

Segurança em Redes

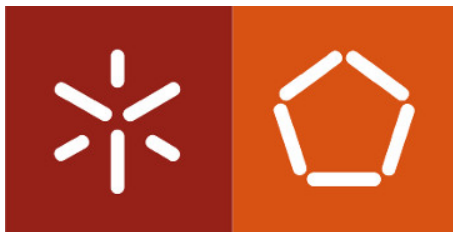
Trabalho Prático 5

18 de dezembro de 2020

Grupo 3

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

IPTables



Mestrado Integrado em Engenharia Informática
Universidade do Minho

Conteúdo

1	Introdução	3
2	Tarefa 1	4
2.1	Passo 1 - Set-up inicial das máquinas	4
2.2	Passo 2 - Ativar a firewall	4
2.3	Passo 3 - Verificar a firewall default	4
2.3.1	Tabela de entrada	5
2.3.2	Tabela de encaminhamento	5
2.3.3	Tabela de saída	5
2.3.4	Análise final	5
2.4	Passo 4 - Fazer um backup das tabelas	5
2.5	Passo 5 - Desativar a firewall	6
2.6	Passo 6 - Ativar a firewall	6
3	Tarefa 2	7
3.1	Passo 1 - Verificar a conectividade	7
3.2	Passo 2 - nmap	7
3.3	Passo 3 - pedido HTTP	8
3.4	Passo 4 - FTP	8
3.5	Passo 5 - SSH	9
4	Tarefa 3	9
4.1	Passo 1 - Modificar as regras da firewall	9
4.2	Passo 2 - Verificar as alterações nas tabela	9
4.3	Passo 3 - ping	11
4.4	Passo 4 - HTTP	11
4.5	Passo 5 - FTP	12
4.6	Passo 5 - SSH	12
4.7	Passo 6 - nmap	13
4.8	Passo 7 - Verificar as tabelas depois de algum tráfego	15
5	Tarefa 4	16
6	Conclusão	19
A	iptables.dump	20

Lista de Figuras

1	Estado "default" da "firewall"	4
2	Tabelas se a <i>firewall</i> for desativada	6
3	Resultado do comando ping	7
4	Resultado do comando "nmap -sS 10.0.0.7"	7
5	Resultado do comando "nmap -sn 10.0.0.7"	8
6	Resultado do pedido HTTP para a máquina servidor com a <i>firewall</i> por defeito	8
7	Resultado do pedido FTP para a máquina servidor com a <i>firewall</i> por defeito	9
8	Resultado do pedido SSH para a máquina servidor com a <i>firewall</i> por defeito	9
9	Tabela de <i>filter</i> depois de permitido o tráfego SSH, FTP E HTTP e proibido pedidos ICMP do tipo "ping"	10
10	Tentativa de ping	11
11	Tentativa de pedido HTTP, depois de atualizar as regras	12
12	Tentativa de conexão ao servidor FTP, depois de atualizar as regras	12
13	Tentativa de conexão SSH, depois de atualizar as regras	13
14	Output de "nmap -sS" depois de atualizar as regras da firewall	13
15	Output de "nmap -sV" depois de atualizar as regras da firewall	14
16	Output de "nmap -sn" depois de atualizar as regras da firewall	14
17	Output de "nmap -PE" depois de atualizar as regras da firewall	15
18	Tabelas depois de algum tráfego	15
19	Aplicação GUI para gestão da firewall	16
20	Regras customizadas para introdução na firewall	17
21	Aplicação GUI para gestão da firewall	17
22	Aplicação GUI para gestão da firewall	17

1 Introdução

A segurança nas redes de computadores é uma área que desde há muitos anos até aos dias de hoje tem sido bastante desenvolvida, pelo que a cada dia que passa, o papel relevante que este ramo da informática tem assumido têm sido um impulso para toda esta evolução. Desta forma, o tema deste trabalho prático está relacionado com uma das várias políticas de segurança associadas às redes, as *firewalls*.

Geralmente, as *firewalls* inserem-se no âmbito da proteção de redes internas e servem para filtrar tráfego de rede proveniente de redes externas não confiáveis. Deste modo, aquilo que vai ser feito é usar a estrutura de processamento e filtragem de pacotes dos sistemas Linux para definir um conjunto de regras com base numa aplicação Linux chamada *iptables*. Com esta aplicação é possível listar/adicionar/eliminar/alterar as regras de filtragem de pacotes presentes na configuração da *firewall* (estas regras ficam guardadas no *Kernel Linux*, pelo que se quisermos guardar e restaurar futuramente as regras, também existem primitivas que podemos usar para o fazer) e até saber o número de pacotes que cada regra já verificou ao longo do tempo. A estrutura do *iptables* é uma organização em tabelas, cada uma contendo *chains* (cadeias) e cada uma das cadeias com regras. Ora a tabela que por *default* é usada é a *filter*, pelo que é precisamente esta que vamos usar. Dentro desta tabela existem 3 *chains* - *INPUT*, *OUTPUT* e *FORWARD* - que servem, respetivamente, para filtrar pacotes que entram no servidor local, que saem do servidor local e que são para serem encaminhados para outro adaptador de rede local.

