



Universidade do Minho
Escola de Engenharia

GESTÃO E VIRTUALIZAÇÃO DE REDES
SEGURANÇA EM REDES
NETWORK SECURITY
(SR) - HOMEWORK TP4

PRÁTICAS COM FIREWALLS (IPTABLES)

GRUPO 2

| | |
|--------|--|
| A85308 | Filipe Miguel Teixeira Freitas Guimarães |
| A79799 | Gonçalo Nogueira Costeira |
| A84912 | Joana Isabel Afonso Gomes |
| A75480 | Marco Matias Pereira Gonçalves |
| A42040 | Miriam Miranda Pinto |
| A57041 | Simão Pedro Santa Cruz Oliveira |

Braga,
30 de novembro de 2020

Conteúdo

| | | |
|----------|--|-----------|
| 1 | Introdução & Contextualização | 2 |
| 2 | Resposta ao problema proposto | 3 |
| 2.1 | Tarefa 1 | 3 |
| 2.1.1 | Ponto 1 | 3 |
| 2.1.2 | Ponto 2 | 7 |
| 2.1.3 | Ponto 3 | 8 |
| 2.1.4 | Ponto 4 | 9 |
| 2.1.5 | Ponto 4 | 9 |
| 2.1.6 | Ponto 5 | 10 |
| 2.2 | Tarefa 2 | 10 |
| 2.2.1 | Ponto 1 | 10 |
| 2.2.2 | Ponto 2 | 11 |
| 2.2.3 | Ponto 3 | 12 |
| 2.2.4 | Ponto 4 | 13 |
| 2.2.5 | Ponto 5 | 13 |
| 2.3 | Tarefa 3 | 14 |
| 2.3.1 | Ponto 1 | 14 |
| 2.3.2 | Ponto 2 | 14 |
| 2.3.3 | Ponto 3 | 15 |
| 2.3.4 | Ponto 4 | 16 |
| 2.3.5 | Ponto 5 | 16 |
| 2.3.6 | Ponto 6 | 16 |
| 2.3.7 | Ponto 7 | 17 |
| 2.4 | Exercício de conclusão | 18 |
| 3 | Conclusão | 22 |

1 Introdução & Contextualização

No âmbito da Unidade Curricular de Segurança em Redes foi-nos proposto este trabalho prático, com incidência na temática de *Firewalls*, utilizando o conceito de *IPtables*, de forma a ser possível configurar corretamente uma *Firewall* que apresenta um determinado tipo de segurança.

Delineamos, assim, que haveria necessidade de fazer um estudo prévio da teoria interligada a este guião, com a recurso essencialmente ao material de suporte fornecido.

A *Firewall* é um recurso informático fundamental formado por software e hardware específicos, em que o seu papel fulcral é estabelecer segurança entre duas redes, usualmente a nossa privada e o nosso *Internet Provider*. Torna-se, assim, um sistema de segurança utilizado como forma de proteção contra software possivelmente malicioso, permitindo apenas que as aplicações que têm permissões possam atravessá-la.

À semelhança daquilo que é apresentado como nota introdutória deste trabalho, numa *Firewall* é possível usar a tabela *filter*, que permite dividir os pacotes de dados essencialmente em três cadeias:

- **Input:** aplica-se ao tráfego de entrada na máquina local.
- **Forward:** aplica-se ao tráfego de pacotes que entram na máquina local e que vão "sair", com destino a outras máquinas.
- **Output:** aplica-se ao tráfego criado na máquina local e que se destina a outras máquinas.

O comando *iptables* é constituído por várias *tables* (camadas) as e pelas suas cadeias, possibilitando a análise do tráfego de rede recebido. Das camadas existentes, destacamos (em relevância para o trabalho em questão) :

- **Filter(Input, Forward e Output)** - responsável pela aceitação (ou não) de um determinado pacote. Possibilita realizar ACCEPT ou REJECT / DROP (aceitar ou rejeitar pacotes) e LOG (pacotes ainda não estão sujeitos a nenhuma das três ações possíveis e aguardam a respetiva ação de que serão alvo).
- **NAT(Network Address Translation)** responsável por traduzir os endereços que passam pelo *router* em que se encontram e verificam se existiu alguma alteração a nível dos IP's origem e destino.

De seguida iremos apresentar com maior detalhe a metodologia seguida e a respetiva aplicação.

2 Resposta ao problema proposto

De forma a reproduzir o que é solicitado no enunciado decidimos optar usar o *CentOS 6* como servidor (transferindo a imagem em <https://www.osboxes.org/>) e o *Kali* como cliente (transferindo a imagem em <https://www.offensive-security.com/>). Sendo que usamos o *CentOS 6* os comandos vão ser idênticos aos exemplificados no enunciado.

Ambos os sistemas foram configurados para funcionar em *NAT Network* na *VirtualBox* de forma a que consigam ter diferentes endereços IP (o que não acontece em *NAT*).

2.1 Tarefa 1

2.1.1 Ponto 1

Instalamos então os serviços desejados, como se pode ver nas figuras abaixo. Alguns destes já se encontravam presentes.

```
[root@localhost ~]# yum install httpd
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Install Process
Loading mirror speeds from cached hostfile
Package httpd-2.2.15-69.el6.centos.x86_64 already installed and latest version
Nothing to do
```

Figura 1: Instalação do serviço *HTTP*

```
[root@localhost ~]# yum install ftp
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Install Process
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
--> Package ftp.x86_64 0:0.17-54.el6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package Arch Version Repository Size
=====
Installing:
ftp x86_64 0.17-54.el6 base 58 k
Transaction Summary
-----
Install 1 Package(s)

Total download size: 58 k
Installed size: 95 k
Is this ok [y/N]: y
Downloading Packages:
ftp-0.17-54.el6.x86_64.rpm | 58 kB 00:00
Running rpm_check debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
Installing : ftp-0.17-54.el6.x86_64 1/1
Verifying : ftp-0.17-54.el6.x86_64 1/1

Installed:
ftp.x86_64 0:0.17-54.el6

Complete!
```

Figura 2: Instalação do serviço *FTP*

```
[root@localhost ~]# yum install openssh-server openssh-clients
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Install Process
Loading mirror speeds from cached hostfile
Package openssh-server-5.3p1-124.el6_10.x86_64 already installed and latest version
Package openssh-clients-5.3p1-124.el6_10.x86_64 already installed and latest version
Nothing to do
```

Figura 3: Instalação do serviço *SSH*

```
[root@localhost ~]# yum install httpd
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Install Process
Loading mirror speeds from cached hostfile
Package httpd-2.2.15-69.el6.centos.x86_64 already installed and latest version
Nothing to do
```

Figura 4: Instalação do serviço *httpd*

```
[root@localhost ~]# yum install vsftpd
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Install Process
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
--> Package vsftpd.x86_64 0:2.2.2-24.el6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch           Version           Repository        Size
=====
Installing:
vsftpd            x86_64         2.2.2-24.el6      base              156 k
=====

Transaction Summary
=====
Install          1 Package(s)

Total download size: 156 k
Installed size: 340 k
Is this ok [y/N]: y
Downloading Packages:
vsftpd-2.2.2-24.el6.x86_64.rpm | 156 kB    00:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : vsftpd-2.2.2-24.el6.x86_64              1/1
  Verifying  : vsftpd-2.2.2-24.el6.x86_64
Installed:
vsftpd.x86_64 0:2.2.2-24.el6

Complete!
```

Figura 5: Instalação do serviço *vsftpd*

Prosseguimos então para a ativação dos mesmos (em *System -> Administration -> Services*).

