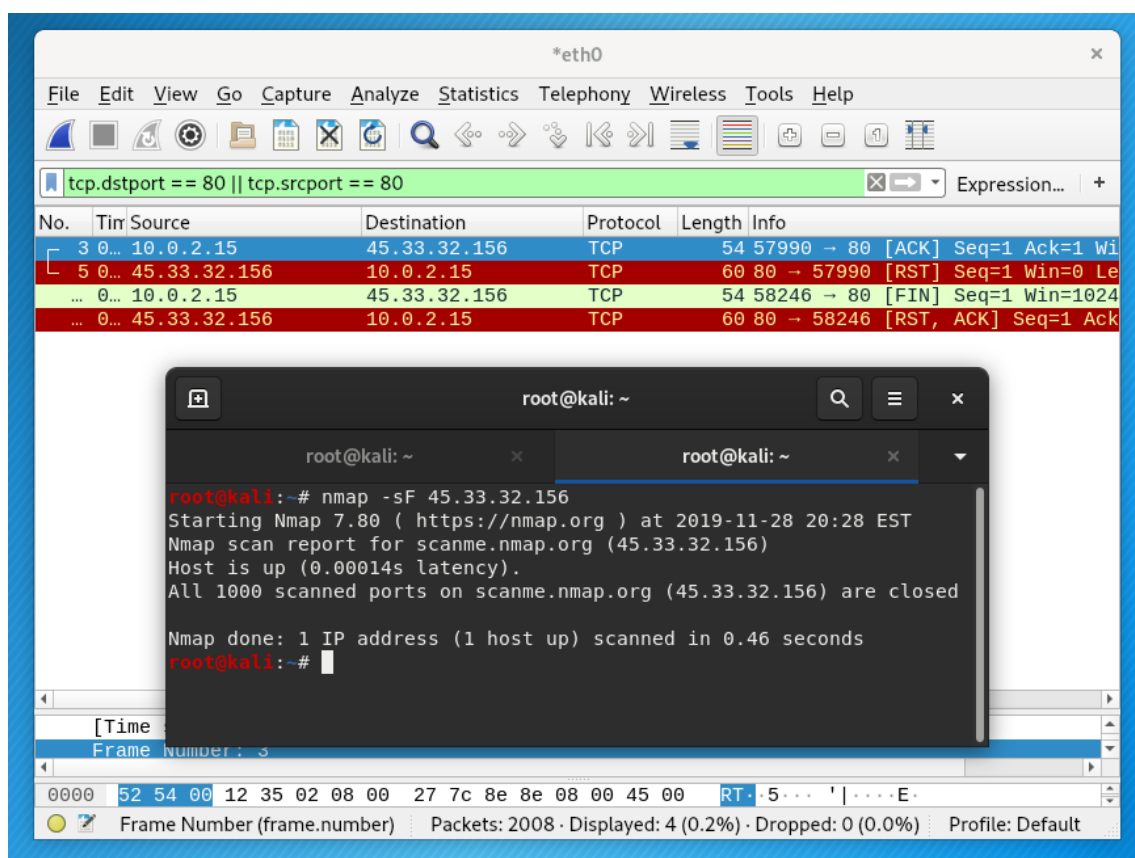
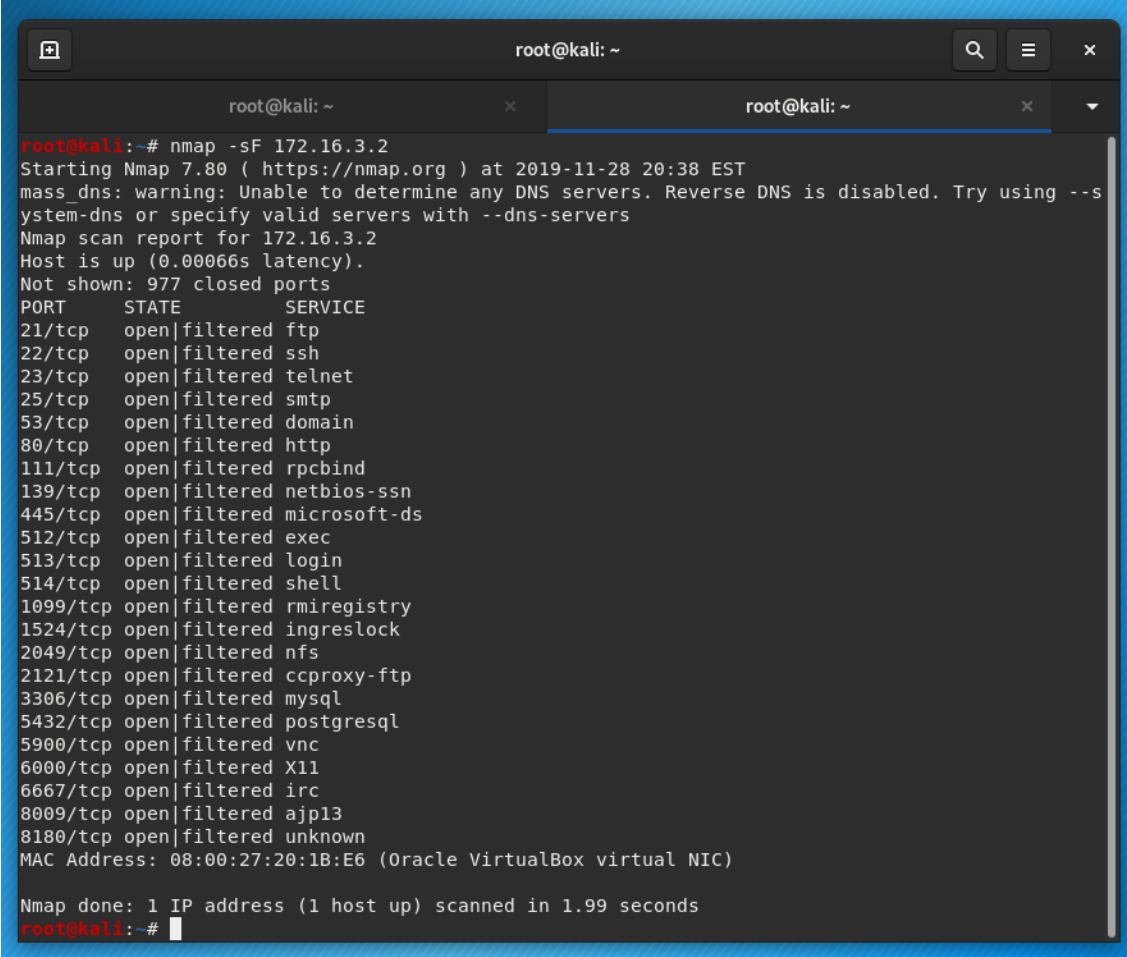


Figura 20. Captura Tráfego do Servidor com Endereço IP 45.33.32.156 do modo de scan `-sF` e respetivo output

2. Máquina Virtual Metasploitable 2

Todas as portas são definidas como *open/filtered*, à exceção das restantes 977 que são dadas como *close*.



```
root@kali: ~  
root@kali: ~  
root@kali:~# nmap -sF 172.16.3.2  
Starting Nmap 7.80 ( https://nmap.org ) at 2019-11-28 20:38 EST  
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers  
Nmap scan report for 172.16.3.2  
Host is up (0.00066s latency).  
Not shown: 977 closed ports  
PORT      STATE      SERVICE  
21/tcp    open|filtered ftp  
22/tcp    open|filtered ssh  
23/tcp    open|filtered telnet  
25/tcp    open|filtered smtp  
53/tcp    open|filtered domain  
80/tcp    open|filtered http  
111/tcp   open|filtered rpcbind  
139/tcp   open|filtered netbios-ssn  
445/tcp   open|filtered microsoft-ds  
512/tcp   open|filtered exec  
513/tcp   open|filtered login  
514/tcp   open|filtered shell  
1099/tcp  open|filtered rmiregistry  
1524/tcp  open|filtered ingreslock  
2049/tcp  open|filtered nfs  
2121/tcp  open|filtered ccproxy-ftp  
3306/tcp  open|filtered mysql  
5432/tcp  open|filtered postgresql  
5900/tcp  open|filtered vnc  
6000/tcp  open|filtered X11  
6667/tcp  open|filtered irc  
8009/tcp  open|filtered ajp13  
8180/tcp  open|filtered unknown  
MAC Address: 08:00:27:20:1B:E6 (Oracle VirtualBox virtual NIC)  
  
Nmap done: 1 IP address (1 host up) scanned in 1.99 seconds  
root@kali:~#
```

