Netfilter/iptables

Sumario

1 Netfilter

Netfilter é un firewall de estado presente nas versións 2.4/2.6/3/4 do núcleo de Linux e Iptables é a aplicación que permite configurar Netfilter. Con elas pódese facer, entre outras:

- Filtrado de paquetes nas capas 2 e 3 da pila TCP/IP (packet filtering).
- Seguimento de conexións (connection tracking).
- Tradución de direccións e portos (NAT e NAPT).
- Manipulación das cabeceiras dos paquetes.

Esto permite que con netfilter se poidan montar:

- Firewalls con filtrado de paquetes con ou sen seguimento de conexións.
- Firewalls de host e de rede.
- Routers-firewalls con NAT.
- Proxys transparentes usando NAT para redirixir as peticións dos clientes.
- Sistemas de encamiñamento con calidade de servizo (QoS).
- etc.

Netfilter está presente tanto en distribucións Linux de carácter xeral, como Debian, Ubuntu, Red Hat coma en distribucións específicas de seguridade coma Endian Firewall, BrazilFW e ZeroShell.

Para poder traballar con Netfilter é necesario coñecer a súa estrutura interna; é dicir, os elementos que a compoñen e a súa organización. Hai tres conceptos básicos que se estudarán:

- Regras (rules).
- Cadeas (chains).
- Táboas (tables).

Fig. Exemplos de distribucións Linux con netfilter/iptables

1.1 Regras (rules)

Cada regra de Iptables é unha liña que Netfilter comproba para saber que facer cun paquete. Temos que ver as regras en netfilter/iptables como liñas formadas por condicións e unha acción. Se todas as condicións que contén a regra se cumpren, Netfilter executa a acción especificada sobre o paquete en cuestión.

- **Condición (*match*)**: indica a Netfilter que requisitos ten cumprir un paquete para casar cunha regra (dirección ip, protocolo, etc.). Por exemplo:
  - `-p udp --dport 53` → indica que o paquete a nivel de transporte debe usar o protocolo é UDP e ir dirixido ó porto destino o 53.
  - `-i eth0 -s 192.168.0.0/24` → indica que o paquete debe ter orixe un equipo da rede 192.168.0.0/24 e ademais debe recibirse no equipo pola interface `eth0`.
- **Acción (*target*)**: indica a Netfilter qué facer co paquete se cumpre todas as condicións da regra. Por exemplo, entre as accións máis importantes atópanse:
  - `ACCEPT`: o paquete acéptase.
  - `DROP`: o paquete descártase de forma silenciosa.
  - `REJECT`: o paquete descártase informando ó emisor que o seu paquete foi descartado.
  - `LOG`: rexístrase nos logs información sobre o paquete.
  - `DNAT`: fai NAT modificando a dirección (e/ou porto) destino do paquete.
  - `SNAT`: fai NAT modificando a dirección (e/ou porto) orixe do paquete.
  - `REDIRECT`: redirección de portos na mesma máquina.
  - `MARK`: marcaxe de paquetes para ser usados por sistemas de colas (QoS).
  - Pódese non indicar ningunha acción na regra. Neste caso actualízanse os contadores de paquetes e bytes da regra e inmediatamente despois pásase á seguinte regra da cadea.

Ó longo deste tema estudaremos con exemplos prácticos como crear regras especificando múltiples condicións a cumprir polos paquetes e indicando a acción a realizar. A modo de exemplo amósase un conxunto de regras netfilter/iptables nun equipo:

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1     1913  143K ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
2        0     0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0           state NEW
3        1    60 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:22 state NEW

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

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1     1528  174K ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
2        1    60 ACCEPT     all  --  *      lo      0.0.0.0/0            0.0.0.0/0           state NEW
3        2   116 ACCEPT     udp  --  *      *       0.0.0.0/0            8.8.8.8             udp dpt:53 state NEW
4        0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            8.8.4.4             udp dpt:53 state NEW
5        0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           multiport dports 80,443 state NEW
```

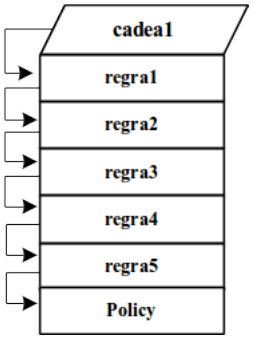
1.2 Cadeas (chains)

Unha **cadea** é unha **lista ordenada de regras**. Para cada paquete vaise comprobando se lle é de aplicación cada regra da cadea; é dicir, se se cumpre a condición da regra:

- Se unha regra non se lle aplica a un paquete, pásase á seguinte regra da cadea.
- Se unha regra si se lle aplica a un paquete, executase a acción definida na devandita regra e; salvo excepcións, xa non se comprobarán máis regras da cadea.

Na seguinte imaxe pode verse como é o procesamento dun paquete ó longo dunha cadea: regra a regra ata cumprir as condicións dunha.