The **httpd** service is started once, usually when the system is booted, runs in the background and wakes up when needed.

- This service is enabled.
- 🔊 This service is running.

Figura 6: Ativação do serviço *httpd*

The **sshd** service is started once, usually when the system is booted, runs in the background and wakes up when needed.

- This service is enabled.
- 🔊 This service is running.

Figura 7: Ativação do serviço *sshd*

The **vsftpd** service is started once, usually when the system is booted, runs in the background and wakes up when needed.

- This service is enabled.
- 🔊 This service is running.

Figura 8: Ativação do serviço *vsftpd*

Usamos então o *netstat -l* para verificar se os serviços estão preparados.

```
[root@localhost ~]# netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      0 *:sunrpc                *:.*                    LISTEN
tcp      0      0 *:ftp                   *:.*                    LISTEN
tcp      0      0 *:ssh                   *:.*                    LISTEN
tcp      0      0 localhost.localdomain:ipp *:.*                    LISTEN
tcp      0      0 *:36471                 *:.*                    LISTEN
tcp      0      0 localhost.localdomain:smtp *:.*                    LISTEN
tcp      0      0 *:sunrpc                *:.*                    LISTEN
tcp      0      0 *:http                  *:.*                    LISTEN
tcp      0      0 *:ssh                   *:.*                    LISTEN
tcp      0      0 localhost6.localdomain6:ipp *:.*                    LISTEN
tcp      0      0 *:52845                 *:.*                    LISTEN
udp      0      0 *:45619                 *:.*                    LISTEN
udp      0      0 *:bootpc                *:.*                    LISTEN
udp      0      0 *:sunrpc                *:.*                    LISTEN
udp      0      0 *:ipp                   *:.*                    LISTEN
udp      0      0 10.0.2.15:ntp           *:.*                    LISTEN
udp      0      0 localhost.localdomain:ntp *:.*                    LISTEN
udp      0      0 *:ntp                   *:.*                    LISTEN
udp      0      0 localhost.localdomain:778 *:.*                    LISTEN
udp      0      0 *:vacdsm-app            *:.*                    LISTEN
udp      0      0 *:53957                 *:.*                    LISTEN
udp      0      0 *:sunrpc                *:.*                    LISTEN
udp      0      0 fe80::a00:27ff:fec7:3ff8:ntp *:.*                    LISTEN
udp      0      0 localhost6.localdomain6:ntp *:.*                    LISTEN
udp      0      0 *:ntp                   *:.*                    LISTEN
udp      0      0 *:vacdsm-app            *:.*                    LISTEN
```

Figura 9: *netstat -l*

Conseguimos ainda verificar o http, ftp e ssh no localhost.

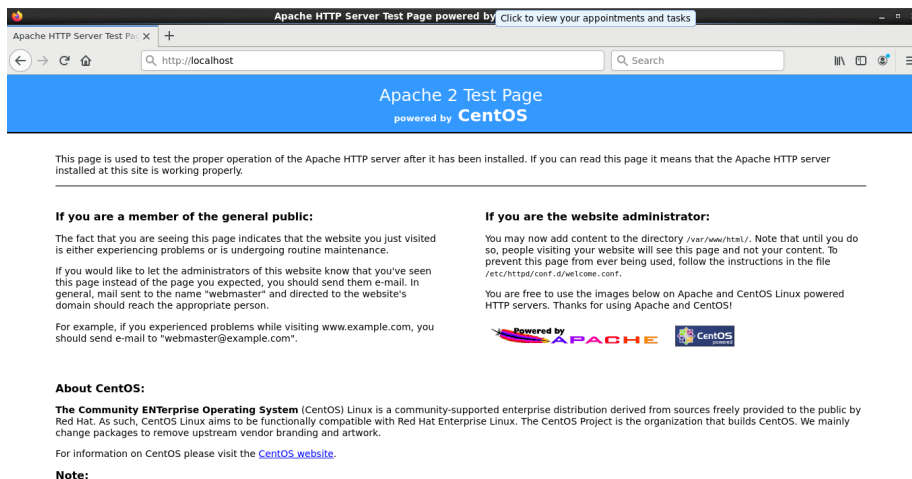


Figura 10: Página *http* no localhost

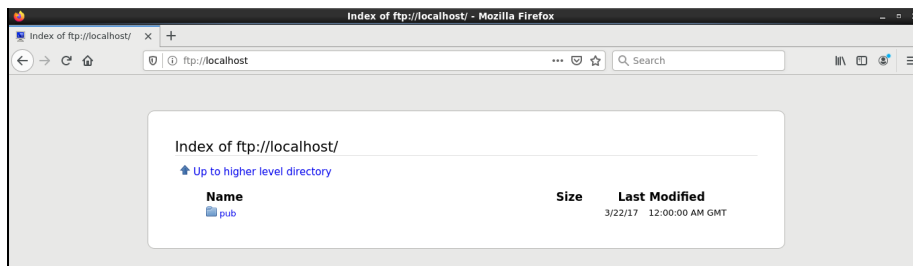


Figura 11: Página *ftp* no localhost

```
[osboxes@localhost ~]$ ftp localhost
Connected to localhost (127.0.0.1).
220 (vsFTPd 2.2.2)
Name (localhost:osboxes): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> exit
221 Goodbye.
```

Figura 12: *ftp* no localhost no terminal

```
[root@localhost ~]# ssh localhost
root@localhost's password:
Last login: Mon Dec 14 20:13:28 2020 from localhost.localdomain
```

Figura 13: *ssh* no localhost no terminal

2.1.2 Ponto 2

Como referido, estamos a utilizar *CentoOS 6* e, por isso, temos à nossa disposição o *system-config-firewall-tui*. Ao executar este comando verificamos que a *firewall* já está ativa não sendo preciso fazer nada neste ponto.