Figura 21. Output do modo de scan *-sF* feito à Máquina Virtual Metasploitable 2

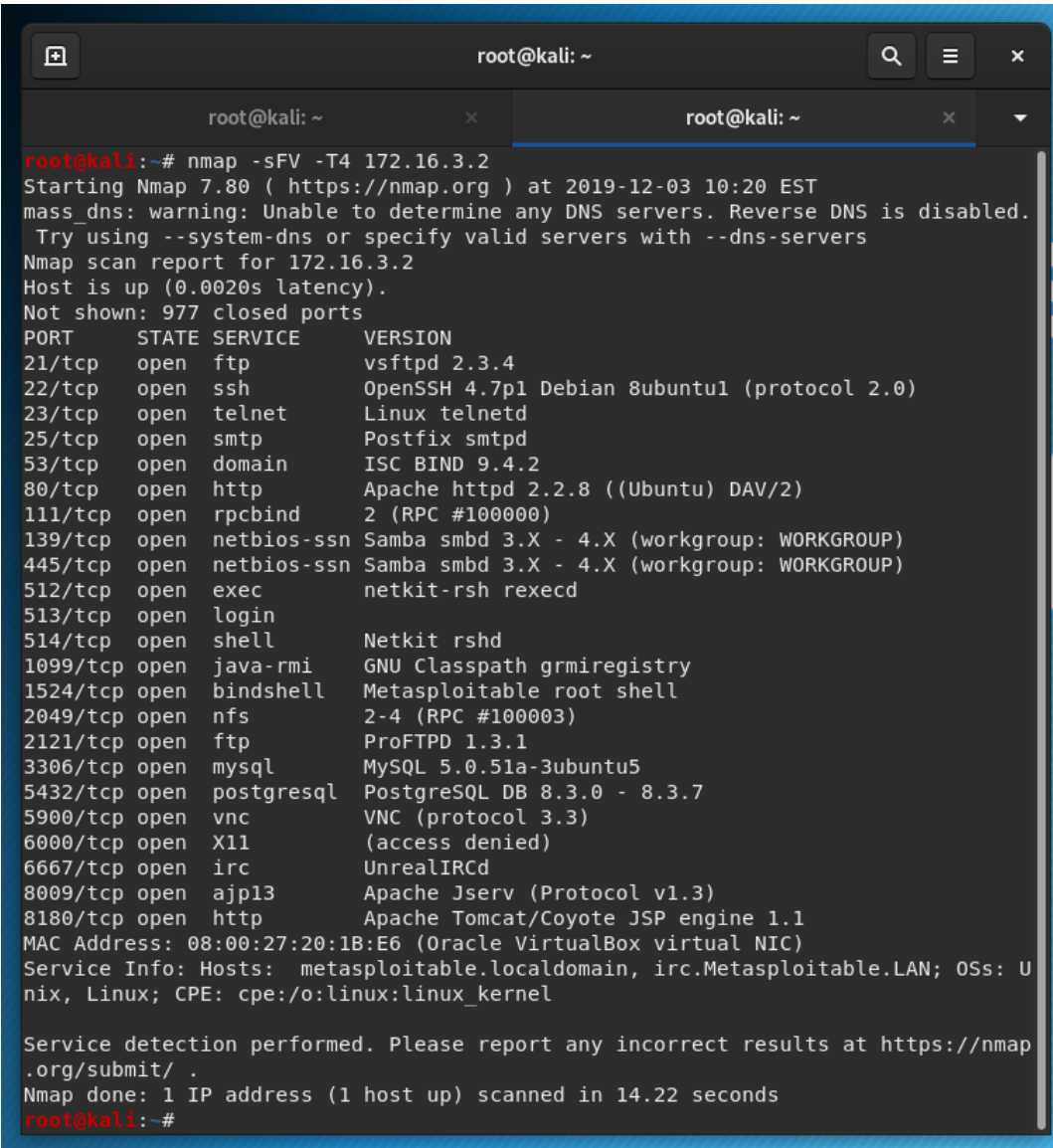
Uma desvantagem deste tipo de verificações é o facto de não serem capazes de distinguir portas abertas de determinadas portas filtradas.

Ao adicionarmos novamente as *flags* podemos diminuir a ambiguidade, como acontece com o UDP *scan*, mas isso quebra a natureza oculta que está na base deste tipo de *scans*.

Conforme fizemos para o comando `-sU`, vamos tornar a adicionar as seguintes *flags*.

- *Flag-sFV*: pedimos que o NMap determine o serviço que está a correr para cada porta;
- *Flag-T*: adicionamos um *timeout* de 5 minutos para cada *host*.

Veja-se, no entanto, a diferença de resultado. De todas as portas listadas como *open* / *filtered*, são dadas agora como *open*.



```
root@kali: ~  
root@kali: ~  
root@kali:~# nmap -sFV -T4 172.16.3.2  
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-03 10:20 EST  
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.  
Try using --system-dns or specify valid servers with --dns-servers  
Nmap scan report for 172.16.3.2  
Host is up (0.0020s latency).  
Not shown: 977 closed ports  
PORT      STATE SERVICE      VERSION  
21/tcp    open  ftp          vsftpd 2.3.4  
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)  
23/tcp    open  telnet       Linux telnetd  
25/tcp    open  smtp         Postfix smtpd  
53/tcp    open  domain       ISC BIND 9.4.2  
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)  
111/tcp   open  rpcbind      2 (RPC #100000)  
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)  
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)  
512/tcp   open  exec         netkit-rsh rexecd  
513/tcp   open  login        Netkit rshd  
514/tcp   open  shell        Netkit rshd  
1099/tcp  open  java-rmi     GNU Classpath grmiregistry  
1524/tcp  open  bindshell    Metasploitable root shell  
2049/tcp  open  nfs          2-4 (RPC #100003)  
2121/tcp  open  ftp          ProFTPD 1.3.1  
3306/tcp  open  mysql        MySQL 5.0.51a-3ubuntu5  
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7  
5900/tcp  open  vnc          VNC (protocol 3.3)  
6000/tcp  open  X11          (access denied)  
6667/tcp  open  irc          UnrealIRCd  
8009/tcp  open  ajp13        Apache Jserv (Protocol v1.3)  
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1  
MAC Address: 08:00:27:20:1B:E6 (Oracle VirtualBox virtual NIC)  
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: U  
nix, Linux; CPE: cpe:/o:linux:linux_kernel  
  
Service detection performed. Please report any incorrect results at https://nmap  
.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 14.22 seconds  
root@kali:~#
```

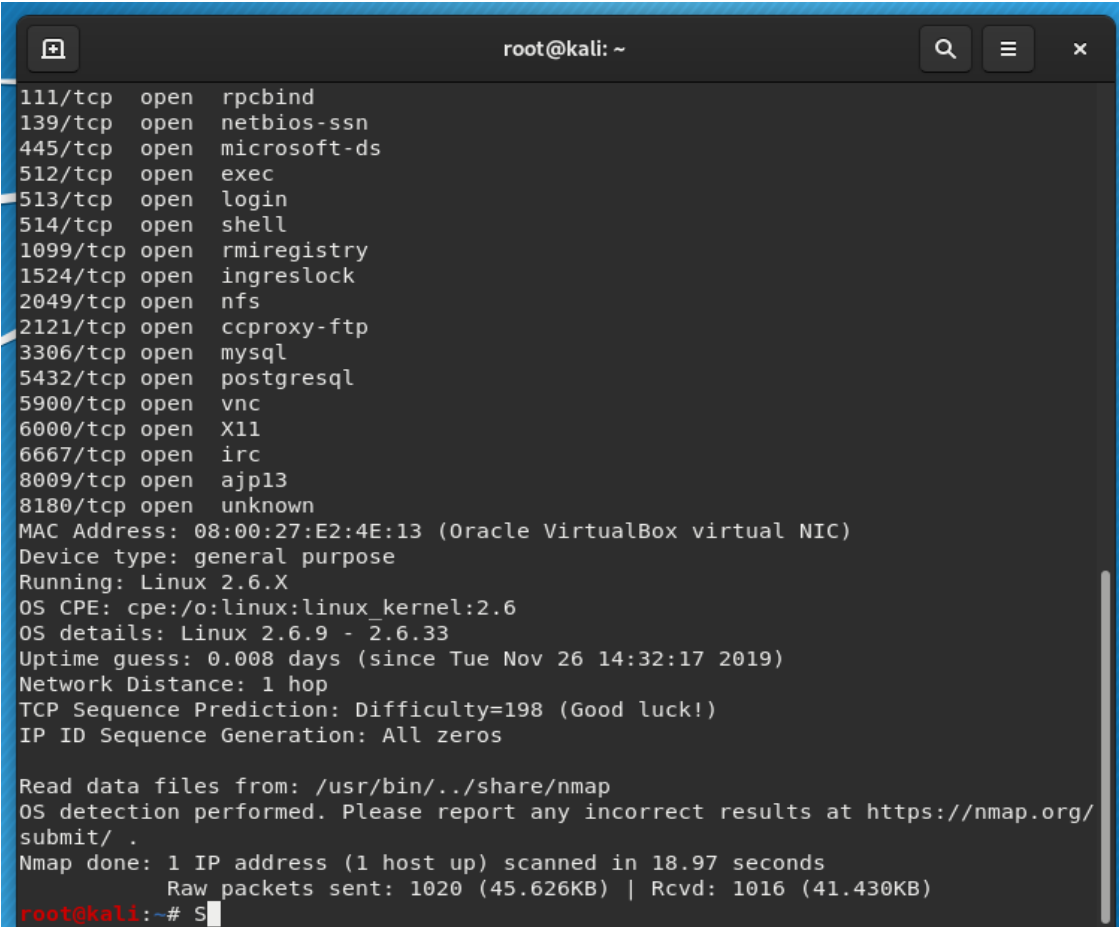
Figura 22. Output do modo de scan `-sF` com a flag `-V` e `-T` feito à Máquina Virtual Metasploitable 2

O modo de *scan-sF* não define qualquer tipo de *bit* no pacote TCP que é enviado. Já o modo de *scan-sX* define as *flags* FIN, PSH e URG.