```
Chain INPUT (policy DROP)
num target prot opt source destination
```




```
1 ACCEPT udp -- 8.8.8.8 0.0.0.0/0 udp spt:53
2 ACCEPT udp -- 8.8.4.4 0.0.0.0/0 udp spt:53
3 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22
4 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp spt:80
5 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp spt:443
```

Fig. Fluxo do procesamento dun paquete

Unha cadea pode ter definida unha **política** (***policy***), que ven sendo a **acción por defecto para a cadea**. Cando para un paquete non casa con ningunha das regras da cadea, executase para él a política da cadea. No exemplo da figura anterior, a política da cadea chamada INPUT é DROP; é dicir, se un paquete atravesa as 5 regras é descartado de xeito silencioso.

Existen dous tipos de cadeas:

- **Predefinidas**: PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING.
- **Definidas polo usuario**.

Cadeas Predefinidas

A un paquete ó chegar a unha maquina, aplícanselle as regras das cadeas predeterminadas en distintos momentos segundo o seguinte esquema:

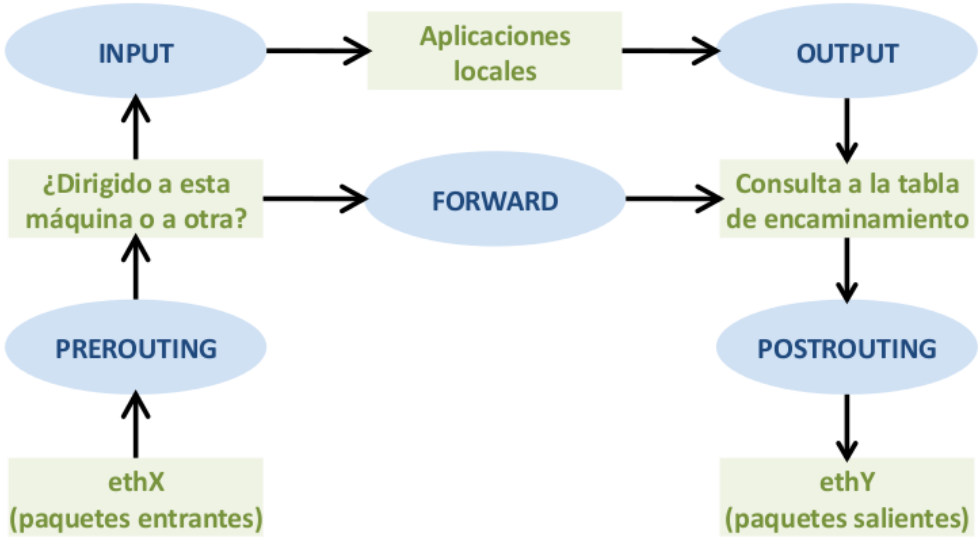


Fig. Cadeas predeterminadas en netfilter. GSyC (CC BY-SA 2.1 ES)

Para entendernos, **as cadeas poden interpretarse coma puntos onde o sistema de filtrado pode manipular os paquetes**:

- Cadea **PREROUTING**: ten as regras que se aplican ós paquetes que chegan á máquina. Esta cadea execútase antes de comprobar se o paquete é para a propia maquina ou hai que reenvialo hacia outra.
- Cadea **INPUT**: ten as regras que se aplican ós paquetes destinados á propia máquina (a aplicacións correndo na propia máquina). Esta cadea execútase xusto antes de entregalos á aplicación local.
- Cadea **FORWARD**: ten as regras que se aplican ós paquetes que chegaron á máquina pero van destinados a outra e hai que reenvialos. Esta cadea execútase antes de consultar a táboa de encamiñamento.
- Cadea **OUTPUT**: ten as regras que se aplican ós paquetes creados pola propia máquina (por aplicacións correndo na propia máquina). Esta cadea execútase xusto despois de que a aplicación lle pase os datos a enviar ó kernel do sistema operativo e antes de consultar a táboa de encamiñamento.
- Cadea **POSTROUTING**: ten as regras que se aplican ós paquetes que saen da máquina, tanto os creados por ela coma os que se reenvían. Esta cadea execútase despois de consultar a táboa de encamiñamento.

Únicamente as accións ACCEPT e DROP poden ser usadas como política nas cadeas predefinidas; sendo ACCEPT, a acción por defecto.

Cadeas definidas polo usuario