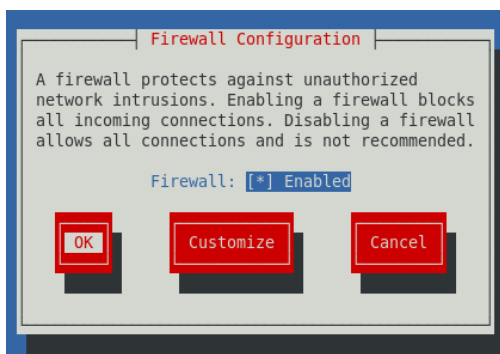


Figura 14: Ativação da firewall

2.1.3 Ponto 3

Para a execução do *iptables*, e como é referido no enunciado, é necessário permissões de super utilizador, para isso entramos neste modo fazendo *"su -"* e introduzindo a password da máquina. O resultado do comando *"iptables -L -v"* é o mostrado na figura 15.

```
[root@localhost ~]# iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out    source            destination        state
169 14428 ACCEPT      all  --  any    any    anywhere          anywhere           state RELATED,ESTABLISHED
0      0 ACCEPT      icmp --  any    any    anywhere          anywhere
2    400 ACCEPT      all  --  lo     any    anywhere          anywhere
0      0 ACCEPT      tcp  --  any    any    anywhere          anywhere           state NEW tcp dpt:ssh
0      0 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        state
0      0 REJECT      all  --  any    any    anywhere          anywhere           reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT 171 packets, 14828 bytes)
pkts bytes target      prot opt in     out    source            destination
```

Figura 15: Políticas por defeito na *firewall*

Algumas das regras das cadeias, utilizam a tabela *conntrack*, que passamos a expôr uma breve descrição da sua função.

Conntrack é uma tabela onde são armazenadas todas as conexões que passam pelo servidor, sejam as mesmas de dentro ou de fora dele.

Através dessa tabela, o *firewall* consegue identificar o estado de determinada conexão, que pode ser *NEW*, *ESTABLISHED*, *RELATED* ou *INVALID*.

A tabela *NAT* do *iptables* trabalha diretamente com a tabela *conntrack*, sem ela por exemplo, não conseguiríamos fazer com que diversos *hosts* internos navegassem através de um único IP válido, já que o *conntrack* é que garante que na volta dos pacotes, o mesmo seja retornado para o devido IP que o solicitou. Ou seja, o *conntrack* é essencial para o funcionamento do *NAT*.

As regras das cadeias são:

- **Input**

- São aceites pacotes de qualquer origem para qualquer destino, considerando que o seu estado na tabela *conntrack* seja *RELATED* ou *ESTABLISHED*;
- São aceites pacotes de qualquer origem para qualquer destino, desde que o protocolo em vigor seja o ICMP;
- São aceites pacotes de si próprio, ou seja, do *localhost* (lo);
- São aceites pacotes de qualquer origem para qualquer destino desde que sejam pedidos *SSH*, através do protocolo *TCP*. Na tabela *conntrack* o seu estado deve ser *NEW*;
- Qualquer outro serviço terá o seu pedido de acesso rejeitado e será notificado com uma mensagem de erro **icmp-host-prohibited**.

- **Forward**

- São rejeitados todos os pacotes e todos os protocolos de qualquer origem para qualquer destino. A *firewall* não permite que o servidor reencaminhe os pacotes que recebe com destino a outros *hosts* gerando como mensagem de erro **icmp-host-prohibited**.

- **Output**

- Não existem regras de cadeia, ou seja, como podemos ver na figura 15, são aceites pacotes de qualquer origem para qualquer destino sem qualquer tipo de restrição.

Conseguimos ver que é uma implementação de uma *firewall* que não é complexa, relativamente ao nível de segurança podemos detetar algumas vulnerabilidades.

O dispositivo encontra-se vulnerável a ataques por *ICMP flood* ou seja ataques *DDoS*, devido à inexistência de filtros que restrinjam que, por exemplo, qualquer cliente faça *ping*.

Temos ainda o serviço *SSH* sempre disponível, o que permite que qualquer cliente se ligue ao servidor através do mesmo conseguindo, usando um ataque de força bruta, tomar conta do servidor.

2.1.4 Ponto 4

De forma a guardar a configuração atual da *firewall* em caso de complicações fizemos então *iptables-save > iptables.dump* de forma a podermos voltar a este ponto caso seja necessário.

```
[root@localhost ~]# iptables-save > iptables.dump
[root@localhost ~]# ls
anaconda-ks.cfg  iptables.dump  post-install  post-install.log
```

Figura 16: Dump à *firewall*

2.1.5 Ponto 4

Para desativar a *firewall* usamos, mais uma vez, o *firewall-config-firewall-tui* verificando que houve uma alteração nas regras presentes quando executamos *iptables -L -v*.

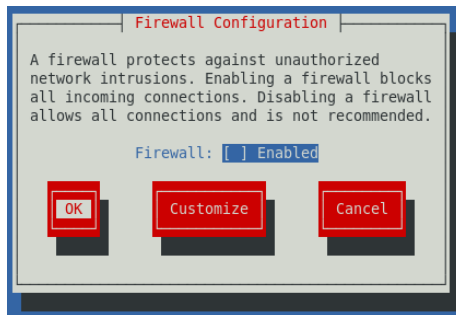


Figura 17: Desativar a *firewall*

```
[root@localhost ~]# iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 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 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination
```

Figura 18: *iptables* com a *firewall* desativada

Como se pode verificar as regras mudaram. Agora é permitido todo o tráfego *Input*, *Forward* e *Output*.

2.1.6 Ponto 5

Voltamos então a ligar a *firewall* recorrendo ao *system-config-firewall-tui*.

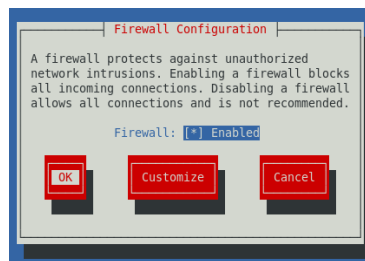


Figura 19: Reativação da *firewall*

2.2 Tarefa 2

2.2.1 Ponto 1

Para verificar a conectividade fizemos *ping* no cliente para o servidor. Para sabermos o IP do servidor usamos o comando *ip addr* descobrindo que usa o endereço *10.0.2.4*.

```
[root@localhost ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:b2:38:d7 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.4/24 brd 10.0.2.255 scope global eth1
    inet6 fe80::a00:27ff:feb2:38d7/64 scope link
        valid_lft forever preferred_lft forever
```

Figura 20: *ip addr* no servidor

```
→ ~ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=1.88 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=64 time=1.19 ms
^C
--- 10.0.2.4 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1010ms
rtt min/avg/max/mdev = 1.194/1.536/1.879/0.342 ms
```

Figura 21: *Ping* ao servidor

Recebemos a resposta ao *ping* o que quer dizer que conseguimos ter conexão ao servidor.