Assim, aquilo que se segue neste presente relatório é um cenário de teste à aplicação *iptables*, através do seguimento do enunciado deste trabalho prático, de forma a por em prática os conhecimentos aprendidos nas aulas teóricas e também experimentar novas ferramentas que fazem precisamente uso da estrutura mencionada anteriormente.

2 Tarefa 1

2.1 Passo 1 - Set-up inicial das máquinas

Foram usadas duas máquina virtuais para os efeitos deste trabalho. Numa delas foi instalada o Sistema Operativo "Kali", máquina que será referida como "cliente". Na outra, foi instalado o "CentOS 7.9", máquina que será referida como máquina "servidor". Houve uma primeira tentativa de instalar a versão 6.9 do "CentOS". No entanto algo estava a impossibilitar a conexão à internet necessária para instalar as ferramentas necessárias.

Usando o gestor de redes da Virtual Box, foi criada uma rede privada para as máquinas poderem comunicar entre si. Foram-lhes atribuídos endereços no bloco 10.0.0.0/24. Neste caso específico, a máquina servidor ficou com o endereço 10.0.0.7 enquanto que a máquina cliente ficou com o endereço 10.0.0.4.

Foram então instalados, na máquina que corre "CentOS", os 3 *softwares* que implementam os serviços necessários: "sshd", "httpd" e "vsftpd". Sendo estes um servidor de SSH, HTTP e FTP, respetivamente.

2.2 Passo 2 - Ativar a firewall

Usando o comando "system-config-firewall-tui" somos apresentado com um ecrã que nos permite ativar a "firewall" que pode ser controlada posteriormente com o comando "iptables".

2.3 Passo 3 - Verificar a firewall default

O comando "iptables -L -v -n" devolve como resposta as regras que estão impostas atualmente na "firewall". O modificador -v apresenta um contador de pacotes, enquanto que o modificar -n apresenta as portas.

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source            destination
 0      0 ACCEPT      all  --  *      *       0.0.0.0/0         0.0.0.0/0         state RELATED,ESTABLISHED
 0      0 ACCEPT      icmp --  *      *       0.0.0.0/0         0.0.0.0/0
 0      0 ACCEPT      all  --  lo     *       0.0.0.0/0         0.0.0.0/0
 0      0 REJECT      all  --  *      *       0.0.0.0/0         0.0.0.0/0         reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source            destination
 0      0 REJECT      all  --  *      *       0.0.0.0/0         0.0.0.0/0         reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source            destination
```

Figura 1: Estado "default" da "firewall"

2.3.1 Tabela de entrada

A primeira linha da tabela diz que todo e qualquer tráfego deve ser aceite, com uma condição: o "state" do tráfego tem que ser "RELATED" ou "ESTABLISHED". Isto significa que o tráfego tem que estar relacionado com uma *stream* anterior (por exemplo, a abertura de uma nova *stream* para *download* de uma imagem numa interação HTTP) ou tráfego de uma mesma *stream* que já estava aberta, por exemplo, a resposta a um pedido que foi efetuado pelo host. Logo, novas conexões que sejam originadas no exterior não vão ser aceites por esta regra.

A segunda linha define que todo o tráfego ICMP deve ser aceite, enquanto a sexta aceita todo o tráfego proveniente da interface "Loopback", ou seja, tráfego dentro do "localhost".

Por fim, a última linha rejeita todo e qualquer tráfego que não seja aceite por uma das regras anteriores.

Isto significa que na prática, apesar de a política por *default* seja aceitar o tráfego, todo o tráfego externo que não seja ICMP é recusado. Para além disso, é especificado que o tráfego é recusado com a mensagem "icmp-host-prohibited".

2.3.2 Tabela de encaminhamento

A única linha desta tabela proíbe todo e qualquer tráfego de ser reencaaminhado, o que faz sentido visto que, em princípio, uma máquina deste tipo não deveria fazer o papel de router.

2.3.3 Tabela de saída

Esta tabela está vazia, o que significa que todos os pacotes serão tratados de acordo com a política *default*, que neste caso é "ACCEPT". Assim sendo, todo o tráfego que pretende sair da máquina não será impedido pela *firewall*.

2.3.4 Análise final

O nível de segurança da *firewall* por defeito é, na nossa opinião, muito alto. Qualquer tráfego de entrada que não seja ICMP será recusado, a não ser que seja uma resposta a algo pedido pelo nosso *host*.

2.4 Passo 4 - Fazer um backup das tabelas

Com o comando "iptables-save > iptables.dump", foi criado um ficheiro que, no caso de algum erro irreversível, nos permite restaurar a *firewall*. Este

ficheiro está presente no Anexo A.

2.5 Passo 5 - Desativar a firewall

Depois de desativar a *firewall*, estas são as regras que existem nas tabelas.