Os resultados para ambos os comandos são exatamente iguais aos obtidos no primeiro comando. O resultado relativamente à Máquina Virtual Metasploitable 2 também é alterado quando é introduzida a *flag* da versão, passando as portas para o estado *open*.

2.3 Questão 3

Através do *scanning* ativo do NMap é possível descobrir uma panóplia de informação sobre o sistema operativo da VM Metasploitable 2. Nesta informação fornecida temos o **tipo do dispositivo (*Device Type*)** que foi “adivinhado” pelo NMap como um “*general purpose*” da vasta lista existente nas definições e algoritmos do NMap, dado que podia *router*, *printer*, *firewall* ou até uma mistura destes.



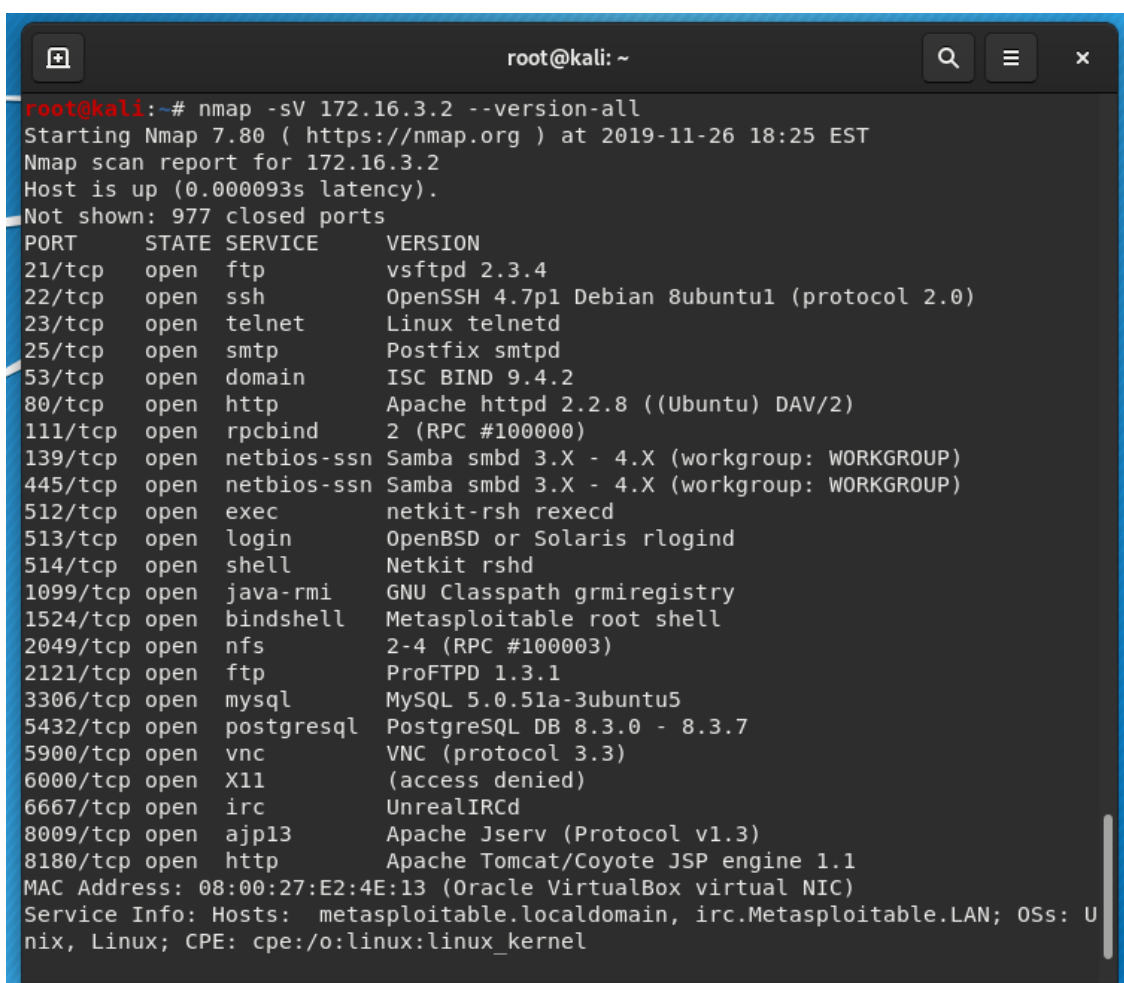
```
root@kali: ~  
111/tcp open  rpcbind  
139/tcp open  netbios-ssn  
445/tcp open  microsoft-ds  
512/tcp open  exec  
513/tcp open  login  
514/tcp open  shell  
1099/tcp open  rmiregistry  
1524/tcp open  ingreslock  
2049/tcp open  nfs  
2121/tcp open  ccproxy-ftp  
3306/tcp open  mysql  
5432/tcp open  postgresql  
5900/tcp open  vnc  
6000/tcp open  X11  
6667/tcp open  irc  
8009/tcp open  ajp13  
8180/tcp open  unknown  
MAC Address: 08:00:27:E2:4E:13 (Oracle VirtualBox virtual NIC)  
Device type: general purpose  
Running: Linux 2.6.X  
OS CPE: cpe:/o:linux:linux_kernel:2.6  
OS details: Linux 2.6.9 - 2.6.33  
Uptime guess: 0.008 days (since Tue Nov 26 14:32:17 2019)  
Network Distance: 1 hop  
TCP Sequence Prediction: Difficulty=198 (Good luck!)  
IP ID Sequence Generation: All zeros  
  
Read data files from: /usr/bin/./share/nmap  
OS detection performed. Please report any incorrect results at https://nmap.org/  
submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 18.97 seconds  
Raw packets sent: 1020 (45.626KB) | Rcvd: 1016 (41.430KB)  
root@kali:~# S
```

Figura 23. Scanning Ativo à Máquina Virtual Metasploitable 2

Também temos a parte do *Running* que nos informa a família e a geração do sistema operativo descoberto, sendo que neste caso sabemos que é um Linux 2.6.X. Existe também um campo chamado de *OS CPE* que nos indica se o que estamos a analisar é um sistema operativo, que empresa o criou, que produto e versão é e outras informações. Neste caso verificamos que é um sistema operativo pelo primeiro “o” e depois sabemos quem o criou e qual o produto, ou seja, a organização Linux e o produto sendo o *kernel*/Linux e a sua respetiva versão.

Outras informações indiretas sobre o sistema operativo como a adivinhação do *uptime* do sistema operativo que estamos a analisar ou mesmo a distância em rede entre o nosso *host* e o outro, contando assim os *routers* pelo meio. Neste caso não poderia ser 0, dado que isso só aconteceria em *localhost*, mas é 1, dado que o Metasploitable 2 está na mesma rede interna que a nossa Kali VM: 172.16.3.0/24.