2.2.2 Ponto 2

Fizemos o *nmap* ao servidor como se vê na figura 22.

```
→ ~ sudo nmap -sS 10.0.2.4
[sudo] password for kali:
Starting Nmap 7.91 ( https://nmap.org ) at 2020-12-15 09:47 EST
Nmap scan report for 10.0.2.4
Host is up (0.0014s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: 08:00:27:B2:38:D7 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 5.33 seconds
```

Figura 22: *nmap -sS* ao servidor

Conseguimos perceber com o *output* do *nmap* que o servidor está ligado, que conseguiu descobrir 1000 portas e, como era esperado, uma delas está aberta, aquela que é usada para comunicações com o serviço *SSH*.

Para explorar ainda mais esta ferramenta recorreremos ao *help nmap* para saber se há algum argumento que nos daria mais informações úteis. Encontramos o *-sV* que nos fornece a versão dos serviços com portas abertas como se pode ver na figura 24.

```
SERVICE/VERSION DETECTION:
-sV: Probe open ports to determine service/version info
```

Figura 23: *help -sV*

```
→ ~ sudo nmap -sV 10.0.2.4
Starting Nmap 7.91 ( https://nmap.org ) at 2020-12-15 09:53 EST
Nmap scan report for 10.0.2.4
Host is up (0.0076s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 5.3 (protocol 2.0)
MAC Address: 08:00:27:B2:38:D7 (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 5.61 seconds
```

Figura 24: *nmap -sV* ao servidor

Deparamo-nos também com o argumento *-A* que nos fornece todas as informações já encontradas e também a versão do *Linux* e *Kernel* instalados na máquina destino.

```
-A: Enable OS detection, version detection, script scanning, and traceroute
```

Figura 25: *help -A*

```
→ ~ sudo nmap -A 10.0.2.4
Starting Nmap 7.91 ( https://nmap.org ) at 2020-12-15 09:55 EST
Nmap scan report for 10.0.2.4
Host is up (0.00088s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 5.3 (protocol 2.0)
| ssh-hostkey:
|_  1024 84:93:7b:5a:26:90:bd:99:89:64:95:9b:f9:16:c2:10 (DSA)
|_  2048 d9:ae:d0:03:d0:8a:bb:23:cb:49:39:2d:e0:c0:49:99 (RSA)
MAC Address: 08:00:27:B2:38:D7 (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.32 - 3.10, Linux 2.6.32 - 3.13, Linux 3.4 - 3.10
Network Distance: 1 hop

TRACEROUTE
HOP RTT      ADDRESS
1   0.88 ms  10.0.2.4

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 8.46 seconds
```

Figura 26: *nmap -A* ao servidor

2.2.3 Ponto 3

Ao tentar aceder ao *http://10.0.2.4* reparamos, como era esperado por estar a ser bloqueado pela *firewall*, que não conseguimos.

```
→ ~ w3m http://10.0.2.4
w3m: Can't load http://10.0.2.4.
```

Figura 27: *w3m http://10.0.2.4*

2.2.4 Ponto 4

Mais uma vez não obtivemos resposta do servidor a tentar fazer *ftp 10.0.2.4*, como era esperado porque também é bloqueado pela *firewall*.

```
→ ~ ftp 10.0.2.4
ftp: connect: No route to host
ftp> █
```

Figura 28: *w3m http://10.0.2.4*

2.2.5 Ponto 5

Desta vez, ao fazer *ssh 10.0.2.4* temos resposta do servidor. Ao introduzir a password usada no servidor conseguimos aceder aos ficheiros presentes na máquina. Isto era previsível visto que já tínhamos verificado que esta porta encontra-se aberta.

```
→ ~ sudo ssh 10.0.2.4
root@10.0.2.4's password:
Last login: Tue Dec 15 15:05:32 2020 from 10.0.2.15
[root@localhost ~]# ls
anaconda-ks.cfg  iptables.dump  post-install  post-install.log
[root@localhost ~]# cat iptables.dump
# Generated by iptables-save v1.4.7 on Tue Dec 15 13:55:42 2020
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [446:43154]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
# Completed on Tue Dec 15 13:55:42 2020
[root@localhost ~]# exit
logout
Connection to 10.0.2.4 closed.
```

Figura 29: *w3m http://10.0.2.4*

2.3 Tarefa 3

2.3.1 Ponto 1

Voltamos a recorrer ao comando *system-config-firewall-tui* desta vez para autorizar novas portas (*FTP*, *SSH*, *WWW (HTTP)*). Rejeitamos ainda no protocolo *ICMP* os *ECHO REQUEST* (ping). Pode ver-se estas alterações nas figuras abaixo.

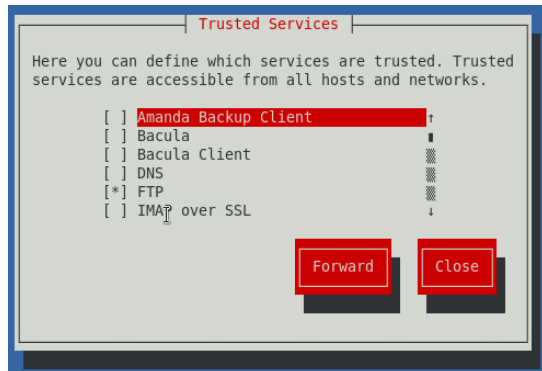


Figura 30: *FTP*

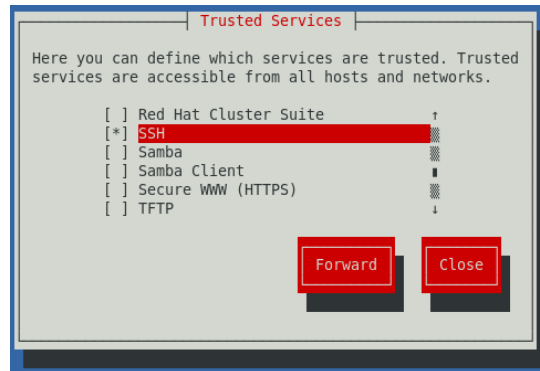


Figura 31: *SSH*

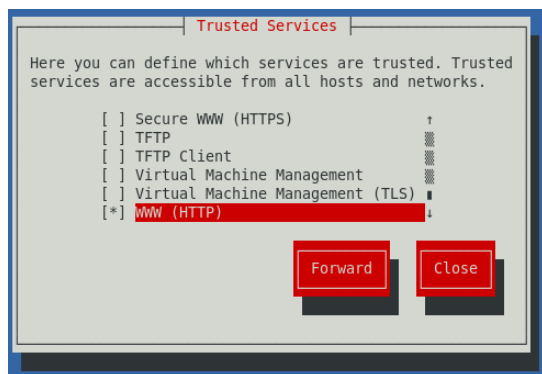


Figura 32: *WWW (HTTP)*

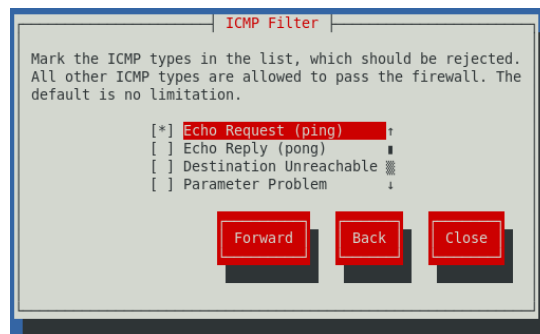


Figura 33: *ECHO REQUEST* (ping)