Se un paquete está sendo procesado nunha cadea, por exemplo OUTPUT, é posible indicar unha acción nunha regra para facer saltar ó paquete a unha cadea diferente. O paquete será procesado na nova cadea; regra por regra, ata que cumpra unha delas ou chegue ó final e volva a cadea orixinal. É moi recomendable dividir conxuntos de regras complexas en cadeas separadas para facilitar a creación e xestión do firewall. Un exemplo de uso das cadeas de usuario sería unha cadea que conteña as regras para controlar o tráfico de administración remota do equipo.

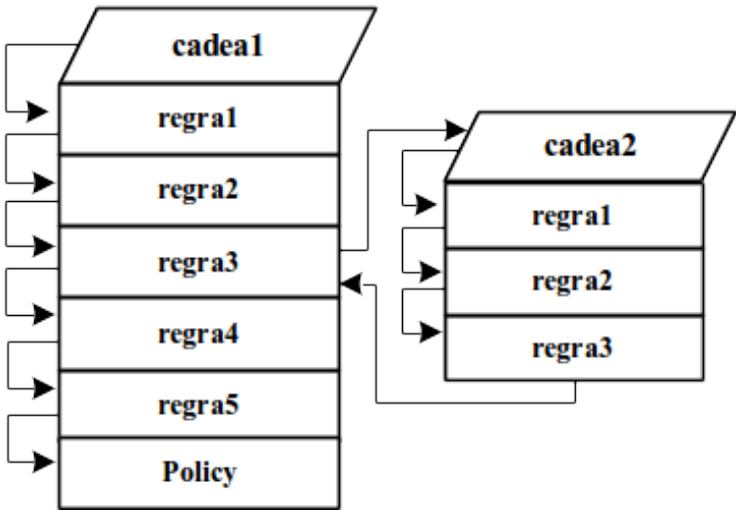


Fig. Fluxo do procesamento con salto a unha cadea definida polo usuario

Todas as cadeas de usuario teñen como política a acción RETURN que devolve o paquete á cadea orixe. Esta política non pode ser cambiada; sen embargo, nada nos impide crear e engadir regras á cadea para establecer accións para todos os paquetes que atravesen a cadea; é dicir, estariamos creando nós unha politica a man. Por exemplo, na Cadea#2 da figura anterior a regra nº 3 podería ser unha regra que descarte todos os paquetes con DROP; é dicir, todos os paquetes derivados á Cadea#2 e que non sexan explicítamente aceptados polas regras #1 e #2, serán descartados sen voltar á Cadea#1 (é dicir, creamos unha política por defecto a man para a Cadea#2).

1.3 Táboas (tables)

Unha táboa de iptables contén un conxunto de cadeas, tanto predefinidas coma de usuario. Unha táboa concreta engloba as regras (agrupadas en cadeas) relacionadas cun **tipo de procesamento** dos paquetes. Netfilter define as seguintes táboas:

- **filter**: engloba as regras de filtrado de paquetes; é dicir, as que deciden que un paquete continúe o seu camiño ou sexa descartado.
- **nat**: engloba as regras de modificación de direccións IP e portos (NAT e NAPT).
- **mangle**: engloba as regras de modificación dalgúns campos das cabeceiras dos paquetes (TTL, ToS) e ademais permite marcalos para que outros programas poidan recoñecelos (p. ex. para proporcionar QoS).
- **raw:** engloba as regras que permiten marcar excepcións ó seguimento que fai o kernel das conexións da máquina das cabeceiras do paquete.

Na seguinte táboa pode verse a relación entre cadeas e táboas en netfilter:

Táboas e cadeas	
A táboa filter inclúe as cadeas: • FORWARD • INPUT • OUTPUT	A táboa nat inclúe as cadeas: • PREROUTING • OUTPUT • POSTROUTING
A táboa mangle inclúe as cadeas: • PREROUTING • FORWARD • INPUT • OUTPUT • POSTROUTING	A táboa raw inclúe as cadeas: • PREROUTING • OUTPUT

Fig. Relación entre cadeas e táboas en netfilter/iptables