```
Chain INPUT (policy ACCEPT 1 packets, 76 bytes)
pkts bytes target      prot opt in      out     source      destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination

Chain OUTPUT (policy ACCEPT 1 packets, 76 bytes)
pkts bytes target      prot opt in      out     source      destination
```

Figura 2: Tabelas se a *firewall* for desativada

Como podemos verificar, as tabelas estão completamente vazias. Assim sendo, todas elas vão reverter para a sua política *default*, de aceitar todo o tráfego. Assim sendo, na prática, isto é o mesmo que não ter nenhuma *firewall*.

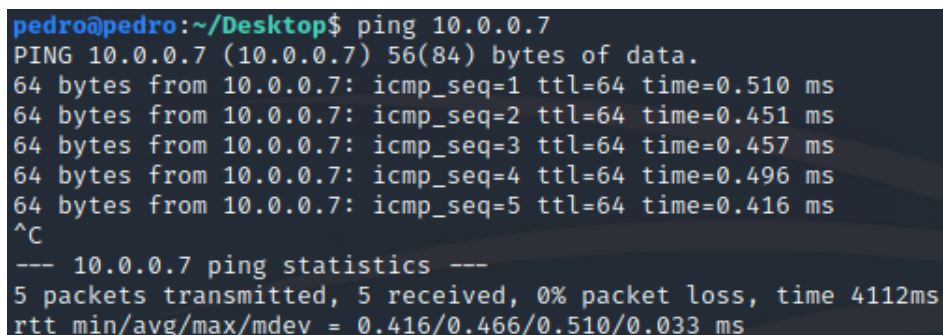
2.6 Passo 6 - Ativar a firewall

A *firewall* foi ativada novamente, e as regras voltaram a ser as mesmas que analisamos no passo 3.

3 Tarefa 2

3.1 Passo 1 - Verificar a conectividade

Com um "ping", a partir da máquina cliente, foi verificada a conectividade para a máquina servidor.

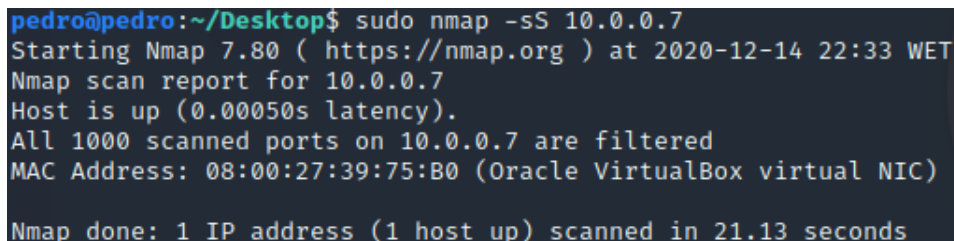


```
pedro@pedro:~/Desktop$ ping 10.0.0.7
PING 10.0.0.7 (10.0.0.7) 56(84) bytes of data.
64 bytes from 10.0.0.7: icmp_seq=1 ttl=64 time=0.510 ms
64 bytes from 10.0.0.7: icmp_seq=2 ttl=64 time=0.451 ms
64 bytes from 10.0.0.7: icmp_seq=3 ttl=64 time=0.457 ms
64 bytes from 10.0.0.7: icmp_seq=4 ttl=64 time=0.496 ms
64 bytes from 10.0.0.7: icmp_seq=5 ttl=64 time=0.416 ms
^C
--- 10.0.0.7 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4112ms
rtt min/avg/max/mdev = 0.416/0.466/0.510/0.033 ms
```

Figura 3: Resultado do comando ping

3.2 Passo 2 - nmap

Na máquina cliente, foi corrido o comando "nmap -sS 10.0.0.7".



```
pedro@pedro:~/Desktop$ sudo nmap -sS 10.0.0.7
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-14 22:33 WET
Nmap scan report for 10.0.0.7
Host is up (0.00050s latency).
All 1000 scanned ports on 10.0.0.7 are filtered
MAC Address: 08:00:27:39:75:B0 (Oracle VirtualBox virtual NIC)
Nmap done: 1 IP address (1 host up) scanned in 21.13 seconds
```

Figura 4: Resultado do comando "nmap -sS 10.0.0.7"

Como podemos ver, o nmap reporta que o "host" está ativo e que todas as 1000 portas que analisou estão filtradas, resultado da *firewall* ativa, que responde com "icmp-host-prohibited".

O nmap consegue também descobrir que estamos a correr numa máquina virtual, já que ele reconhece o endereço MAC como o pertencente ao NIC (Network Interface Controller) que a VirtualBox simula.

Podemos experimentar correr o comando "nmap -sn 10.0.0.7". Este comando corre um *scan* de "ping", ao invés do anterior que corria um *scan* usando pacotes "SYN".