2.3.2 Ponto 2

Após as alterações nas configurações da *firewall* voltamos a ver as *iptables* para perceber as diferenças que existem face ao resultado anterior.

```
[root@localhost ~]# iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination state
20 2020 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 anywhere
0 0 ACCEPT all -- lo any anywhere 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
0 0 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 21 packets, 1848 bytes)
pkts bytes target prot opt in out source destination
```

Figura 34: *iptables -L -v*

Conseguimos perceber pela análise da figura 34 que apenas a cadeia de acesso (*INPUT*) sofreu alterações, onde foram adicionadas três regras às já existentes e mencionadas na Tarefa 1, Ponto 3.

- **Input**

- São aceites pacotes de qualquer origem para qualquer destino, considerando que o seu estado na tabela *conntrack* seja *RELATED* ou *ESTABLISHED*;
- São rejeitados todos os pacotes *echo-request* através do protocolo ICMP, gerando uma mensagem de erro ***reject-with icmp-host-prohibited***;
- São aceites pacotes de qualquer origem para qualquer destino, desde que o protocolo em vigor seja o ICMP;
- São aceites pacotes de si próprio, ou seja, do *localhost* (lo);
- São aceites pacotes de qualquer origem para qualquer destino desde que sejam pedidos *SSH*, *FTP* e *HTTP* através do protocolo *TCP*. Na tabela *conntrack* o seu estado deve ser *NEW* para cada um destes serviços;
- Qualquer outro serviço terá o seu pedido de acesso rejeitado e será notificado com uma mensagem de erro ***icmp-host-prohibited***.

Podemos constatar que a nível das cadeias *FORWARD* e *OUTPUT* não se verificaram alterações.

2.3.3 Ponto 3

Desta vez ao realizar o *ping* ao nível do cliente já não obtivemos resposta do servidor mas sim uma mensagem de erro **HOST PROHIBITED**, visto que bloqueamos esta funcionalidade na *firewall* (resultado esperado).

```
➔ ~ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
From 10.0.2.4 icmp_seq=1 Destination Host Prohibited
From 10.0.2.4 icmp_seq=2 Destination Host Prohibited
From 10.0.2.4 icmp_seq=3 Destination Host Prohibited
^C
--- 10.0.2.4 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2008ms
```

Figura 35: Novo *ping 10.0.2.4*

2.3.4 Ponto 4

Ao tentar aceder ao servidor via *w3m* *http://10.0.2.4* conseguimos, ao contrario do que acontecia antes, visualizar uma página.

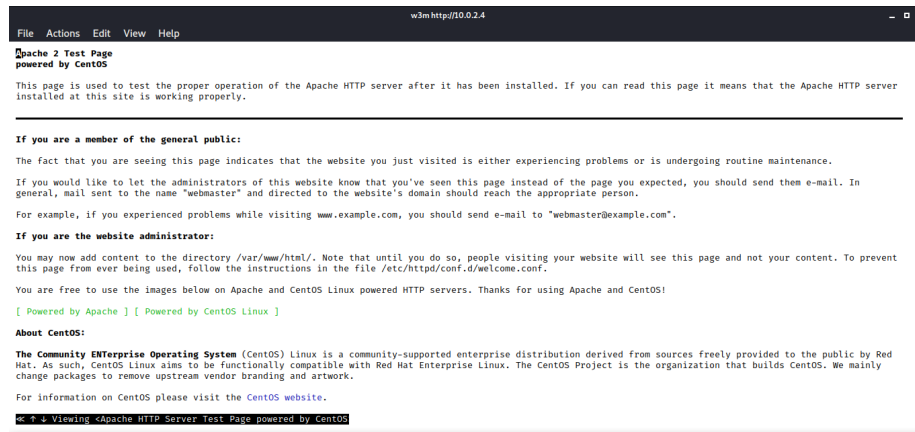


Figura 36: *w3m http://10.0.2.4*

Como deixamos a *firewall* aceites os pedidos *www (http)* isto é o esperado.

2.3.5 Ponto 5

À semelhança do ponto anterior e o que não se verificava antes, agora conseguimos aceder ao servidor via *FTP* (como era esperado visto que configuramos a *firewall* para aceitar este tipo de pedidos).

```
→ ~ ftp 10.0.2.4
Connected to 10.0.2.4.
220 (vsFTPd 2.2.2)
Name (10.0.2.4:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x  2 0          0          4096 Mar 22  2017 pub
226 Directory send OK.
ftp> exit
221 Goodbye.
```

Figura 37: *ftp 10.0.2.4*

2.3.6 Ponto 6

Executando o comando *nmap -sS 10.0.2.4*, desta vez, conseguimos perceber que temos mais duas portas abertas. Agora estão também as portas 21 e 80, correspondentes, respetivamente, aos serviços *ftp* e *http*.

```

→ ~ sudo nmap -sS 10.0.2.4
[sudo] password for kali:
Starting Nmap 7.91 ( https://nmap.org ) at 2020-12-15 10:34 EST
Nmap scan report for 10.0.2.4
Host is up (0.0011s 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:B2:38:D7 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 5.36 seconds

```

Figura 38: *nmap -sS 10.0.2.4*

Ao nível da segurança, sempre que abrimos portas estamos a "enfraquecer" o sistema pois, quantos mais serviços disponíveis, mais maneiras existem de explorar vulnerabilidades.

2.3.7 Ponto 7

Após recorrer novamente ao comando *iptables -L -v* observamos que desta vez as alterações ocorreram ao nível dos valores de *pkts* (pacotes) e *bytes*. Esta alteração de valores surge como resultado dos pedidos efetuados, nos pontos anteriores, ao servidor.

```

[root@localhost ~]# iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source            destination
 92 8615 ACCEPT     all  --  any    any     anywhere          anywhere          state RELATED,ESTABLISHED
 3 252 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
 1 44 ACCEPT     tcp  --  any    any     anywhere          anywhere          state NEW tcp dpt:ssh
 2 104 ACCEPT     tcp  --  any    any     anywhere          anywhere          state NEW tcp dpt:http
 2 104 ACCEPT     tcp  --  any    any     anywhere          anywhere          state NEW tcp dpt:ftp
1985 87340 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 108 packets, 14064 bytes)
pkts bytes target      prot opt in     out     source            destination

```

Figura 39: *iptables -L -v*

Na cadeia *INPUT*, relativamente aos serviços que ficaram ativos, na Tarefa 3, Ponto 2, é possível verificar um maior número de pacotes, os quais correspondem a pedidos por parte do cliente ao servidor para aceder aos mesmos.