2.4 Questão 4



```
root@kali:~# nmap -sV 172.16.3.2 --version-all
Starting Nmap 7.80 ( https://nmap.org ) at 2019-11-26 18:25 EST
Nmap scan report for 172.16.3.2
Host is up (0.000093s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login        OpenBSD or Solaris rlogind
514/tcp   open  shell        Netkit rshd
1099/tcp  open  java-rmi     GNU Classpath grmiregistry
1524/tcp  open  bindshell    Metasploitable root shell
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ftp          ProFTPD 1.3.1
3306/tcp  open  mysql        MySQL 5.0.51a-3ubuntu5
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
6667/tcp  open  irc          UnrealIRCd
8009/tcp  open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 08:00:27:E2:4E:13 (Oracle VirtualBox virtual NIC)
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

Figura 24. Output da deteção de versões dos serviços das portas da Máquina Virtual Metasploitable 2

No serviço PostgreSQL ativo no sistema temos uma vulnerabilidade recente e bem comprometedora: **CVE-2019-10164**. Através da análise deste CVE mais recente vemos que versões deste servidor de base de dados estiveram vulneráveis duma forma gigante caso ocorresse um *stack-based buffer overflow*. Assim qualquer utilizador autenticado podia trocar a sua *password* para um valor criado para acontecer o *overflow* fazendo com que toda a memória seja vista e um *shutdown* do servidor/sistema aconteça.

CVSS Score	9.0
Confidentiality Impact	Complete (There is total information disclosure, resulting in all system files being revealed.)
Integrity Impact	Complete (There is a total compromise of system integrity. There is a complete loss of system protection, resulting in the entire system being compromised.)
Availability Impact	Complete (There is a total shutdown of the affected resource. The attacker can render the resource completely unavailable.)
Access Complexity	Low (Specialized access conditions or extenuating circumstances do not exist. Very little knowledge or skill is required to exploit.)
Authentication	Single system (The vulnerability requires an attacker to be logged into the system (such as at a command line or via a desktop session or web interface).)
Gained Access	None
Vulnerability Type(s)	Execute Code Overflow
CWE ID	119

Figura 25. Detalhes da vulnerabilidade de CVE-2019-10164 correspondente ao serviço PostgreSQL

No serviço MySQL ativo no sistema temos uma vulnerabilidade recente e relativamente comprometedora: **CVE-2015-2575**. Ao analisar este CVE encontrou-se uma vulnerabilidade no conector de JAVA que afetava a confidencialidade e integridade através de vetores introduzidos.

CVSS Score	4.9
Confidentiality Impact	Partial (There is considerable informational disclosure.)
Integrity Impact	Partial (Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited.)
Availability Impact	None (There is no impact to the availability of the system.)
Access Complexity	Medium (The access conditions are somewhat specialized. Some preconditions must be satisfied to exploit)
Authentication	Single system (The vulnerability requires an attacker to be logged into the system (such as at a command line or via a desktop session or web interface).)
Gained Access	None
Vulnerability Type(s)	
CWE ID	CWE id is not defined for this vulnerability

Figura 26. Detalhes da vulnerabilidade de CVE-2015-2575 correspondente ao serviço MySQL

No serviço Apache HTTP Server ativo no sistema temos uma vulnerabilidade interessante: **CVE-2019-10098**. A mesma pode configurar os redireccionamentos para um URL inesperado criando desta forma um impacto na integridade e na verdade do sistema dado que é algo não pedido no URL fornecido pelo utilizador.

CVSS Score	5.8
Confidentiality Impact	Partial (There is considerable informational disclosure.)
Integrity Impact	Partial (Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited.)
Availability Impact	None (There is no impact to the availability of the system.)
Access Complexity	Medium (The access conditions are somewhat specialized. Some preconditions must be satisfied to exploit)
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	
CWE ID	601

Figura 27. Detalhes da vulnerabilidade de CVE-2019-10098 correspondente ao serviço Apache HTTP Server

3. Parte 2 – OpenVAS/ Nessus e Snort

As ferramentas OpenVAS/ Nessus e Snort vão-nos permitir observar as tais vulnerabilidades que caracterizam a Máquina Virtual Metasploitable 2, através de um resultado agrupado em gráficos, níveis de gravidade e até mesmo palavras-chave. Com isso, vamos conseguir estabelecer diferenças entre as ferramentas, chegando mesmo a resolver algumas vulnerabilidades existentes neste sistema *target*.

3.1. Questão 5

Começando o serviço Nessus através da linha de comandos e ativando também o IDS Snort, ao mesmo tempo que abrimos o *packet monitor* Wireshark, pudemos começar esta parte 2 da melhor forma e desta forma obter as informações detalhadas de todas as vulnerabilidades “apanhadas” pelo Nessus.

172.16.3.2

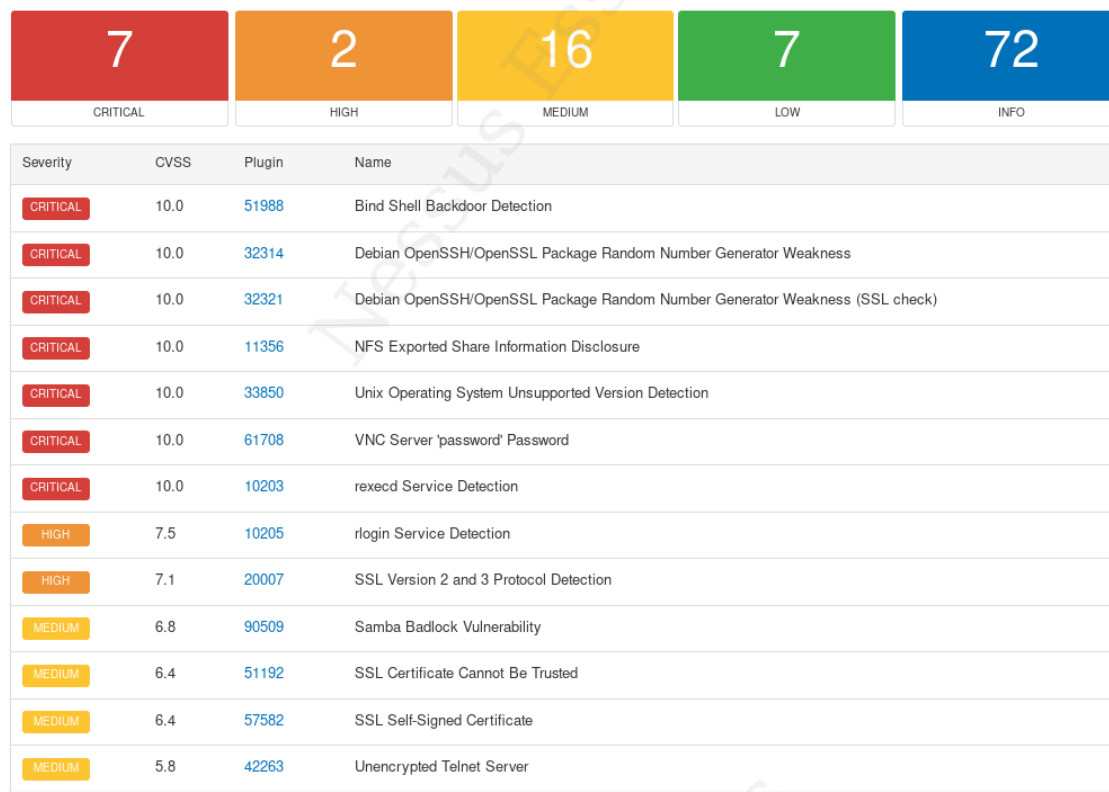


Figura 28. Contagem das vulnerabilidades da Máquina Virtual Metasploitable 2 e listagem de algumas delas

Após correr o *Basic Network Scan* conseguimos obter um leque de vulnerabilidades existentes na máquina virtual Metasploitable 2 e a classificação da mesma por ordem decrescente de CVSS (*Common Vulnerability Scoring System*) sendo que encontramos no *host* 172.16.3.2 um total de 104 vulnerabilidades sendo 7 críticas e com um CVSS de 10 como é possível ver na figura abaixo do relatório gerado pelo *Nessus*.

3.2 Questão 6

Examinando o *output* do *Snort* conseguimos obter algumas intrusões detetadas como tráfego anómalo. Como primeira vulnerabilidade temos a *Bind Shell Backdoor Detection*, que podemos verificar que foi tratada como anómala pelo *Snort* na Figura abaixo, demonstrando claramente que houve uma resposta ao ataque feito pelo *Nessus* e que devolveu tanto o *userid*, como o *root*.

```
[**] [1:498:6] ATTACK-RESPONSES id check returned root [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
12/04-17:57:34.403597 172.16.3.2:1524 -> 172.16.3.1:51566
TCP TTL:64 TOS:0x0 ID:9408 IpLen:20 DgmLen:91 DFS|
***AP*** Seq: 0x873D8D68 Ack: 0x99B84222 Win: 0xB5 TcpLen: 32
TCP Options (3) => NOP NOP TS: 43504 4247205762

[**] [1:1882:10] ATTACK-RESPONSES id check returned userid [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
12/04-17:57:34.403597 172.16.3.2:1524 -> 172.16.3.1:51566
TCP TTL:64 TOS:0x0 ID:9408 IpLen:20 DgmLen:91 DF
***AP*** Seq: 0x873D8D68 Ack: 0x99B84222 Win: 0xB5 TcpLen: 32
TCP Options (3) => NOP NOP TS: 43504 4247205762
```

CRITICAL Bind Shell Backdoor Detection

Description

A shell is listening on the remote port without any authentication being required. An attacker may use it by connecting to the remote port and sending commands directly.

Solution

Verify if the remote host has been compromised, and reinstall the system if necessary.

Output

```
Nessus was able to execute the command "id" using the
following request :
```