Na seguinte imaxe pode verse a relación entre cadeas e táboas en netfilter. Ademais, a imaxe permite ter unha visión global da ruta dos paquetes e das posibilidades de procesamento que proporciona netfilter en cada momento. Como xa se comentou, netfilter permite traballar cos paquetes en diferentes puntos; por exemplo, sobre un paquete recén chegado (PREROUTING) pode aplicarse NAT e cambiar a dirección IP destino para reenvialo a outro equipo.

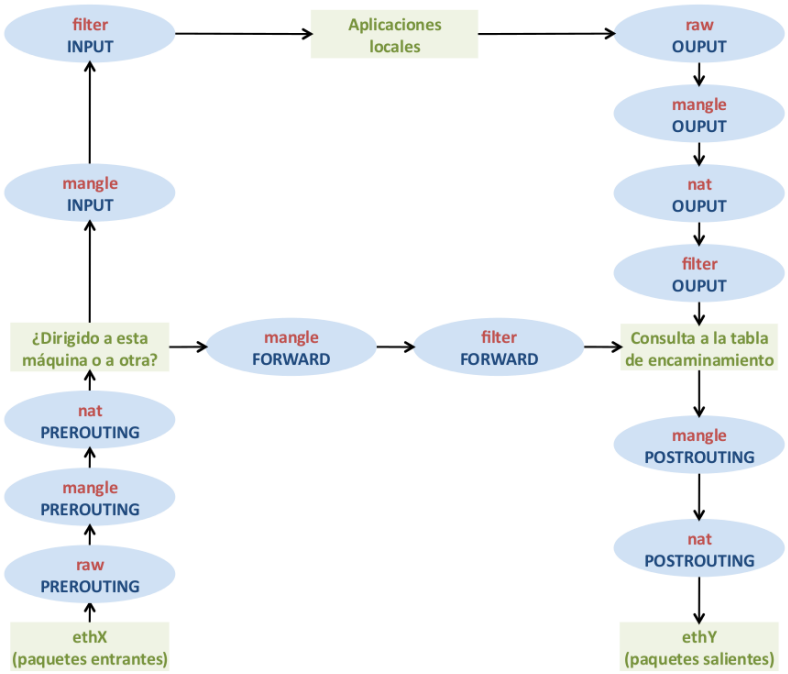


Fig. Táboas e cadeas predeterminadas en netfilter. GSyC (CC BY-SA 2.1 ES)

Aínda que a imaxe anterior da unha visión completa do esquema de funcionamento de netfilter, adóitase a traballar cunha versión máis simplificada que deixa de lado os procesamentos máis avanzados (raw e mangle). É moi interesante ter a man algunha destas imaxes cando se está comezando a traballar con netfilter/iptables para asociar a sintaxe dos comandos iptables coa secuencia de procesamento de netfilter:

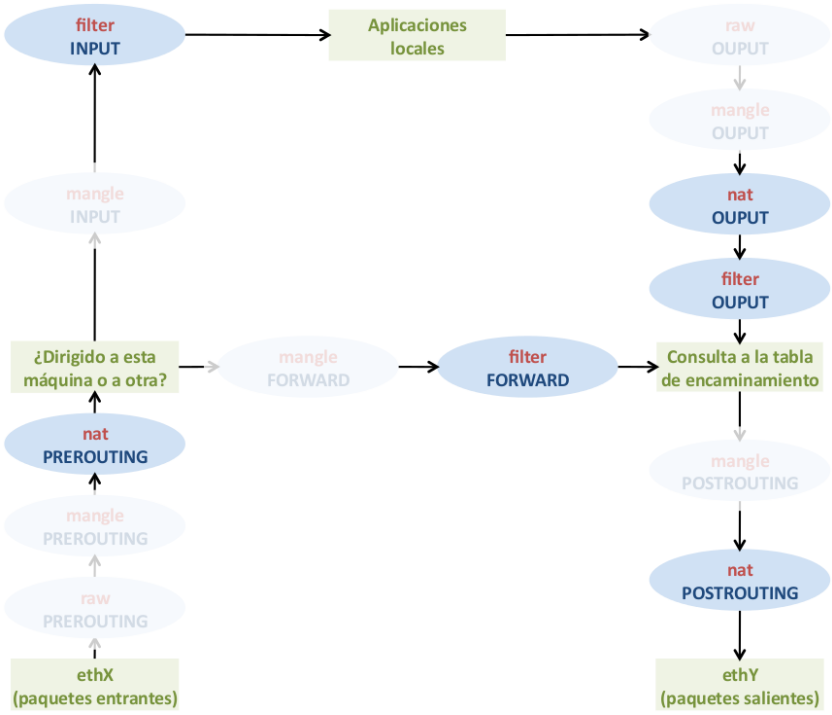


Fig. Táboas e cadeas predeterminadas máis usadas en netfilter. GSyC (CC BY-SA 2.1 ES)

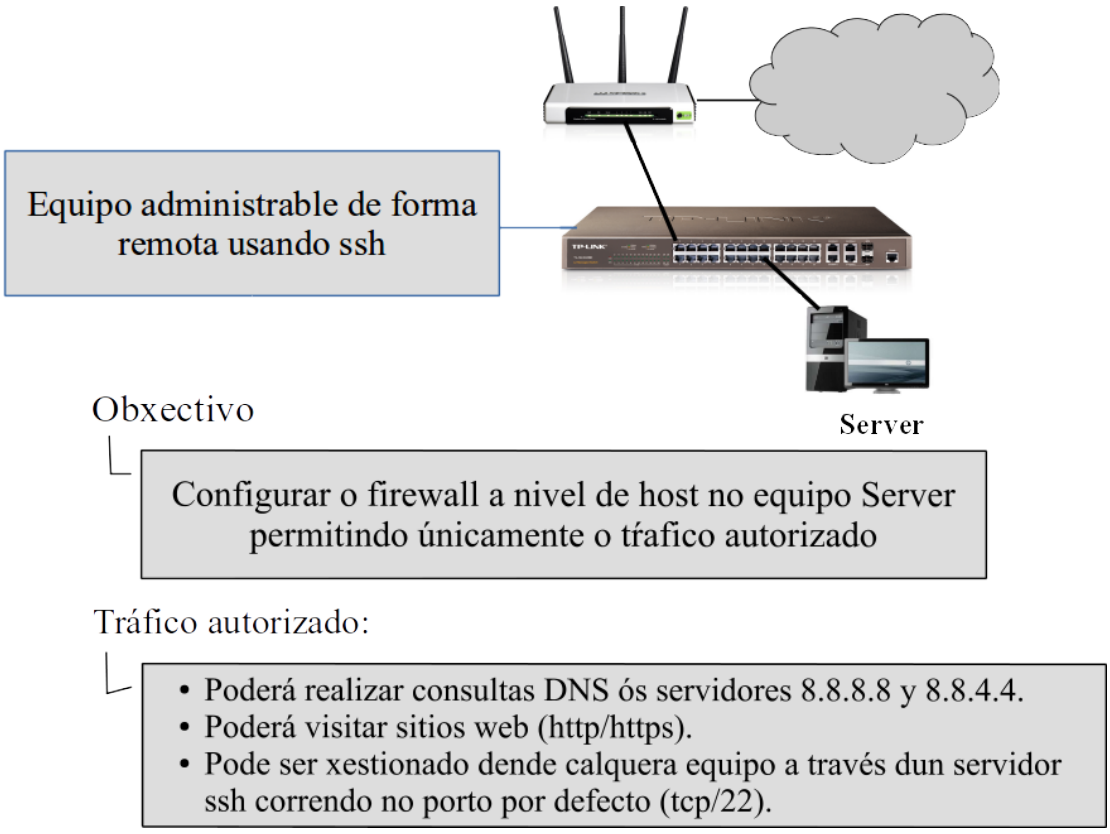
NOTAS

Non é habitual o seu uso e normalmente non aparece na documentación sobre netfilter/iptables pero existen:

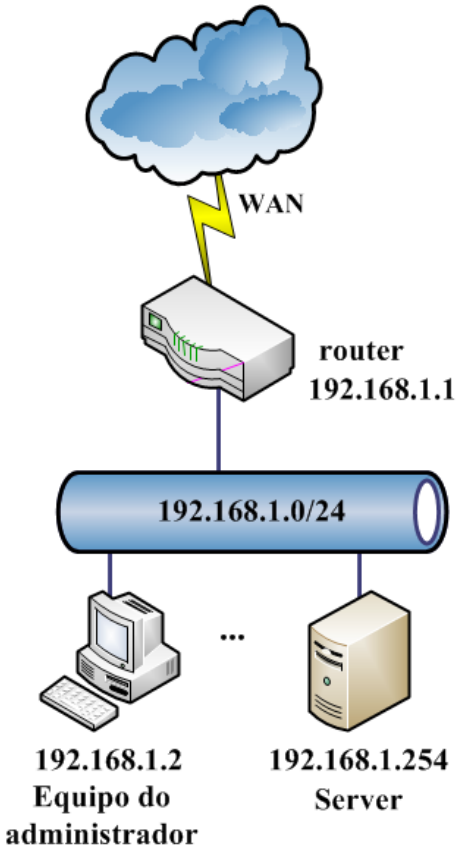
- Dende o ano 2010 a táboa nat inclúe unha cadea INPUT onde se pode facer SNAT.
- Existe unha quinta táboa chamada security usada para que módulos de seguridad como SELinux implementen controis de seguridade MAC (Mandatory Access Control).

2 Escenario 2A: Firewall de host con netfilter/iptables

Nesta práctica configurarase un firewall de host nunha máquina Linux Ubuntu Server. Netfilter configurarase usando comandos iptables para permitir únicamente o tráfico autorizado, denegando por defecto o resto do tráfico.



2.1 Configuración do escenario



Configuración do server para o escenario 2A:

En Virtualbox

No caso de empregar Virtualbox para realizar o escenario 2A, o server corre nunha máquina virtual coas seguintes características:

- Sistema Operativo Ubuntu Server 14.04 (64 bits).
- Memoria RAM: 512 MBytes
- Rede: un único adaptador de rede configurado en modo ponte (bridge).

Ó estar en modo bridge o server terá unha configuración válida para a rede local. No exemplo, a configuración escollida é:

- Interface de rede: eth0
- IP do server: 192.168.1.254/255.255.255.0
- Default gateway: 192.168.1.1
- Servidores DNS: 8.8.8.8 e 8.8.4.4

Para configurar o server hai que editar o arquivo `/etc/network/interfaces`.