Ainda na cadeia *INPUT* é de notar que o número de pacotes dos serviços que devem ser rejeitados pelo servidor é bastante considerável o que é uma consequência dos vários pedidos de serviços do *nmap* que levam a *firewall* a confirmar quais os que estão ativos.

Na cadeia *OUTPUT* há um aumento do número de pacotes que saem do servidor, ou seja, podemos constatar que o servidor responde a pedidos do cliente o que se reflete num aumento do número de pacotes.

Na cadeia *FORWARD* não há nenhuma alteração a registar.

2.4 Exercício de conclusão

Para este exercício final optamos por usar o *fwbuilder*. O “Firewall Builder” já não é mais suportado pelos criadores, mas visto que estamos a usar *CentOS 6*, achamos por bem usar uma ferramenta que é das mais intuitivas.

Firewall Builder é uma interface gráfica que ajuda a configurar o *iptables*. Estas configurações são guardadas num ficheiro que pode escalar para gerir centenas de firewalls na mesma UI.

Para instalar esta interface no *CentOS 6* acedemos <http://fwbuilder.sourceforge.net/> e corremos os seguintes códigos de maneira a compilar a ultima versão disponível:

```
# cd /fwbuilder-5.1.0.3599
# ./autogen.sh
# make
# make install
# fwbuilder
```

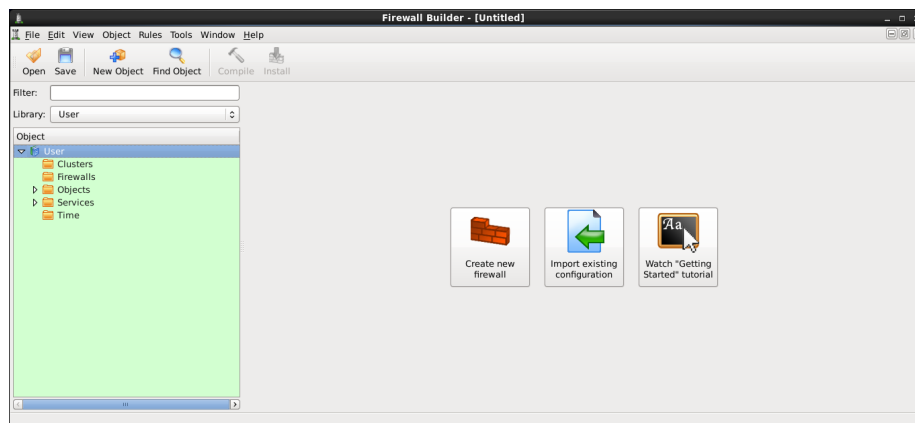


Figura 40: Verificar Instalação

Procedemos então à criação de uma nova *firewall*, fornecendo um nome e o IP da interface usada para comunicação com o exterior.

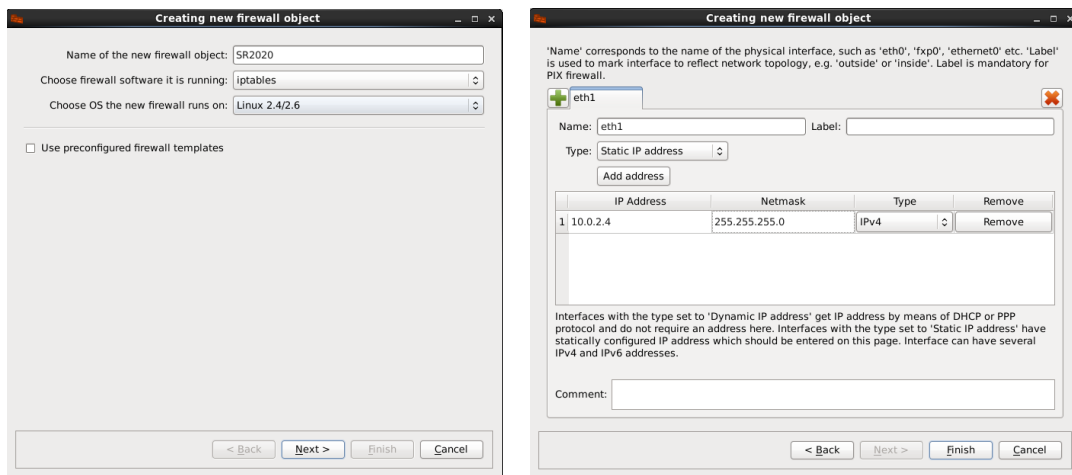


Figura 41: Criação de uma firewall

Procedendo com a configuração, criamos um objeto para filtrar a rede NAT. Para isso colocamos o endereço 10.0.2.0 com a máscara de rede /24 (255.255.255.0) para apanhar todos os IP's pertencentes a esta rede.

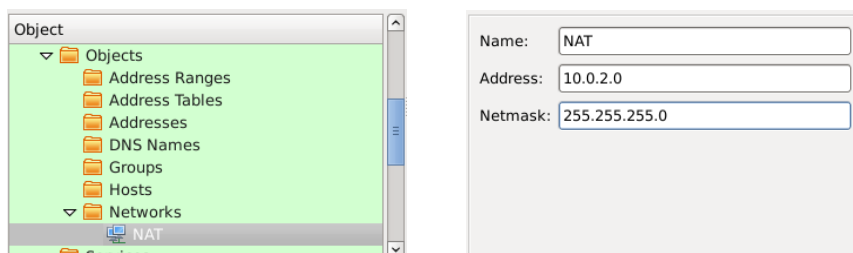


Figura 42: Objeto para a rede NAT

Quanto aos restantes objetos fomos à biblioteca standard como se pode ver na figura abaixo.