```
This produced the following truncated output (limited to 10 lines) :
----- snip -----
root@metasploitable:/# uid=0(root) gid=0(root) groups=0(root)
root@metasploitable:/#
----- snip -----
```

Figura 29. Excerto do output do Snort para a vulnerabilidade Bind Shell Backdoor Detection/Detalhes do Nessus sobre a mesma

Com a informação fornecida pelo *Snort* também conseguimos identificar o pacote TCP que veio da Máquina Virtual Metasploitable 2 e assim demonstrar o mesmo no Wireshark como mostra a Figura abaixo.

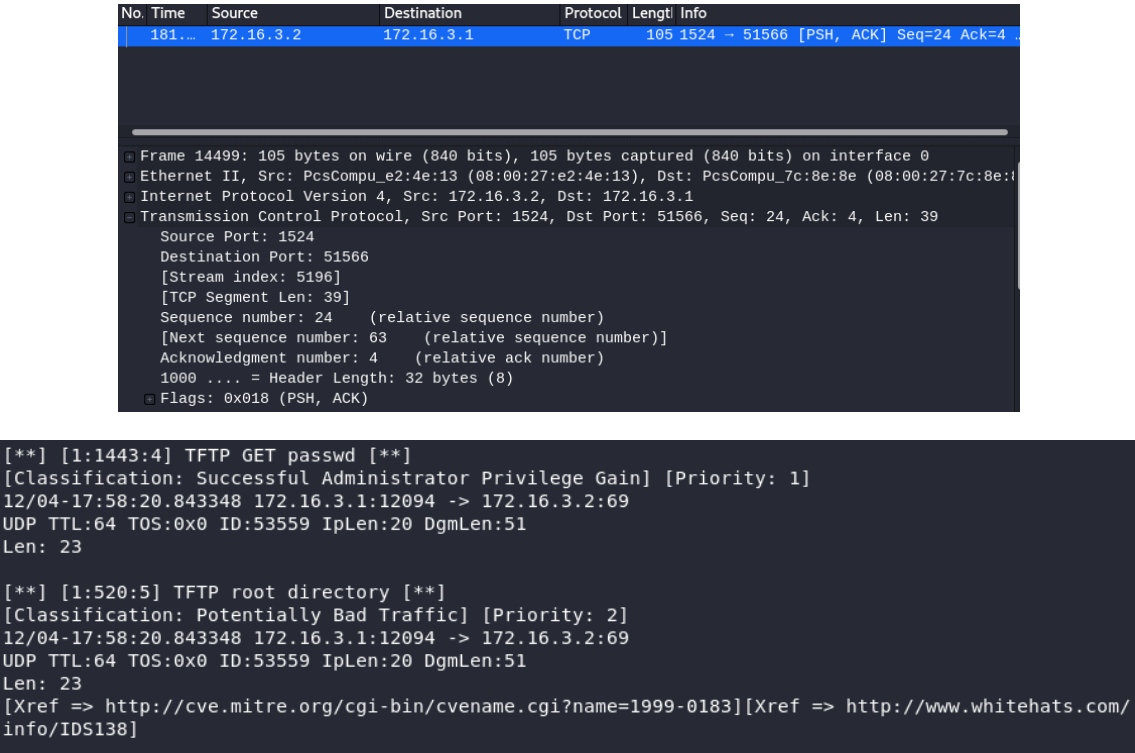
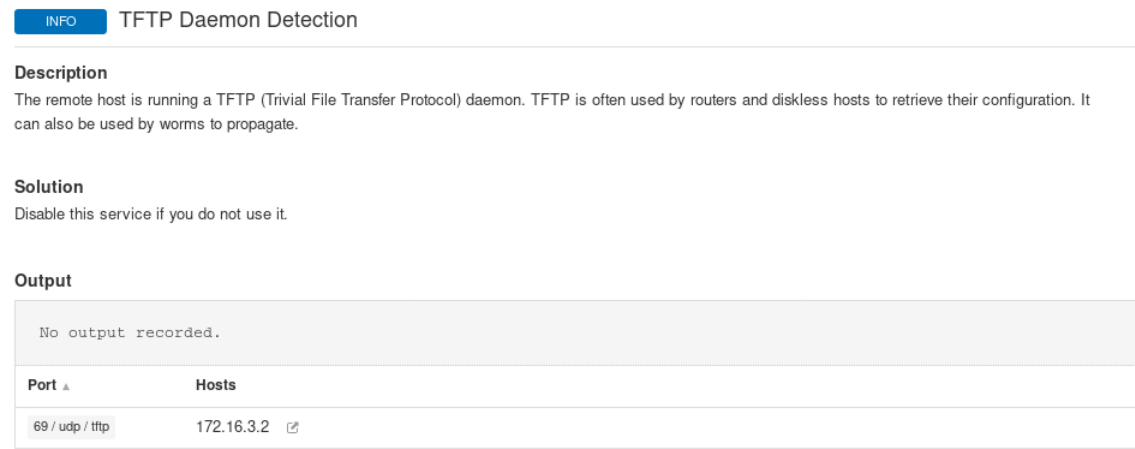


Figura 30. Excerto do output do pacote TCP vindo da Máquina Virtual Metasploitable 2/Tráfego Wireshark

Outro exemplo encontrado no *Snort* foi a tentativa de saber a *password* e o *root directory* do TFTP Server e a respetiva informação dita no *Nessus* na Figura abaixo.



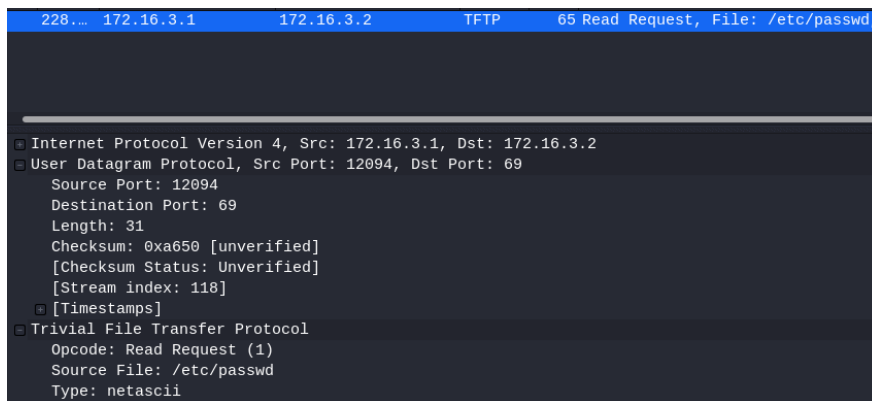


Figura 31. Detalhes do Nessus para a tentativa de descobrir a password e o root directory do TFTP Server/Tráfego Wireshark

3.3 Questão 7

A ferramenta *Snort* é uma mais valia poderosíssima e útil no que toca à deteção de ataques ao sistema. O princípio desta ferramenta consiste em observar os pacotes que circulam através do tráfego da rede, fazendo uma comparação com a base de regras que internamente possui e com isso gerar um alerta caso haja uma igualdade entre o pacote e as eventuais regras dessa base de conhecimento.

O objetivo desta questão passa essencialmente por entender certas diferenças no funcionamento desta ferramenta com a ferramenta de *scan* usada e de que forma isso leva a que depois existam notificações do IDS *Snort* que não possuem em si uma vulnerabilidade no sistema de *scan* usado até então.

Abrindo o ficheiro **alert.full** notamos logo a existência de pacotes que não têm qualquer ligação com o habitual padrão de uma vulnerabilidade. Na Figura 32 podemos pegar para análise logo o primeiro pacote, que denota o envio de um pacote em modo *broadcast* por parte do localhost, permitindo assim concluir que isso não representa de todo uma vulnerabilidade em termos de sistema de *scan*. Este tipo de alertas acontecem quando um *host* está a tentar conectar à rede e precisa de enviar pacotes UDP em *broadcast* como forma de requisição, correspondendo assim a um passo de reconhecimento da rede ou das máquinas que existem nela.

```

[**] [1:527:8] BAD-TRAFFIC same SRC/DST [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
12/04-17:52:14.975946 0.0.0.0:68 -> 255.255.255.255:67
UDP TTL:64 TOS:0x10 ID:0 IpLen:20 DgmLen:328
Len: 300
[Xref => http://www.cert.org/advisories/CA-1997-28.html][Xref => http://cve.mitre.org

```

Figura 32. Excerto do output do Snort correspondente aos pacotes UDP enviados em broadcast

Na Figura imediatamente a seguir, podemos analisar tráfego que consiste em alertas gerados no decorrer do *scan* e que também não possuem qualquer vulnerabilidade detetada por parte do sistema OpenVAS.

Isso fica perceptível se analisarmos os dois pacotes da Figura. O primeiro pacote corresponde à execução do comando *ping* que em si trata de enviar pacotes **ICMP Echo Request** para o *host* de destino, recebendo dele pacotes **ICMP Echo Reply**. Isto faz todo o sentido, dado que esta interação acontece entre a Máquina Virtual Kali e a Máquina Virtual Metasploitable 2.