```
pedro@pedro:~/Desktop$ sudo nmap -sn 10.0.0.7
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-14 22:40 WET
Nmap scan report for 10.0.0.7
Host is up (0.00036s latency).
MAC Address: 08:00:27:39:75:B0 (Oracle VirtualBox virtual NIC)
Nmap done: 1 IP address (1 host up) scanned in 13.16 seconds
```

Figura 5: Resultado do comando "nmap -sn 10.0.0.7"

Apesar de ter menos informação, visto que este *scan* não permite a diferenciação de portas, o que podemos verificar é que desta vez o que "nmap" não reconhece qualquer filtragem, o que faz sentido visto que como verificamos na Tarefa 1, a *firewall* não bloqueia tráfego ICMP.

Foram experimentados outros tipo de *scan*, usando pacotes "ACK", "FIN", "WINDOWS", entre outros, e até fragmentar os pacotes. No entanto, todos deram exatamente o mesmo resultado.

3.3 Passo 3 - pedido HTTP

Foi realizado um pedido HTTP a partir da máquina cliente para a máquina servidor, usando o software "W3M".

```
pedro@pedro:~/Desktop$ w3m http://10.0.0.7
w3m: Can't load http://10.0.0.7.
```

Figura 6: Resultado do pedido HTTP para a máquina servidor com a *firewall* por defeito

No entanto, como era de esperar, a página não pode ser carregada pois a *firewall* bloqueia o pedido.

3.4 Passo 4 - FTP

De forma semelhante, tentar aceder ao servidor FTP não dá resultado, pois mais uma vez a *firewall* vai bloquear o pedido.

```
pedro@pedro:~/Desktop$ ftp 10.0.0.7
ftp: connect: No route to host
ftp> █
```

Figura 7: Resultado do pedido FTP para a máquina servidor com a *firewall* por defeito

3.5 Passo 5 - SSH

Tentar aceder à máquina servidor por SSH dá, como de esperado, exatamente o mesmo resultado.

```
pedro@pedro:~/Desktop$ ssh pedro@10.0.0.7
ssh: connect to host 10.0.0.7 port 22: No route to host
```

Figura 8: Resultado do pedido SSH para a máquina servidor com a *firewall* por defeito

4 Tarefa 3

4.1 Passo 1 - Modificar as regras da firewall

Usando o comando "system-config-firewall-tui", é nos apresentado um TUI (Terminal User Interface) que nos permite acrescentar novas regras à nossa *firewall*. Este programa permite acrescentar regras sem ter que usar os comandos "iptables" diretamente, evitando erros e simplificando o processo.

Assim sendo, usamos o programa para permitir a passagem de tráfego SSH, HTTP e FTP. O programa automaticamente gera as regras que vão permitir o tráfego nas portas adequadas a esses serviços.

Para além disso, pedimos também para gerar uma regra que bloqueie os pedidos ICMP do tipo "ping".

4.2 Passo 2 - Verificar as alterações nas tabela

Depois de aplicar estas regras, esta é a tabela resultante.