```
uadmin@server:~$ sudo nano /etc/network/interfaces
```

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.1.254
        netmask 255.255.255.0
        gateway 192.168.1.1
        dns-nameservers 8.8.8.8 8.8.4.4
```

Unha vez modificado o arquivo hai que aplicar esta configuración. En Ubuntu Server 14.04 pódese reiniciar a máquina ou aplicar os cambios sen reiniciar co comando:

```
uadmin@server:~$  sudo sh -c "ifdown eth0 && ifup eth0"
```

Pode verificarse que a nova configuración foi aplicada:

```
uadmin@server:~$ ifconfig -a ; netstat -nr
eth0 Link encap:Ethernet direcciónHW 08:00:27:c1:56:ad
 Direc. inet:192.168.1.254 Difus.:192.168.1.255 Másc:255.255.255.0
 Dirección inet6: fe80::a00:27ff:fec1:56ad/64 Alcance:Enlace
 ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Métrica:1
 Paquetes RX:5297 errores:0 perdidos:0 overruns:0 frame:0
 Paquetes TX:3176 errores:0 perdidos:0 overruns:0 carrier:0
 colisiones:0 long.colaTX:1000
 Bytes RX:6383905 (6.3 MB) TX bytes:265264 (265.2 KB)
lo Link encap:Bucle local
 Direc. inet:127.0.0.1 Másc:255.0.0.0
 Dirección inet6: ::1/128 Alcance:Anfitrión
 ACTIVO BUCLE FUNCIONANDO MTU:65536 Métrica:1
 Paquetes RX:32 errores:0 perdidos:0 overruns:0 frame:0
 Paquetes TX:32 errores:0 perdidos:0 overruns:0 carrier:0
 colisiones:0 long.colaTX:0
 Bytes RX:2480 (2.4 KB) TX bytes:2480 (2.4 KB)


Tabla de rutas IP del núcleo
Destino Pasarela Genmask Indic MSS Ventana irtt Interfaz
```

```
0.0.0.0 192.168.1.1 0.0.0.0 UG 0 0 0 eth0
192.168.1.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
uadmin@server:~$
```

Para instalar o servidor openssh e verificar que o servizo está levantado execútanse os seguintes comandos:

```
uadmin@server:~$ sudo apt-get update
uadmin@server:~$ sudo apt-get install openssh-server
uadmin@server:~$ sudo netstat -putan
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local Dirección remota Estado PID/Program name
tcp 0 0 0.0.0.0:22 0.0.0.0:* ESCUCHAR 1959/sshd
```

É recomendable facer unha actualización do sistema usando os comandos:

```
uadmin@server:~$ sudo apt-get update
uadmin@server:~$ sudo apt-get dist-upgrade
```

Apuntes para Ubuntu Server 16.04

En Ubuntu Server 16.04 fixéronse cambios que afectan ó modo de nomear as interfaces de rede, que xa non reciben os clásicos nomes de eth0, eth1, wlan0,... O contido do arquivo `/etc/network/interfaces` é semellante ó visto, pero tendo en conta o nome asignado polo sistema á tarxeta de rede. No meu caso a interface chámase enp0s3:

```
uadmin@server16:~$ sudo nano /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
source /etc/network/interfaces.d/*
# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
auto enp0s3
iface enp0s3 inet static
 address 192.168.1.254
 netmask 255.255.255.0
 gateway 192.168.1.1
 dns-nameservers 8.8.8.8 8.8.4.4
```

Hai erros reportados á hora de reiniciar a rede usando o procedemento visto para Ubuntu 14.04 con

ifdow && ifup (tampocuo funciona o método alternativo con service networking). O motivo destes erros parece estar en que cando se configura unha nova IP na interface e procédese a reiniciar a rede, non se borra a IP vella. Polo tanto, para que se colla a nova configuración de rede podemos proceder a reiniciar a máquina ou executar os seguintes comandos:

```
uadmin@server16:~$ sudo ip addr flush enp0s3
uadmin@server16:~$ sudo systemctl restart networking.service
uadmin@ubuntu16:~$ ifconfig -a ; netstat -nr
enp0s3 Link encap:Ethernet HWaddr 08:00:27:a7:60:1f
 inet addr:192.168.1.254 Bcast:192.168.1.255 Mask:255.255.255.0
 inet6 addr: fe80::a00:27ff:fea7:601f/64 Scope:Link
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:59318 errors:0 dropped:0 overruns:0 frame:0
 TX packets:34464 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:78782076 (78.7 MB) TX bytes:2521167 (2.5 MB)
lo Link encap:Local Loopback
 inet addr:127.0.0.1 Mask:255.0.0.0
 inet6 addr: ::1/128 Scope:Host
 UP LOOPBACK RUNNING MTU:65536 Metric:1
 RX packets:160 errors:0 dropped:0 overruns:0 frame:0
 TX packets:160 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1
 RX bytes:11840 (11.8 KB) TX bytes:11840 (11.8 KB)
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 192.168.1.1 0.0.0.0 UG 0 0 0 enp0s3
192.168.1.0 0.0.0.0 255.255.255.0 U 0 0 0 enp0s3
uadmin@ubuntu16:~$
```