```

[**] [1:384:5] ICMP PING [**]
[Classification: Misc activity] [Priority: 3]
12/04-17:55:29.245147 172.16.3.1 -> 172.16.3.2
ICMP TTL:64 TOS:0x0 ID:55075 IpLen:20 DgmLen:29
Type:8 Code:0 ID:1 Seq:1 ECHO

[**] [1:408:5] ICMP Echo Reply [**]
[Classification: Misc activity] [Priority: 3]
12/04-17:55:29.245417 172.16.3.2 -> 172.16.3.1
ICMP TTL:64 TOS:0x0 ID:29006 IpLen:20 DgmLen:29
Type:0 Code:0 ID:1 Seq:1 ECHO REPLY

```

Figura 33. Excerto do output do Snort correspondente a um comando *ping*

Isto são exemplos de pacote que existem por parte do *Snort*, mas que não se assumem como vulnerabilidade no sistema de *scan*. Este tipo de alertas são, no entanto muito importantes, tendo em conta que um atacante pode querer “estudar” o *target* antes mesmo de fazer o seu *exploit*. Assim, através deste tipo de alertas, pode-se prevenir um possível ataque, pelo tráfego alheio que possa estar a existir na rede.

3.4 Questão 8

Para se terminar este trabalho prático, espera-se solucionar algumas das vulnerabilidades encontradas na VM Metasploitable 2. A ideia é estudar um pouco de cada uma delas através de uma análise atenta nas Bases de Dados usadas/estudadas até então, tentando assim criar um esboço da forma como pode ser contornado o problema em si. No final disso, por aplicação das soluções criadas, será feito um novo *scan* à Máquina Virtual Metasploitable 2 na esperança de se ter resolvido as vulnerabilidades em causa.

Para a resolução desta última questão, foi usada a ferramenta OpenVAS, ao contrário do que acontece na Questão 5. É importante ter isto em mente, dado que esta mudança de ferramenta faz com que o nome da vulnerabilidade possa ser diferente, assim como a sua classificação.

3.4.1. HTTP TRACK/TRACE Methods Allowed

1. Descrição/Detalhes Vulnerabilidade

No OpenVAS, a descrição desta vulnerabilidade é dada por **HTTP Debugging Methods (TRACE/TRACK) Enabled** e é classificada com um grau de 5.8 em termos de gravidade, o que faz com que se insira numa vulnerabilidade de risco nível MEDIUM.

Através do resultado apresentado pelo OpenVAS podemos apurar as seguintes informações:

- **Descrição:** As funções de depuração estão ativadas no servidor WEB remoto. Esse servidor suporta os métodos TRACE e/ou TRACK.
- **Problema em si:** O servidor WEB o método HTTP TRACE ativo;
- **Impacto:** Um atacante pode usar essa falha para enganar os seus utilizadores WEB a fornecerem as suas credenciais;
- **Possível Solução:** Desativar os métodos TRACE e TRACK na configuração do servidor WEB.



Vulnerability	Severity	QoD	Host	Location	Actions
HTTP Debugging Methods (TRACE/TRACK) Enabled	5.8 (Medium)	99%	172.16.3.2	80/tcp	
Summary Debugging functions are enabled on the remote web server. The remote web server supports the TRACE and/or TRACK methods. TRACE and TRACK are HTTP methods which are used to debug web server connections.					
Vulnerability Detection Result The web server has the following HTTP methods enabled: TRACE					
Impact An attacker may use this flaw to trick your legitimate web users to give him their credentials.					
Solution Solution type:  Mitigation Disable the TRACE and TRACK methods in your web server configuration. Please see the manual of your web server or the references for more information.					
Affected Software/OS Web servers with enabled TRACE and/or TRACK methods.					

Figura 34. Resultado do OpenVAS da vulnerabilidade HTTP Debugging Methods Enabled

2. Possível Resolução Vulnerabilidade

Esta informação é suficiente para se tentar solucionar o problema em causa. Tendo em conta que se sabe que a vulnerabilidade acontece na porta 80, cujo serviço em execução é o TCP, há algo a mais que podemos descobrir. Na realidade, vimos anteriormente que, ativando a *flag* de deteção de versão ao comando do NMap, ficávamos a saber também a versão do serviço TCP para cada porta aberta.

```

root@kali: ~
root@kali: ~
root@kali:~# nmap -sV 172.16.3.2 -p 80
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-05 16:46 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 172.16.3.2
Host is up (0.00029s latency).

PORT      STATE SERVICE VERSION
80/tcp    open  http      Apache httpd 2.2.8 ((Ubuntu) DAV/2)
MAC Address: 08:00:27:20:1B:E6 (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.58 seconds
root@kali:~#

```

Figura 35. Verificação do serviço (e versão) que corre na porta 80 da Máquina Virtual Metasploitable 2

Assim se descobre que o serviço em causa é o **Apache 2.2.8**. Esta informação é essencial para que se consiga perceber aquilo que pode ser alterado para resolver o problema.

A resolução passa por alterar o ficheiro de configuração principal do **Apache2**. No caso da Máquina Virtual Metasploitable 2, esse ficheiro encontra-se na pasta dedicada ao próprio **Apache2**.

Para resolver a vulnerabilidade, seguem-se os seguintes passos:

- Localizar a pasta do Apache2: Basta digitar no terminal `cd /etc/apache2/`
- Abrir Ficheiro .conf com permissões sudo: Basta digitar no terminal `sudo vim httpd.conf`
- Editar o ficheiro: Adicionar a linha `TraceEnable off` ao ficheiro

É importante fazer *restart* a ambas a máquinas, principalmente à Máquina Virtual Metasploitable 2, visto que só assim conseguimos garantir a 100% que o Apache2 é reiniciado.

3. Diferenças nos Resultados

O desaparecimento da vulnerabilidade em causa é validado pela fatia do número de vulnerabilidades de gravidade MEDIUM.

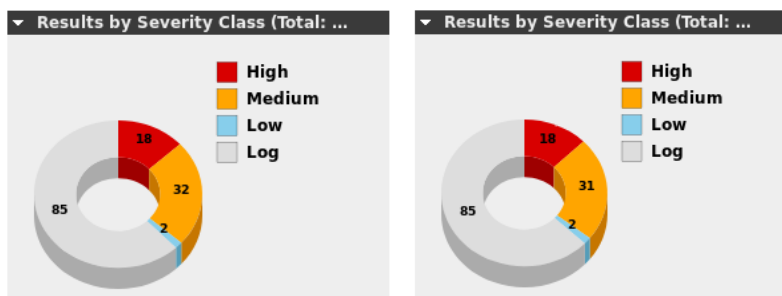


Figura 36. Gráficos do scan inicial e do scan após correção da vulnerabilidade correspondentes aos resultados das vulnerabilidades por nível de gravidade

Note-se que o OpenVAS permite agrupar as vulnerabilidades por palavra. Neste caso, viram-se todas as vulnerabilidades cuja palavra HTTP se encontrava escrita. Comprovamos então a resolução da vulnerabilidade.

Vulnerability	Severity	QoD	Host	Location	Created
HTTP Server type and version	0.0 (Log)	80%	172.16.3.2	80/tcp	Thu Dec 5 19:20:48 2019
Test HTTP dangerous methods	7.5 (High)	99%	172.16.3.2	80/tcp	Thu Dec 5 19:24:47 2019
Cleartext Transmission of Sensitive Information via HTTP	4.8 (Medium)	80%	172.16.3.2	80/tcp	Thu Dec 5 19:20:48 2019
HTTP Security Headers Detection	0.0 (Log)	80%	172.16.3.2	80/tcp	Thu Dec 5 19:13:53 2019
HTTP Debugging Methods (TRACE/TRACK) Enabled	5.8 (Medium)	99%	172.16.3.2	80/tcp	Thu Dec 5 19:21:35 2019
Apache HTTP Server 'httpOnly' Cookie Information Disclosure Vulnerability	4.3 (Medium)	99%	172.16.3.2	80/tcp	Thu Dec 5 19:24:24 2019

Vulnerability	Severity	QoD	Host	Location	Created
HTTP Server type and version	0.0 (Log)	80%	172.16.3.2	80/tcp	Thu Dec 5 22:29:57 2019
Test HTTP dangerous methods	7.5 (High)	99%	172.16.3.2	80/tcp	Thu Dec 5 22:33:31 2019
Cleartext Transmission of Sensitive Information via HTTP	4.8 (Medium)	80%	172.16.3.2	80/tcp	Thu Dec 5 22:29:57 2019
HTTP Security Headers Detection	0.0 (Log)	80%	172.16.3.2	80/tcp	Thu Dec 5 22:21:01 2019
Apache HTTP Server 'httpOnly' Cookie Information Disclosure Vulnerability	4.3 (Medium)	99%	172.16.3.2	80/tcp	Thu Dec 5 22:33:43 2019

Figura 37. Lista das vulnerabilidades do scan inicial e do scan após correção da vulnerabilidade

3.4.2. rexecd Service Detection

1. Descrição/Detalhes Vulnerabilidade

No OpenVAS, a descrição desta vulnerabilidade é dada por **rexec Passwordless / Unencrypted Cleartext Login** e é classificada com um grau de 10.0 em termos de gravidade, o que faz com que se insira numa vulnerabilidade de risco nível HIGH.

Sabe-se a correspondência em termos de Nessus e OpenVAS pelas referências anexadas no resultado do OpenVAS, onde conseguimos ver a informação do número de CVE.

Através do resultado apresentado pelo OpenVAS podemos apurar as seguintes informações:

- **Descrição:** O serviço **rexecd** está a ser executado no *host* remoto;
- **Problema em si:** O serviço **rexecd** não está a permitir quaisquer conexões por parte desse *host*;
- **Possível Solução:** Desativar o serviço **rexecd** e usar alternativas como o **SSH**.








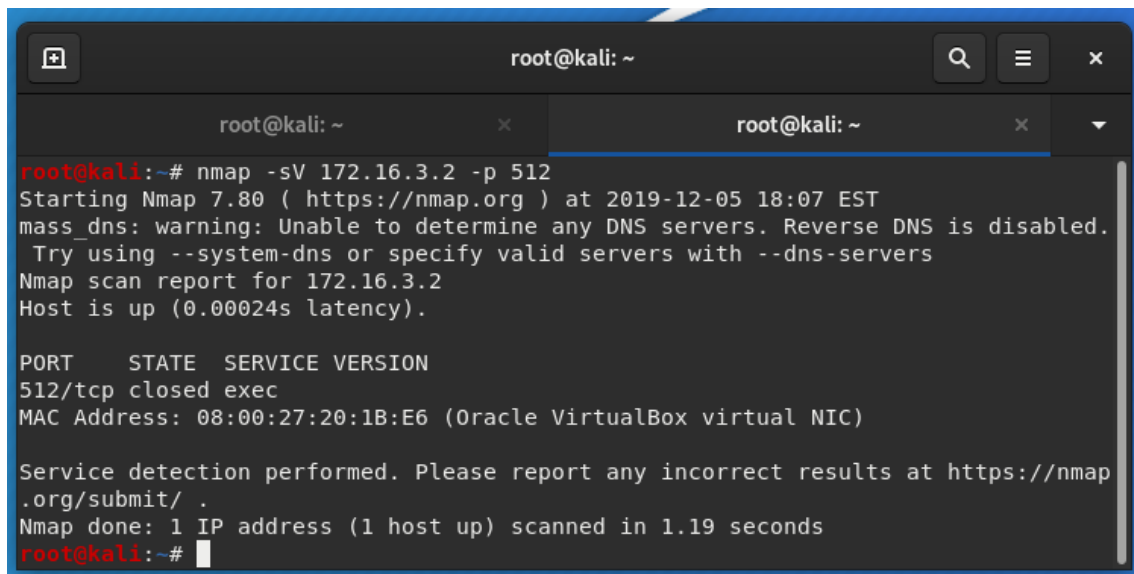
Vulnerability		Severity		QoD	Host	Location	Actions
rexec Passwordless / Unencrypted Cleartext Login		10.0 (High)		80%	172.16.3.2	512/tcp	 
Summary This remote host is running a rexec service.							
Vulnerability Detection Result The rexec service is not allowing connections from this host.							
Solution Solution type:  Mitigation Disable the rexec service and use alternatives like SSH instead.							
Vulnerability Insight rexec (Remote Process Execution) has the same kind of functionality that rsh has: you can execute shell commands on a remote computer. The main difference is that rexec authenticate by reading the username and password *unencrypted* from the socket.							
Vulnerability Detection Method Details: rexec Passwordless / Unencrypted Cleartext Login (OID: 1.3.6.1.4.1.25623.1.0.100111) Version used: \$Revision: 13541 \$							
References Other: https://web.nvd.nist.gov/view/vuln/detail?vulnid=CVE-1999-0618							

Figura 38. Resultado do OpenVAS da vulnerabilidade rexec Service Detection

2. Possível Resolução Vulnerabilidade

Aplicando-se a mesma ideia da vulnerabilidade anterior, verificamos novamente o serviço que está a correr na porta em questão, neste caso a 512. Com o serviço consegue-se mais facilmente pesquisar acerca do assunto e saber o que pode estar a provocar diretamente a falha.