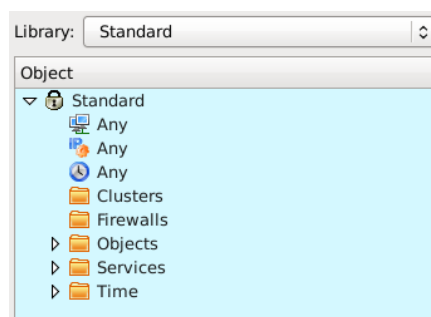


Figura 43: Objetos da biblioteca *standard*

Quanto á tabela de políticas de *firewall* optamos por seguir um pouco o que fizemos ao logo do enunciado mas, desta vez, só aceitamos o acesso ao *http*, *ftp* e *ssh* com origem na rede *NAT* anteriormente criada.

| | Source | Destination | Service | Interface | Direction | Action | Time | Options |
|---|-------------------|-------------------|-------------------|-----------|-----------|----------------------|------|---------|
| 0 | IP SR2020:eth1:ip | Any | ICMP ping reply | Any | Both | Reject:ICMP host ... | Any | log |
| 1 | Any | IP SR2020:eth1:ip | ICMP ping request | Any | Both | Reject:ICMP host ... | Any | log |
| 2 | NAT | IP SR2020:eth1:ip | TCP http | Any | Both | Accept | Any | log |
| 3 | NAT | IP SR2020:eth1:ip | TCP ftp | Any | Both | Accept | Any | log |
| 4 | NAT | IP SR2020:eth1:ip | TCP ssh | Any | Both | Accept | Any | log |
| 5 | Any | Any | Any | Any | Both | Deny | Any | log |

Figura 44: Verificar Instalação

Após proceder à instalação desta configuração executamos o comando *iptables -L -v*. Reparando que desta vez é muito mais extensa, tendo regras "desnecessárias"o que é o esperado por esta ser uma configuração gerada pelo *fwbuilder*. Repara-se também que agora na source temos o IP 10.0.2.0/24 (que é o da rede *NAT*).

```
[root@localhost osboxes]# iptables -L -v
Chain INPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
  28 2536 ACCEPT     all  --  any    any    anywhere             anywhere             state RELATED,ESTABLISHED
  0 0 RULE_0     icmp --  any    any    10.0.2.4             anywhere             icmp type 0 code 0
  0 0 RULE_1     icmp --  any    any    anywhere             10.0.2.4             icmp type 8 code 0
  1 60 RULE_2     tcp  --  any    any    10.0.2.0/24          10.0.2.4             tcp dpt:http state NEW
  0 0 RULE_3     tcp  --  any    any    10.0.2.0/24          10.0.2.4             tcp dpt:ftp state NEW
  0 0 RULE_4     tcp  --  any    any    10.0.2.0/24          10.0.2.4             tcp dpt:ssh state NEW
  0 0 RULE_5     all  --  any    any    anywhere             anywhere

Chain FORWARD (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
  0 0 ACCEPT     all  --  any    any    anywhere             anywhere             state RELATED,ESTABLISHED
  0 0 RULE_5     all  --  any    any    anywhere             anywhere

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination
  29 7479 ACCEPT     all  --  any    any    anywhere             anywhere             state RELATED,ESTABLISHED
  0 0 RULE_0     icmp --  any    any    10.0.2.4             anywhere             icmp type 0 code 0
  0 0 RULE_1     icmp --  any    any    anywhere             10.0.2.4             icmp type 8 code 0
  21 1852 RULE_5     all  --  any    any    anywhere             anywhere

Chain RULE_0 (2 references)
  pkts bytes target     prot opt in     out     source               destination
  0 0 LOG        all  --  any    any    anywhere             anywhere             LOG level info prefix 'RULE 0 -- REJECT '
  0 0 REJECT     all  --  any    any    anywhere             anywhere             reject-with icmp-host-prohibited

Chain RULE_1 (2 references)
  pkts bytes target     prot opt in     out     source               destination
  0 0 LOG        all  --  any    any    anywhere             anywhere             LOG level info prefix 'RULE 1 -- REJECT '
  0 0 REJECT     all  --  any    any    anywhere             anywhere             reject-with icmp-host-prohibited

Chain RULE_2 (1 references)
  pkts bytes target     prot opt in     out     source               destination
  1 60 LOG        all  --  any    any    anywhere             anywhere             LOG level info prefix 'RULE 2 -- ACCEPT '
  1 60 ACCEPT    all  --  any    any    anywhere             anywhere

Chain RULE_3 (1 references)
  pkts bytes target     prot opt in     out     source               destination
  0 0 LOG        all  --  any    any    anywhere             anywhere             LOG level info prefix 'RULE 3 -- ACCEPT '
  0 0 ACCEPT    all  --  any    any    anywhere             anywhere

Chain RULE_4 (1 references)
  pkts bytes target     prot opt in     out     source               destination
  0 0 LOG        all  --  any    any    anywhere             anywhere             LOG level info prefix 'RULE 4 -- ACCEPT '
  0 0 ACCEPT    all  --  any    any    anywhere             anywhere

Chain RULE_5 (3 references)
  pkts bytes target     prot opt in     out     source               destination
  21 1852 LOG        all  --  any    any    anywhere             anywhere             LOG level info prefix 'RULE 5 -- DENY '
  21 1852 DROP      all  --  any    any    anywhere             anywhere
```

Figura 45: *iptables -L -v*

De forma a fazer testes a esta nova configuração recorreremos à máquina com o *Kali*. Confirmando que na rede NAT não é possível fazer *ping*, mas os serviços *http*, *ftp* e *ssh* estão disponíveis.

```
→ ~ ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
From 10.0.2.4 icmp_seq=1 Destination Host Prohibited
From 10.0.2.4 icmp_seq=2 Destination Host Prohibited
From 10.0.2.4 icmp_seq=3 Destination Host Prohibited
^C
--- 10.0.2.4 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2004ms
```

Figura 46: *ping*

```
→ ~ sudo nmap -sS 10.0.2.4
[sudo] password for kali:
Starting Nmap 7.91 ( https://nmap.org ) at 2020-12-19 13:06 EST
Nmap scan report for 10.0.2.4
Host is up (0.0012s 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:B2:38:D7 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 5.45 seconds
```