```

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out    source            destination
 4   304 ACCEPT      all  --  any    any    anywhere          anywhere          state RELATED,ESTABLISHED
 0     0 REJECT      icmp --  any    any    anywhere          anywhere          icmp echo-request reject-with icmp-host-prohibited
 0     0 ACCEPT      icmp --  any    any    anywhere          anywhere
 0     0 ACCEPT      all  --  lo     any    anywhere          anywhere
 0     0 ACCEPT      tcp  --  any    any    anywhere          anywhere          state NEW tcp dpt:ssh
 0     0 ACCEPT      tcp  --  any    any    anywhere          anywhere          state NEW tcp dpt:http
 0     0 ACCEPT      tcp  --  any    any    anywhere          anywhere          state NEW tcp dpt:ftp
 2  1152 REJECT      all  --  any    any    anywhere          anywhere          reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out    source            destination
 0     0 REJECT      all  --  any    any    anywhere          anywhere          reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT 4 packets, 304 bytes)
pkts bytes target      prot opt in     out    source            destination

```

Figura 9: Tabela de *filter* depois de permitido o tráfego SSH, FTP E HTTP e proibido pedidos ICMP do tipo "ping"

Como podemos verificar, apenas a tabela de "INPUT" foi alterada, na qual estão presentes 4 novas linhas.

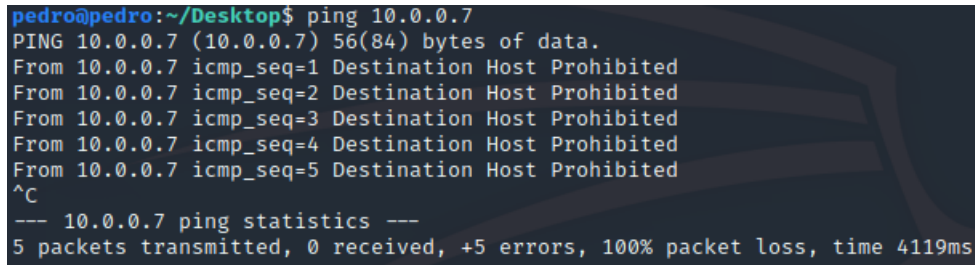
A primeira nova linha foi introduzida quase no início da tabela, na segunda posição, a qual proíbe tráfego ICMP do tipo "ping", que deve responder com "icmp-host-prohibited". De facto, para ser eficaz, esta linha tem que se posicionar antes da linha imediatamente a seguir, que permite todo o tipo de tráfego ICMP. Como a verificação de regras é parada no primeiro *match*, se a segunda e terceira linhas trocassem de posição a mesma seria inútil, pois qualquer tráfego ICMP seria aceite e a regra nunca seria verificada.

Da mesma maneira, as 3 linhas que permitem novas conexões SSH, HTTP e FTP têm que vir antes da última linha da tabela, que rejeita todo o tráfego que não seja "RELATED" ou "ESTABLISHED". Se estas 3 linhas fossem colocadas no fim da tabela, a regra que rejeita o tráfego tomaria precedência sobre todas elas.

A primeira linha da tabela, em conjunto com as 3 novas linhas que permitem conexões SSH, HTTP e FTP, têm uma relação interessante. A 3 novas linhas só serão verificadas no primeiro pacote de uma ligação de qualquer um dos tipos, visto que até as regras especificam que o *state* tem que ser "NEW". Depois de a ligação ser aceite, todos os pacotes posteriores serão aceites logo na primeira regra, pois terão um dos *state* "RELATED" ou "ESTABLISHED". Isto é ótimo de um ponto de vista de eficiência, visto que todos os pacotes posteriores ao primeiro serão aceites logo na primeira regra, evitando a verificação de todas as regras até chegar à regra específica do seu protocolo.

4.3 Passo 3 - ping

Se, novamente na máquina cliente, tentarmos fazer um "ping" para a máquina servidor, temos um resultado interessante, devido à natureza do tráfego ICMP.

A terminal window with a dark background and light-colored text. The prompt is 'pedro@pedro:~/Desktop\$'. The command entered is 'ping 10.0.0.7'. The output shows five ICMP echo requests, each failing with the message 'Destination Host Prohibited'. The sequence numbers for the requests are 1 through 5. After the fifth request, the user presses the Ctrl+C key, indicated by '^C'. The terminal then displays the ping statistics: '--- 10.0.0.7 ping statistics ---', '5 packets transmitted, 0 received, +5 errors, 100% packet loss, time 4119ms'.

```
pedro@pedro:~/Desktop$ ping 10.0.0.7
PING 10.0.0.7 (10.0.0.7) 56(84) bytes of data.
From 10.0.0.7 icmp_seq=1 Destination Host Prohibited
From 10.0.0.7 icmp_seq=2 Destination Host Prohibited
From 10.0.0.7 icmp_seq=3 Destination Host Prohibited
From 10.0.0.7 icmp_seq=4 Destination Host Prohibited
From 10.0.0.7 icmp_seq=5 Destination Host Prohibited
^C
--- 10.0.0.7 ping statistics ---
5 packets transmitted, 0 received, +5 errors, 100% packet loss, time 4119ms
```

Figura 10: Tentativa de ping

Obtemos como resposta "Destination Host Prohibited", o que nos permite saber que a máquina está online, visto que se tentarmos fazer *ping* para uma máquina que não existe na rede, a resposta que obtemos é "Destination Host Unreachable". Podemos pensar então que podemos responder com um "Destination Host Unreachable" para simular que a máquina não está *online*, mas isso não resolve o nosso problema. A diferença é o endereço que vem associado a essa mesma mensagem, que no nosso caso será o endereço da nossa máquina, enquanto que no caso em que a nossa máquina está de facto *offline*, terá o endereço do *router* que está um salto antes da nossa máquina. Assim sendo, tentar responder com "Destination Host Unreachable" será equivalente a gritar "Não está ninguém em casa!" quando alguém nos bate à porta.

4.4 Passo 4 - HTTP

Como segundo teste, usamos novamente o comando "w3m http://10.0.0.7", do qual obtemos o seguinte resultado.

```
Testing 123..

This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page it means that this site is working properly. This server is powered by CentOS.

Just visiting?

The website you just visited is either experiencing problems or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting www.example.com, you should send e-mail to "webmaster@example.com".

Are you the Administrator?

You should add your website content to the directory /var/www/html/.

To prevent this page from ever being used, follow the instructions in the file /etc/httpd/conf.d/welcome.conf.