Ollo ó facer o flush se estamos usando conexións remotas para administrar o equipo; xa que, se perde a conexión dun xeito inmediato co servidor.

2.2 Estado inicial

Co server correctamente configurado, pode comprobarse que é posible enviar e recibir paquetes sen limitacións. A modo de exemplo, fanse probas ping, navegación web, acceso ftp e resolucións dns para verificar que é posible iniciar conexións dende o server hacia fóra e recibir as respostas.

```
uadmin@server:~$ ping -c 2 www.google.com
PING www.google.com (216.58.211.68) 56(84) bytes of data.
64 bytes from par03s14-in-f4.1e100.net (216.58.211.68): icmp_seq=1 ttl=54
time=75.5 ms
64 bytes from par03s14-in-f4.1e100.net (216.58.211.68): icmp_seq=2 ttl=54
time=75.4 ms
--- www.google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 75.440/75.500/75.560/0.060 ms

uadmin@server:~$ wget http://www.xunta.es
--2015-11-03 18:07:11-- http://www.xunta.es/
Resolviendo www.xunta.es (www.xunta.es)... 85.91.64.254
Conectando con www.xunta.es (www.xunta.es)[85.91.64.254]:80... conectado.
Petición HTTP enviada, esperando respuesta... 301 Moved Permanently
Ubicación: /portada [siguiente]
--2015-11-03 18:07:25-- http://www.xunta.es/portada
Reutilizando la conexión con www.xunta.es:80.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 122388 (120K) [text/html]
Grabando a: “index.html”
100%[================>] 122.388 153KB/s en 0,8s
2015-11-03 18:07:26 (153 KB/s) - “index.html” guardado [122388/122388]

uadmin@server:~$ ftp ftp.rediris.es
Connected to zeppo.rediris.es.
220- Bienvenido al FTP anónimo de RedIRIS.
220-Welcome to the RedIRIS anonymous FTP server.
220 Only anonymous FTP is allowed here
Name (ftp.rediris.es:uadmin): anonymous
331- RedIRIS - Red Académica y de Investigación Española
331- RedIRIS - Spanish National Research Network
331-
331- ftp://ftp.rediris.es -=- http://ftp.rediris.es
331-
```

```
331 Any password will work
Password:
230 Any password will work
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>exit
```

```
uadmin@server:~$ host www.google.com
www.google.com has address 173.194.45.83
www.google.com has address 173.194.45.84
www.google.com has address 173.194.45.82
www.google.com has address 173.194.45.81
www.google.com has address 173.194.45.80
www.google.com has IPv6 address 2a00:1450:4007:806::1012
uadmin@server:~$
```

Tameń é posible iniciar conexións dende outros equipos hacia o server. A modo de exemplo faise ping e accédese por ssh dende o equipo do administrador ó server:

```
manuel@lubuntu:~$ ping -c2 192.168.1.254
PING 192.168.1.254 (192.168.1.254) 56(84) bytes of data.
64 bytes from 192.168.1.254: icmp_seq=1 ttl=255 time=0.483 ms
64 bytes from 192.168.1.254: icmp_seq=2 ttl=255 time=0.496 ms
--- 192.168.1.254 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.483/0.489/0.496/0.023 ms
manuel@lubuntu:~$
```

```
manuel@lubuntu:~$ ssh uadmin@192.168.1.254
uadmin@192.168.1.254's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-30-generic x86_64)
* Documentation: https://help.ubuntu.com/
 System information as of Tue Nov 3 17:56:25 CET 2015
 System load: 0.0 Processes: 71
 Usage of /: 7.7% of 19.07GB Users logged in: 1
 Memory usage: 16% IP address for eth0: 192.168.1.254
 Swap usage: 0%
 Graph this data and manage this system at:
 https://landscape.canonical.com/
Last login: Tue Nov 3 17:56:25 2015 from 192.168.1.2
```

```
uadmin@server:~$
```

O anterior é posible xa que **a configuración por defecto de netfilter en Ubuntu Server é permitir todo o tráfico**, tanto entrante como saínte. Esta configuración de permitir todo por defecto pode verificarse facendo un listado das regras da táboa filter:

```
uadmin@server:~$ sudo iptables -t filter -L
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
uadmin@server:~$
```

Como pode verse, non hai regras en ningunha das cadeas (INPUT, FORWARD e OUTPUT) da táboa filter; polo que, a un paquete procesado nestas cadeas aplicaráselle a **Policy** (acción por defecto da cadea), que é ACCEPT en todas elas.

A opción `-t` do comando iptables permite especificar a táboa na que se está interesado; e no caso de non especificar nada, cóllese a táboa filter por defecto. Polo tanto, o mesmo resultado obtense ó executar:

```
uadmin@server:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
uadmin@server:~$
```

2.3 Configurando netfilter

CleanUp Rules

A opción de '**permitir por defecto**' é a máis cómoda e vén activada por defecto en moitos firewalls e routers, pero non é a máis segura. Obriga ó administrador a crear regras para bloquear o tráfico non desexado; e no caso de esquecer a creación da regra correspondente para bloquear un tipo de tráfico, este tráfico non desexado atravesará o firewall.