Figura 47: *nmap*

Como ativamos os *log's* conseguimos ver a troca de informações na *firewall*. Este ficheiro encontra em */var/log/messages*, no caso do *CentOS 6*.

```
Dec 19 17:29:11 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=67 TOS=0x00 PREC=0x00 TTL=64 ID=21876 DF PROTO=UDP SPT=42976 DPT=53 LEN=47
Dec 19 17:29:11 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=67 TOS=0x00 PREC=0x00 TTL=64 ID=21876 DF PROTO=UDP SPT=53921 DPT=53 LEN=47
Dec 19 17:29:11 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=67 TOS=0x00 PREC=0x00 TTL=64 ID=21876 DF PROTO=UDP SPT=59995 DPT=53 LEN=47
Dec 19 17:29:11 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=67 TOS=0x00 PREC=0x00 TTL=64 ID=21876 DF PROTO=UDP SPT=5634 DPT=53 LEN=47
Dec 19 17:29:11 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=67 TOS=0x00 PREC=0x00 TTL=64 ID=21876 DF PROTO=UDP SPT=59227 DPT=53 LEN=47
Dec 19 17:29:11 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=67 TOS=0x00 PREC=0x00 TTL=64 ID=21876 DF PROTO=UDP SPT=59916 DPT=53 LEN=47
Dec 19 17:29:11 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=67 TOS=0x00 PREC=0x00 TTL=64 ID=21876 DF PROTO=UDP SPT=53865 DPT=53 LEN=47
Dec 19 17:29:11 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=67 TOS=0x00 PREC=0x00 TTL=64 ID=21876 DF PROTO=UDP SPT=37677 DPT=53 LEN=47
Dec 19 17:29:33 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=70 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF PROTO=UDP SPT=68 DPT=67 LEN=308
Dec 19 17:29:33 localhost dhclient[1604]: DHCPREQUEST on eth1 to 10.0.2.3 port 67 (xid=0x37f13351)
Dec 19 17:29:33 localhost dhclient[1604]: send packet: operation not permitted
Dec 19 17:29:33 localhost dhclient[1604]: dhclient.c.2609: Failed to send 300 byte long packet over fallback interface.
Dec 19 17:29:33 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=70 TOS=0x00 PREC=0x00 TTL=64 ID=43576 DF PROTO=UDP SPT=56278 DPT=53 LEN=50
Dec 19 17:29:33 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=70 TOS=0x00 PREC=0x00 TTL=64 ID=43576 DF PROTO=UDP SPT=43775 DPT=53 LEN=50
Dec 19 17:29:33 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=74 TOS=0x00 PREC=0x00 TTL=64 ID=43576 DF PROTO=UDP SPT=57907 DPT=53 LEN=54
Dec 19 17:29:33 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=74 TOS=0x00 PREC=0x00 TTL=64 ID=43576 DF PROTO=UDP SPT=38788 DPT=53 LEN=54
Dec 19 17:29:33 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=70 TOS=0x00 PREC=0x00 TTL=64 ID=43620 DF PROTO=UDP SPT=46459 DPT=53 LEN=50
Dec 19 17:29:33 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=70 TOS=0x00 PREC=0x00 TTL=64 ID=43620 DF PROTO=UDP SPT=38288 DPT=53 LEN=50
Dec 19 17:29:33 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=74 TOS=0x00 PREC=0x00 TTL=64 ID=43620 DF PROTO=UDP SPT=39407 DPT=53 LEN=54
Dec 19 17:29:33 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=74 TOS=0x00 PREC=0x00 TTL=64 ID=43620 DF PROTO=UDP SPT=39833 DPT=53 LEN=54
Dec 19 17:29:33 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=70 TOS=0x00 PREC=0x00 TTL=64 ID=43620 DF PROTO=UDP SPT=54842 DPT=53 LEN=50
Dec 19 17:29:33 localhost kernel: RULE 5 -- DENY IN= OUT=eth1 SRC=10.0.2.4 DST=192.168.1.1 LEN=70 TOS=0x00 PREC=0x00 TTL=64 ID=43620 DF PROTO=UDP SPT=58885 DPT=53 LEN=50
```

Figura 48: Excerto do ficheiro de *log*

3 Conclusão

No início o grupo deparou-se com algumas dificuldades na obtenção de todos os recursos necessários, como por exemplo uma máquina virtual *CentOS* com uma versão mais antiga que nos permitisse estar munidos de todas as ferramentas necessárias para o trabalho, mas após esse percalço ultrapassado podemos afirmar que o projeto correu bem e sem grandes dificuldades.

Uma das questões que começou a surgir no ar no decorrer do projeto foi: **Qual é o futuro das *firewalls*?**

Se pensarmos no papel da *firewall* desde que foi criada, esta foi colocada como um *gatekeeper* na periferia da rede. Ela age como um ponto de controle abrangente, que inspeciona o tráfego da rede enquanto este viaja nesse perímetro. Colocado no ponto de entrada/saída da rede, a *firewall* é responsável pela validação das comunicações: o tráfego de rede interna é considerado confiável e o tráfego externo é considerado não confiável. Foram criadas e aplicadas um conjunto de regras e políticas neste ponto único de controle para garantir que o tráfego desejado fosse permitido dentro e fora da rede e o tráfego indesejável fosse evitado.

Comparando o perímetro da rede a um fosso ao redor de um castelo, a *firewall* age como uma ponte levadiça que controla todo o tráfego de entrada e saída do castelo.

Não demorou muito para que esta prática de reforçar a segurança por meio de um único ponto de controle fosse desafiada. Primeiro, houve o aumento do acesso remoto e da mobilidade empresarial. Mas a transformação realmente começou com a *cloud computing*. Quando as empresas mudaram para a nuvem, os dispositivos e utilizadores começaram a migrar em massa para fora da rede interna controlada, o que tornou o modelo de ponto de controle único ineficaz pois passaram a existir vários perímetros e todos eles precisavam ser protegidos. Não havia um modo eficaz de colocar um fosso em redor da rede.

É o momento de a *firewall* assumir o seu lugar como a base para uma plataforma de segurança de rede ágil e integrada.

No entanto, os requisitos da *firewall* aumentaram significativamente para proteger a ampla gama de infraestruturas de rede, dispositivos conectados e sistemas operacionais de ameaças avançadas. Consequentemente, nossos dispositivos de *firewall* “tradicionais” estão a ser aumentados por uma mistura de dispositivos físicos e virtuais - alguns são incorporados na rede, enquanto outros são fornecidos como um serviço, são baseados em host ou estão incluídos em ambientes de nuvem pública. Alguns estão a assumir novas formas, como dispositivos em *cluster* que se adaptam a grandes requisitos de tráfego, *software* que corre em dispositivos pessoais, *routers SD-WAN* e *gateways* de Internet seguros. A atividade de compartilhar inteligência sobre ameaças em todos esses dispositivos de *firewall* distintos, independentemente de sua localização, é vital para a visibilidade uniforme das ameaças e uma postura de segurança forte.

Para fazer a mudança completa e proteger melhor as redes de hoje, as empresas devem afastar-se da abordagem tradicional de "perímetro". Em vez disso, elas precisam estabelecer pontos de aplicação estratégicos em toda a rede, mais perto das informações ou aplicativos que precisam ser protegidos. Especificamente, a criação de microperímetros em pontos de controle físicos e lógicos tornou-se uma realidade necessária segundo a Cisco [1].

Precisamos pensar menos sobre a firewall como um dispositivo de rede física autônomo e mais sobre a funcionalidade da firewall.

Webgrafia

- [1] Cisco. *The Future of the Firewall White Paper*. URL: <https://www.cisco.com/c/en/us/products/collateral/security/firewalls/ngfw-futureoffirewalling-wp.html>.