Promoting Apache and CentOS

You are free to use the images below on Apache and CentOS Linux powered HTTP servers. Thanks for using Apache and CentOS!

[ Powered by Apache ] [ Powered by CentOS Linux ]
```

Figura 11: Tentativa de pedido HTTP, depois de atualizar as regras

Desta vez, recebemos de volta a página que vem por defeito com o servidor HTTP apache, para efeitos de teste. Como esperado, a firewall deixa passar pedidos HTTP.

4.5 Passo 5 - FTP

Novamente, depois de atualizar as regras da *firewall*, conseguimos conectar ao servidor FTP que está a correr na máquina servidor, e fazer login.

```
pedro@pedro:~/Desktop$ ftp 10.0.0.7
Connected to 10.0.0.7.
220 (vsFTPd 3.0.2)
Name (10.0.0.7:pedro): pedro
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> █
```

Figura 12: Tentativa de conexão ao servidor FTP, depois de atualizar as regras

4.6 Passo 5 - SSH

Por uma terceira vez, depois de atualizar as regras da *firewall*, conseguimos conectar ao servidor SSH que está a correr na máquina servidor, e fazer login.

```

pedro@pedro:~/Desktop$ ssh pedro@10.0.0.7
The authenticity of host '10.0.0.7 (10.0.0.7)' can't be established.
ECDSA key fingerprint is SHA256:HNd8xElovwT8aQTuLZ5hUr2mciEHA3wIykZr6Xr/KDc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.0.7' (ECDSA) to the list of known hosts.
pedro@10.0.0.7's password:
Last login: Mon Dec 14 13:24:29 2020
[pedro@localhost ~]$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
[pedro@localhost ~]$

```

Figura 13: Tentativa de conexão SSH, depois de atualizar as regras

4.7 Passo 6 - nmap

Com a ferramenta nmap, conseguimos verificar que de facto as portas 21, 22 e 80 estão abertas na máquina servidor.

```

pedro@pedro:~/Desktop$ sudo nmap -sS 10.0.0.7
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-15 18:20 WET
Nmap scan report for 10.0.0.7
Host is up (0.00099s latency).
Not shown: 997 filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 08:00:27:39:75:B0 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 18.15 seconds

```

Figura 14: Output de "nmap -sS" depois de atualizar as regras da firewall

Podemos até pedir ao nmap para tentar identificar as versões específicas dos serviços que estão a correr na máquina servidor, processo que tem sucesso.

```
pedro@pedro:~/Desktop$ sudo nmap -sV 10.0.0.7
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-15 18:21 WET
Nmap scan report for 10.0.0.7
Host is up (0.00071s latency).
Not shown: 997 filtered ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.2
22/tcp    open  ssh      OpenSSH 7.4 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.6 ((CentOS))
MAC Address: 08:00:27:39:75:B0 (Oracle VirtualBox virtual NIC)
Service Info: OS: Unix

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 26.17 seconds
```

Figura 15: Output de "nmap -sV" depois de atualizar as regras da firewall

Uma experiência interessante, é tentar fazer um *scan* de "ping". Apesar da filtragem aplicada pela *firewall*, o nmap reconhece na mesma que o *host* está *online*.

```
pedro@pedro:~/Desktop$ sudo nmap -sn 10.0.0.7
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-15 18:26 WET
Nmap scan report for 10.0.0.7
Host is up (0.00055s latency).
MAC Address: 08:00:27:39:75:B0 (Oracle VirtualBox virtual NIC)
Nmap done: 1 IP address (1 host up) scanned in 13.09 seconds
```

Figura 16: Output de "nmap -sn" depois de atualizar as regras da firewall

E claro, podemos sempre fazer scans ICMP de outros tipos que não "ping", por exemplo, usando "ECHO".

```

pedro@pedro:~/Desktop$ sudo nmap -PE 10.0.0.7
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-15 18:33 WET
Nmap scan report for 10.0.0.7
Host is up (0.00077s latency).
Not shown: 997 filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 08:00:27:39:75:B0 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 20.12 seconds

```

Figura 17: Output de "nmap -PE" depois de atualizar as regras da firewall

4.8 Passo 7 - Verificar as tabelas depois de algum tráfego