```
root@kali: ~
root@kali:~# nmap -sV 172.16.3.2 -p 512
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-05 18:07 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 172.16.3.2
Host is up (0.00024s latency).

PORT      STATE SERVICE VERSION
512/tcp    closed exec
MAC Address: 08:00:27:20:1B:E6 (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.19 seconds
root@kali:~#
```

Figura 39. Verificação do serviço (e versão) que corre na porta 512 da Máquina Virtual Metasploitable 2

Pesquisa feita, surge uma forma simples de resolver o problema, pelo simples ato de comentar a linha de um determinado ficheiro que detalha o *daemon* **exec/rexecd**. No entanto, foi necessário entender de antemão o ficheiro onde se devia proceder a tal alteração – o ficheiro **inetd.conf**.

Este ficheiro é o ficheiro padrão para o *daemon* **inetd** e permite especificar os *daemons* a serem iniciados, definindo com isso o modo como o **inetd** lida com as solicitações de serviços Internet. Tal informação e solução acaba por fazer todo o sentido, já que o *daemon* **rexecd** não está a permitir conexões por parte do *host* remoto, sendo então preciso facultá-lo.

Para resolver a vulnerabilidade, seguem-se os seguintes passos:

- Localizar a pasta onde se encontra o ficheiro: Basta digitar no terminal **cd /etc**
- Abrir Ficheiro **.conf** com permissões **sudo**: Basta digitar no terminal **sudo vim inetd.conf**
- Editar o ficheiro: Comentar a linha **exec**

É também importante fazer o *restart* de ambas as Máquinas Virtuais antes de se fazer o novo *scan*.

3. Diferenças nos Resultados

Vemos uma diminuição de 2 vulnerabilidades de nível HIGH. Isso pode-nos levar a induzir que existiu a resolução de outros problemas/vulnerabilidades para além da que tentou efetivamente resolver.

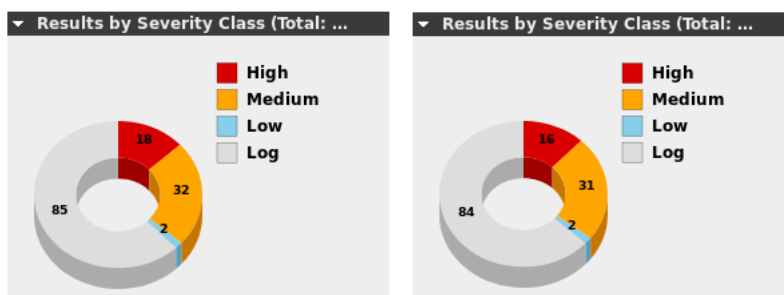


Figura 40. Gráficos do scan inicial e do scan após correção da vulnerabilidade correspondentes aos resultados das vulnerabilidades por nível de gravidade

Também pelas Figura abaixo, validamos a resolução da vulnerabilidade em questão e conseguimos ainda saber ao certo a vulnerabilidade que acabou também por ser resolvida - Java RMI Server.

Vulnerability	Severity	QoD	Host	Location	Created
rexec Passwordless / Unencrypted Cleartext Login	10.0 (High)	80%	172.16.3.2	512/tcp	Thu Dec 5 19:17:12 2019
Possible Backdoor: Ingreslock	10.0 (High)	99%	172.16.3.2	1524/tcp	Thu Dec 5 19:25:55 2019
DistCC Remote Code Execution Vulnerability	9.3 (High)	99%	172.16.3.2	3632/tcp	Thu Dec 5 19:23:19 2019
OS End Of Life Detection	10.0 (High)	80%	172.16.3.2	general/tcp	Thu Dec 5 19:21:46 2019
Distributed Ruby (dRuby/DRb) Multiple Remote Code Execution Vulnerabilities	10.0 (High)	99%	172.16.3.2	8787/tcp	Thu Dec 5 19:23:43 2019
Java RMI Server Insecure Default Configuration Remote Code Execution Vulnerability	10.0 (High)	95%	172.16.3.2	1099/tcp	Thu Dec 5 19:24:45 2019
TWiki XSS and Command Execution Vulnerabilities	10.0 (High)	80%	172.16.3.2	80/tcp	Thu Dec 5 19:22:18 2019

Vulnerability	Severity	QoD	Host	Location	Created
Possible Backdoor: Ingreslock	10.0 (High)	99%	172.16.3.2	1524/tcp	Thu Dec 5 23:29:20 2019
DistCC Remote Code Execution Vulnerability	9.3 (High)	99%	172.16.3.2	3632/tcp	Thu Dec 5 23:27:04 2019
OS End Of Life Detection	10.0 (High)	80%	172.16.3.2	general/tcp	Thu Dec 5 23:26:35 2019
Distributed Ruby (dRuby/DRb) Multiple Remote Code Execution Vulnerabilities	10.0 (High)	99%	172.16.3.2	8787/tcp	Thu Dec 5 23:28:45 2019
TWiki XSS and Command Execution Vulnerabilities	10.0 (High)	80%	172.16.3.2	80/tcp	Thu Dec 5 23:26:14 2019

Figura 41. Lista das vulnerabilidades do scan inicial e do scan após correção da vulnerabilidade

3.4.3. Bind Shell Backdoor Detection

Esta vulnerabilidade não se encontra listada nos resultados fornecidos pelo *scan* via OpenVAS. Ainda assim, como se usou o Nessus na Questão 5, foi possível verificar que de facto ela existe e de que forma se comporta. Na Questão 6 é abordada a base do seu funcionamento e é através da sua solução que realmente se compreende o porquê de ela poder não se manifestar no que toca ao *host* remoto.

Por isso, podemos deduzir que o *host* remoto não foi comprometido no processo de *scan*, não existindo assim a deteção da vulnerabilidade. A sua solução passa apenas por reinstalar o sistema caso o *host* remoto tenha sido comprometido.

3.4.4. VNC Server 'password' Password

1. Descrição/Detalhes Vulnerabilidade

No OpenVAS, a descrição desta vulnerabilidade é dada por **VNC Brute Force Login** é classificada com um grau de 9.0 em termos de gravidade, o que faz com que se insira numa vulnerabilidade de risco nível HIGH.

Através do resultado apresentado pelo OpenVAS podemos apurar as seguintes informações:

- **Descrição:** Tentar fazer *login* com as palavras-passe fornecidas pelo protocolo VNC;
- **Problema em si:** Possibilidade de conectar ao servidor VNC com a palavra-passe: *password*;
- **Possível Solução:** Alterar a palavra-passe para algo realmente forte.

Vulnerability	Severity	QoD	Host	Location	Actions
VNC Brute Force Login	9.0 (High)	95%	172.16.3.2	5900/tcp	
Summary Try to log in with given passwords via VNC protocol.					
Vulnerability Detection Result It was possible to connect to the VNC server with the password: password					
Solution Solution type: Mitigation Change the password to something hard to guess or enable password protection at all.					
Vulnerability Insight This script tries to authenticate to a VNC server with the passwords set in the password preference. It will also test and report if no authentication / password is required at all. Note: Some VNC servers have a blacklisting scheme that blocks IP addresses after five unsuccessful connection attempts for a period of time. The script will abort the brute force attack if it encounters that it gets blocked. Note as well that passwords can be max. 8 characters long.					

Figura 42. Resultado do OpenVAS da vulnerabilidade VNC Brute Force Login

2. Possível Resolução Vulnerabilidade

Visto que se sabia que o problema estava ligado diretamente com o servidor VNC e tendo em conta que a solução dada consistia em alterar a *password* do servidor, a única parte complicada foi perceber como e onde o fazer. Uma vez que não existia ainda a pasta padrão onde a *password* VNC é guardada, foi necessário criá-la e só depois definir a palavra-passe em si.

Para resolver a vulnerabilidade, seguem-se os seguintes passos:

- Mudar permissões para acesso *root*: Basta digitar no terminal `sudo su`
- Abrir a diretoria onde vai/deve ser criada a pasta: Basta digitar no terminal `cd ~`
- Criar a pasta padrão onde vai/deve ser definida a *password* do servidor VNC: Basta digitar no terminal `mkdir .vnc`
- Definir a *password*: Basta digitar no terminal `vncpasswd`

O passo de criação da pasta padrão não tem obrigatoriamente de ser feito, dado que ao executarmos o comando `vncpasswd` já estamos a criar a pasta caso não exista, tal como acontece na Figura 43.

Ao invocarmos este último comando no terminal, vai ser iniciado todo o processo para definir a palavra-passe. Como visto na Figura abaixo, a mesma é pedida duas vezes e após isso deve-se escolher se queremos que a *password* seja apenas para leitura. Caso essa opção seja requerida, é necessário introduzir mais duas vezes a *password*.