```

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source            destination
279 28090 ACCEPT      all  --  any    any     anywhere          anywhere
20  1680 REJECT      icmp --  any    any     anywhere          anywhere
0    0 ACCEPT      icmp --  any    any     anywhere          anywhere
0    0 ACCEPT      all  --  lo     any     anywhere          anywhere
7   356 ACCEPT      tcp  --  any    any     anywhere          anywhere
13  716 ACCEPT      tcp  --  any    any     anywhere          anywhere
6   296 ACCEPT      tcp  --  any    any     anywhere          anywhere
7957 350K REJECT      all  --  any    any     anywhere          anywhere
                                state NEW tcp dpt:ssh
                                state NEW tcp dpt:http
                                state NEW tcp dpt:ftp
                                reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source            destination
0    0 REJECT      all  --  any    any     anywhere          anywhere
                                reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT 332 packets, 59500 bytes)
pkts bytes target      prot opt in     out     source            destination

```

Figura 18: Tabelas depois de algum tráfego

Como podemos ver na primeira coluna da tabela, o número de pacotes que foram aceites ou rejeitados pelas regras existentes subiu. Como esperado as 3 regras referentes aos 3 protocolos que usamos para teste têm valores muito baixos, visto que estas regras só intervêm no início da conexão. Já a primeira regra da tabela tem um valor bastante mais elevado, visto que trata de todos os pacotes que vêm nessas mesmas streams TCP.

A segunda linha que rejeita os pedidos "ping" tem também um valor maior que 0, tanto dos pings que fizemos como do scan do nmap.

No entanto a linha com maior valor, e por uma margem muito grande, é a última. Como fizemos vários *scans* com o "nmap", e cada um deles procura em 1000 portas, 997 das quais estão filtradas, este valor vai ser muito grande, pois cada *scan* do nmap vai incrementá-lo por esse valor. Para além disso, já tínhamos outros pacotes rejeitados nos testes anteriores.

5 Tarefa 4

Como exercício final, melhoramos um pouco a segurança do nosso sistema, restringindo os pedidos SSH, FTP e HTTP apenas a endereços locais, ou seja, endereços do *range* 10.0.0.0/24, e fazendo *log* de qualquer tentativa de conexão que quebre esta regra.

Para ajudar nesta tarefa, usamos o GUI da ferramenta que usamos no início deste exercícios, "system-config-firewall".

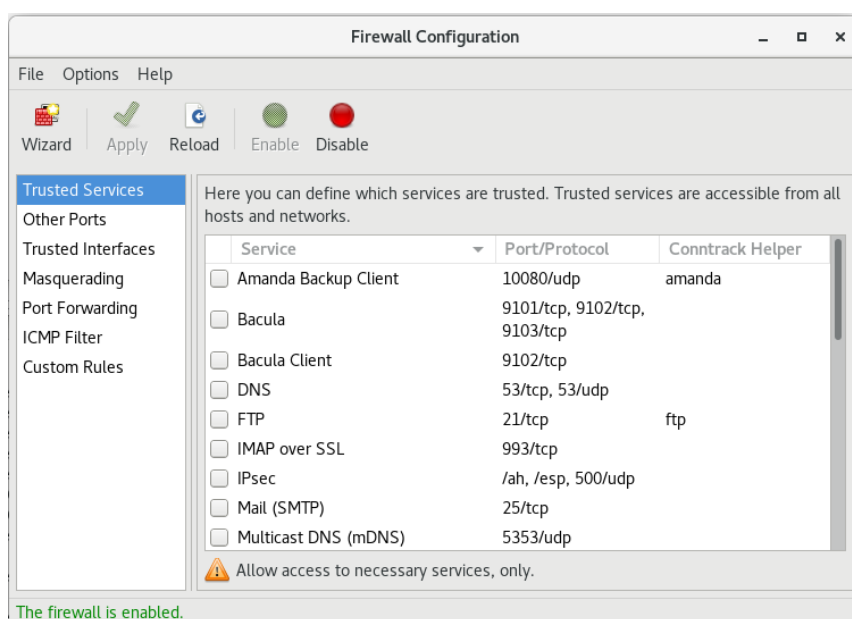


Figura 19: Aplicação GUI para gestão da firewall

Como a nossa configuração da *firewall* é bastante simples, há pouco a fazer na aplicação em si. Apenas tiver que ativar a recusa do pedido "ping", na secção "ICMP filter".

O que tivemos que fazer foi criar regras customizadas, que o software integra nas tabelas. Para esse efeito, temos que criar um ficheiro que contém a lista de regras que queremos implementar, que o software acrescentará às regras que ele cria por defeito. O ficheiro resultante é o seguinte.

```
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -m iprange --src-range 10.0.0.3-10.0.0.255 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -m iprange --src-range 10.0.0.3-10.0.0.255 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 21 -m iprange --src-range 10.0.0.3-10.0.0.255 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j LOG --log-prefix "External SSH attempt"
-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j LOG --log-prefix "External HTTP attempt"
-A INPUT -p tcp -m state --state NEW -m tcp --dport 21 -j LOG --log-prefix "External FTP attempt"
```

Figura 20: Regras customizadas para introdução na firewall

As primeiras três regras definem que os pedidos SSH, FTP e HTTP só devem ser aceites se o endereço de *source* pertencer ao bloco 10.0.0.3 a 10.0.0.255. Isto foi feito para podermos testar esta implementação, visto que dentro da rede virtual criada pelo "Virtual Box" a máquina *host* tem o endereço 10.0.0.1.

As seguintes 3 regras definem que pedidos SSH, HTTP e FTP devem ser logados, com um prefixo adequado para ser fácil de os encontrar no ficheiro de *log*, que cresce muito rapidamente, já que é partilhado por outras aplicações em distribuições de Linux baseadas em "Red Hat". Como estas regras aparecem depois das anteriores, isto significa que apenas pedidos que não sejam dos endereços corretos serão logados já que, como já vimos, as regras só são verificadas até ao primeiro *match*.

Como seria de esperar, uma ligação SSH, por exemplo, continua a funcionar a partir da máquina cliente. No entanto, se tentarmos fazer o mesmo a partir da máquina *host*, recebemos um erro.