```
msfadmin@metasploitable:~$ sudo su
[sudo] password for msfadmin:
root@metasploitable:/home/msfadmin# cd ~
root@metasploitable:~# ls -la
.          Desktop      .gstreamer-0.10  reset_logs.sh    webapp
..         .filezilla      .mozilla         .rhosts          .Xauthority
.bash_history .fluxbox       .mysql_history   .ssh             .vnc
.bashrc     .gconf         .profile         .vnc             vnc.log
.config     .gconfd       .purple
root@metasploitable:~# vncpasswd
Using password file /root/.vnc/passwd
Password:
Verify:
Would you like to enter a view-only password (y/n)? y
Password:
Verify:
```

Figura 43. Comandos no terminal que demonstram a resolução da vulnerabilidade

3. Diferenças nos Resultados

Note-se que o primeiro gráfico corresponde sempre ao estado inicial do *scan* sem qualquer vulnerabilidade resolvida. Assim, as 31 vulnerabilidades de nível MEDIUM correspondem à resolução que foi feita para a vulnerabilidade HTTP TRACK/TRACE Methods Allowed e as 15 de nível HIGH validam já a diminuição desta última vulnerabilidade que ficou resolvida.

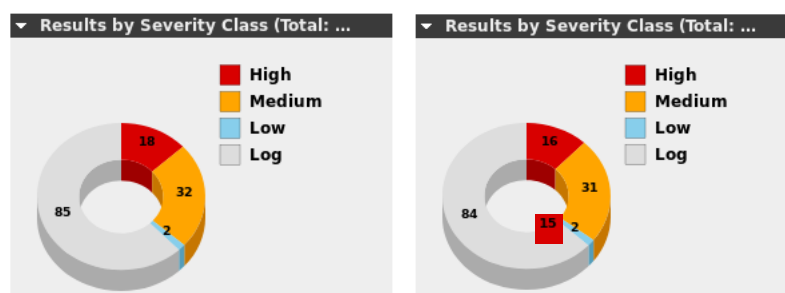





Figura 44. Gráficos do scan inicial e do scan após correção da vulnerabilidade correspondentes aos resultados das vulnerabilidades por nível de gravidade

Também pelas Figuras abaixo, validamos a resolução da vulnerabilidade em questão.

1 - 3 of 3						
Vulnerability		Severity	QoD	Host	Location	Created
MySQL / MariaDB weak password		9.0 (High)	95%	172.16.3.2	3306/tcp	Thu Dec 5 19:23:49 2019
PostgreSQL weak password		9.0 (High)	99%	172.16.3.2	5432/tcp	Thu Dec 5 19:24:06 2019
VNC Brute Force Login		9.0 (High)	95%	172.16.3.2	5900/tcp	Thu Dec 5 19:24:11 2019



1 - 2 of 2						
Vulnerability		Severity	QoD	Host	Location	Created
MySQL / MariaDB weak password		9.0 (High)	95%	172.16.3.2	3306/tcp	Fri Dec 6 02:30:16 2019
PostgreSQL weak password		9.0 (High)	99%	172.16.3.2	5432/tcp	Fri Dec 6 02:31:45 2019

Figura 45. Lista das vulnerabilidades do scan inicial e do scan após correção da vulnerabilidade

4. Conclusões e Observações Finais

Estando toda a parte prática da Unidade Curricular totalmente compreendida e trabalhada, é hora de redigir as observações finais. Tendo em conta a dimensão deste último trabalho prático, pode-se dizer que foi o mais complicado, não só pela complexidade em colocar o ambiente de testes funcional, mas também pelos problemas que foram surgindo nas tentativas de *scan* entre as duas Máquinas Virtuais.

Com o ambiente de testes totalmente preparado, o objetivo principal passou por entender os comandos que envolviam o NMap, aquilo que os distinguiu e as alterações de resultados realmente significantes para cada um deles. Dessa forma, ficou clara a possibilidade de contornar certos comandos e com isso desmitificar certas redundâncias ao nível do estado das várias portas existentes na Máquina Virtual Metasploitable 2. Ao fazer isso, vimos que algumas portas que estavam no estado *open/filtered* foram reduzidas para o estado *open*, acontecimento que em si é relevante quando se está a estudar/planear um possível ataque.

Passando-se à segunda parte do trabalho prático, ficou patente a importância de se usar paralelamente os comandos do NMap com a análise da lista de vulnerabilidades. Isto porque através deles descobrimos detalhes dos *serviços* a correr nas várias portas e com essa informação podemos interpretar melhor como/onde poderíamos insistir/pesquisar para corrigir certas vulnerabilidades.

Não menos importante, ficamos com uma nuance acerca da diferença ao nível das ferramentas *Snort* e *Nessus/OpenVAS*, dado que estudamos os resultados das mesmas e acabamos assim por perceber que existem pacotes no *output* do *Snort* que depois não correspondem a qualquer vulnerabilidade no sistema de *scan Nessus/OpenVAS*.

Estas foram as partes mais importantes, uma vez que geraram um pensamento e pesquisa extra. Ainda assim, e apesar da extensividade em termos de conteúdo, o grupo atingiu todos os objetivos propostos para este último trabalho prático, não existindo assim grandes dúvidas ao nível da matéria em si.

5.Referências

Guia de Referência do Nmap (Página do Manual). (s.d.). Obtido de NMAP.ORG:
https://nmap.org/man/pt_BR/

NMAP. (25 de Fevereiro de 2017). Obtido de TI de Boteco:
<https://tideboteco.wordpress.com/2017/02/25/nmap/>

Zillmer, E. (24 de Julho de 2017). *Viva o Linux*. Obtido de Nmap - 30 Exemplos para Análises de Redes e Portas: <https://www.vivaolinux.com.br/artigo/Nmap-30-Exemplos-para-Analises-de-Redes-e-Portas>