```
pedro@pedro-N751JK:~$ ssh pedro@10.0.0.7
ssh: connect to host 10.0.0.7 port 22: No route to host
pedro@pedro-N751JK:~$ ssh pedro@10.0.0.7
ssh: connect to host 10.0.0.7 port 22: No route to host
```

Figura 21: Aplicação GUI para gestão da firewall

Para além disso, se verificarmos o ficheiro de logs na máquina cliente, encontramos as seguintes duas entradas.

```
[pedro@localhost Documents]$ sudo cat /var/log/messages | grep "External SSH"
Dec 15 17:33:23 localhost kernel: External SSH attemptIN=enp0s8 OUT= MAC=08:00:27:39:75:b0:0a:00:27:00:00:00:08:00 SRC=10.0.0.1
DST=10.0.0.7 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=42976 DF PROTO=TCP SPT=59992 DPT=22 WINDOW=64240 RES=0x00 SYN URG=0
Dec 15 17:35:30 localhost kernel: External SSH attemptIN=enp0s8 OUT= MAC=08:00:27:39:75:b0:0a:00:27:00:00:00:08:00 SRC=10.0.0.1
DST=10.0.0.7 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=55826 DF PROTO=TCP SPT=60014 DPT=22 WINDOW=64240 RES=0x00 SYN URG=0
```

Figura 22: Aplicação GUI para gestão da firewall

Acompanhado da data do incidente e da tag definida na regra, temos informações sobre o pacote IP em si, tal como os endereços IP e MAC, TTL entre outros.

6 Conclusão

Neste trabalho foi nos possível aprofundar os conhecimentos relativos à implementação de regras numa *firewall* usando a aplicação ***iptables*** num ambiente Linux. Nomeadamente, a realização das várias tarefas propostas no enunciado foram todas, na nossa ótica, concluídas com sucesso, dado que pensamos ter atingido os principais objetivos deste Trabalho Prático e até desenvolver alguns extras que achamos que seriam úteis para perceber como facilmente uma *firewall* pode evoluir no número de regras e que a criação de boas regras de segurança não são muito fáceis de fazer.

Assim, as *firewalls* são um meio de introduzir segurança no nível de rede (nível 2, de acordo com o modelo TCP/IP) e introduzem restrições aos pacotes que passam por esta camada, pelo que possíveis ataques à infraestrutura, tais como o uso do ***nmap*** para obter informações como "*port scan*", possam ser impedidos de serem concretizados.

A iptables.dump

Conteúdo do ficheiro "*iptables.dump*".

```
# Generated by iptables-save v1.4.21 on Mon Dec 14 16:10:02
2020
*mangle
:PREROUTING ACCEPT [1161:98183]
:INPUT ACCEPT [1161:98183]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [1327:89980]
:POSTROUTING ACCEPT [1393:96884]
-A POSTROUTING -o virbr0 -p udp -m udp --dport 68 -j CHECKSUM
--checksum-fill
COMMIT
# Completed on Mon Dec 14 16:10:02 2020
# Generated by iptables-save v1.4.21 on Mon Dec 14 16:10:02
2020
*nat
:PREROUTING ACCEPT [16:1836]
:INPUT ACCEPT [4:336]
:OUTPUT ACCEPT [734:60090]
:POSTROUTING ACCEPT [734:60090]
-A POSTROUTING -s 192.168.122.0/24 -d 224.0.0.0/24 -j RETURN
-A POSTROUTING -s 192.168.122.0/24 -d 255.255.255.255/32 -j
RETURN
-A POSTROUTING -s 192.168.122.0/24 ! -d 192.168.122.0/24 -p
tcp -j MASQUERADE --to-ports 1024-65535
-A POSTROUTING -s 192.168.122.0/24 ! -d 192.168.122.0/24 -p
udp -j MASQUERADE --to-ports 1024-65535
-A POSTROUTING -s 192.168.122.0/24 ! -d 192.168.122.0/24 -j
MASQUERADE
COMMIT
# Completed on Mon Dec 14 16:10:02 2020
# Generated by iptables-save v1.4.21 on Mon Dec 14 16:10:02
2020
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [1327:89980]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
```

```
-A INPUT -i lo -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
# Completed on Mon Dec 14 16:10:02 2020
```