A política de '**denegar por defecto**', onde permítese unicamente o tráfico requirido polas necesidades da rede e rexeitase o resto é máis segura. Todo tráfico non autorizado nas regras será eliminado; polo que, para permitilo habería que contactar co administrador do firewall e que éste procedese a autorizalo. As regras de bloquear todo por defecto coñécense como **CleanUp Rules**.

NOTA:

Se estamos accedendo por ssh para configurar o firewall da máquina, non crear as reglas de CleanUp neste momento; xa que, perderíase o acceso por ssh de xeito inmediato. Despois de crear as regras que permiten as conexións ssh ó equipo xa se poden crear as CleanUp Rules.

```
uadmin@server:~$ sudo iptables -P INPUT DROP
uadmin@server:~$ sudo iptables -P OUTPUT DROP
uadmin@server:~$ sudo iptables -P FORWARD DROP
uadmin@server:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target prot opt source destination
Chain FORWARD (policy DROP)
target prot opt source destination
Chain OUTPUT (policy DROP)
target prot opt source destination
uadmin@server:~$
```

Agora, todo o tráfico do server entrante/saínte é bloqueado por netfilter. Comunicacións que antes eran permitidas agora son bloqueadas:

```
uadmin@server:~$ host www.google.com
;; connection timed out; no servers could be reached
uadmin@server:~$ ping -c 4 www.google.com
ping: unknown host www.google.com

uadmin@server:~$ ping -c 4 8.8.8.8
```

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
```

Resolución DNS

Para permitir a resolución DNS hai que:

- Permitir envío de consultas dns: permitir que saian do server os paquetes con destino ó porto 53/udp dos servidores autorizados.
- Permitir entradas das respostas dns: aceptar os paquetes con destino o server procedentes dos servidores autorizados procedentes do porto 53/udp.

A estrutura básica dunha regra iptables é:

`iptables [-t táboa] comando [condición] [-j acción]`

- `-t` → permite especificar a táboa na que actuar (filter por defecto)
- `comando` → qué facer coa regra (engadir, insertar, borrar, …)
- `condición` *(matches)* → requisitos que debe cumprir o paquete (protocolo, ip orixe/destino, porto orixe/destino, interface, …)
- `acción` *(targets)* → que facer co paquete se cumpre coas condicións (aceptar, descarte silencioso/informado, nat, redirección, rexistro, ...)

Tendo en conta a estrutura dunha regra iptables, os comandos iptables para permitir o envío de consultas dns son:

```
uadmin@server:~$ sudo iptables -A OUTPUT -p udp --dport 53 -d 8.8.8.8 -j ACCEPT
uadmin@server:~$ sudo iptables -A OUTPUT -p udp --dport 53 -d 8.8.4.4 -j ACCEPT
```

Fixarse que estas regras hai que engadilas na cadea OUTPUT da táboa filter; xa que, na cadea OUTPUT é onde están as regras que se aplican ós paquetes creados pola propia máquina.

Para permitir a entrada das respostas hai que engadir as regras correspondentes na cadea INPUT; xa que, nesta cadea están as regras que se aplicanós paquetes destinados á propia máquina:

```
uadmin@server:~$ sudo iptables -A INPUT -p udp --sport 53 -s 8.8.8.8 -j ACCEPT
uadmin@server:~$ sudo iptables -A INPUT -p udp --sport 53 -s 8.8.4.4 -j ACCEPT
```

A opción `-A` empregada nas regras engade a regra ó final da cadea. No caso de querer introducir unha regra nunha posición concreta habería que usar a opción `-I`

```
uadmin@server:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target prot opt source destination
ACCEPT udp -- 8.8.8.8 0.0.0.0/0 udp spt:53
```

```
ACCEPT udp -- 8.8.4.4 0.0.0.0/0 udp spt:53
Chain FORWARD (policy DROP)
target prot opt source destination
Chain OUTPUT (policy DROP)
target prot opt source destination
ACCEPT udp -- 0.0.0.0/0 8.8.8.8 udp dpt:53
ACCEPT udp -- 0.0.0.0/0 8.8.4.4 udp dpt:53
```

Agora dende server é posible facer resolucións DNS:

```
uadmin@server:~$ host www.xunta.es
Using domain server:
Name: 8.8.8.8
Address: 8.8.8.8#53
Aliases:
www.xunta.es has address 85.91.64.254
```

Hai que ter en conta que as resolucións dns están autorizadas unicamente contra os servidores DNS 8.8.8.8 e 8.8.4.4 e para o protocolo udp e o porto 53. Facer unha resolución dns contra outro servidor DNS ou estabrecer outro tipo de comunicación diferente con 8.8.8.8 ou 8.8.4.4 será bloqueado:

```
uadmin@server:~$ ping -c 2 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1009ms

uadmin@server:~$ host www.xunta.es 208.67.222.222
;; connection timed out; no servers could be reached
uadmin@ubuntu:~$
```

Navegación web

Os comandos iptables para permitir a navegación web usando http/https son:

```
uadmin@server:~$ sudo iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
uadmin@server:~$ sudo iptables -A OUTPUT -p tcp --dport 443 -j ACCEPT
uadmin@server:~$ sudo iptables -A INPUT -p tcp --sport 80 -j ACCEPT
uadmin@server:~$ sudo iptables -A INPUT -p tcp --sport 443 -j ACCEPT
```

Unha vez executados os comandos o ruleset sería:

```
uadmin@server:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target prot opt source destination
ACCEPT udp -- 8.8.8.8 0.0.0.0/0 udp spt:53
ACCEPT udp -- 8.8.4.4 0.0.0.0/0 udp spt:53
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp spt:80
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp spt:443
Chain FORWARD (policy DROP)
target prot opt source destination
Chain OUTPUT (policy DROP)
target prot opt source destination
ACCEPT udp -- 0.0.0.0/0 8.8.8.8 udp dpt:53
ACCEPT udp -- 0.0.0.0/0 8.8.4.4 udp dpt:53
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:80
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:443
uadmin@server:~$
```

Pode verificarse o correcto funcionamento descargando unha páxina web usando o comando wget:

```
uadmin@server:~$ wget http://www.xunta.es
--2015-11-03 23:09:03-- http://www.xunta.es/
Resolviendo www.xunta.es (www.xunta.es)... 85.91.64.254
Conectando con www.xunta.es (www.xunta.es)[85.91.64.254]:80... conectado.
Petición HTTP enviada, esperando respuesta... 301 Moved Permanently
Ubicación: /portada [siguiente]
--2015-11-03 23:09:13-- http://www.xunta.es/portada
Reutilizando la conexión con www.xunta.es:80.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 122463 (120K) [text/html]
Grabando a: “index.html.1”
100%[=====================================>]122.463 276KB/s en 0,4s
```

```
2015-11-03 23:09:14 (276 KB/s) - "index.html.1" guardado [122463/122463]
uadmin@server:~$
```

Outra posibilidade é usar a opcion multiport que permite definir varios portos destino/orixe nunha mesma regra:

```
uadmin@server:~$ sudo iptables -A OUTPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
uadmin@server:~$ sudo iptables -A INPUT -p tcp -m multiport --sports 80,443 -j ACCEPT
```

A continuación pode verse a diferenza entre usar regras con `dport/sport` e `dports/sports`:

```
uadmin@server:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target prot opt source destination
ACCEPT udp -- 8.8.8.8 0.0.0.0/0 udp spt:53
ACCEPT udp -- 8.8.4.4 0.0.0.0/0 udp spt:53
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp spt:80
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp spt:443
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport sports 80,443
Chain FORWARD (policy DROP)
target prot opt source destination
Chain OUTPUT (policy DROP)
target prot opt source destination
ACCEPT udp -- 0.0.0.0/0 8.8.8.8 udp dpt:53
ACCEPT udp -- 0.0.0.0/0 8.8.4.4 udp dpt:53
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp spt:22
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:80
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:443
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443
uadmin@server:~$
```

Acceso ó servidor ssh

Para permitir a administración do server por ssh hai que:

- Permitir os paquetes tcp con destino ó porto 22/tcp do server.
- Permitir os paquetes tcp saíntes do server con orixe o porto 22/tcp.

Os comandos iptables son:

```
uadmin@server:~$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
uadmin@server:~$ sudo iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT
```

Quedando o ruleset:

```
uadmin@server:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target prot opt source destination
ACCEPT udp -- 8.8.8.8 0.0.0.0/0 udp spt:53
ACCEPT udp -- 8.8.4.4 0.0.0.0/0 udp spt:53
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp spt:80
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp spt:443
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22
Chain FORWARD (policy DROP)
target prot opt source destination
Chain OUTPUT (policy DROP)
target prot opt source destination
ACCEPT udp -- 0.0.0.0/0 8.8.8.8 udp dpt:53
ACCEPT udp -- 0.0.0.0/0 8.8.4.4 udp dpt:53
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:80
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:443
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp spt:22
uadmin@server:~$
```

2.4 Buscando axuda no man (manual no terminal)

Á hora de construír un comando iptables podemos facer uso de diferentes comandos (`-A`, `-D`, `-I`, ...), parámetros (`-s`, `-d`, `-p`, ...), condicións ou matches (`state`, `time`, `limit`, ...), opcións asociadas ós módulos (matches) escollidos (`dport`, `timestart`, `source-port`, `quota`, ...) e accións ou targets (`ACCEPT`, `DROP`, `REJECT`, `REDIRECT`, ...). Podemos acceder rápidamente á axuda a través das páxinas do manual con `man iptables`:

```
manuel@server:~$ man iptables
```

```
IPTABLES(8) iptables 1.4.21
NAME
 iptables/ip6tables — administration tool for IPv4/IPv6 packet filtering and NAT
SYNOPSIS
 iptables [-t table] {-A|-C|-D} chain rule-specification
 ip6tables [-t table] {-A|-C|-D} chain rule-specification
 iptables [-t table] -I chain [rulenum] rule-specification
 iptables [-t table] -R chain rulenum rule-specification
 iptables [-t table] -D chain rulenum
 iptables [-t table] -S [chain [rulenum]]
 iptables [-t table] {-F|-L|-Z} [chain [rulenum]] [options...]
 iptables [-t table] -N chain
 iptables [-t table] -X [chain]
 iptables [-t table] -P chain target
 iptables [-t table] -E old-chain-name new-chain-name
 rule-specification = [matches...] [target]
 match = -m matchname [per-match-options]
 target = -j targetname [per-target-options]
...
```

Con `man iptables-extensions` podemos revisar a documentación relativa áscondicións (*matches*) e as accións (*targets*):

```
manuel@server:~$ man iptables-extensions
iptables-extensions(8) iptables 1.4.21 iptables-extensions(8)
NAME
 iptables-extensions — list of extensions in the standard iptables distribution
SYNOPSIS
 ip6tables [-m name [module-options...]] [-j target-name [target-options...]
 iptables [-m name [module-options...]] [-j target-name [target-options...]
MATCH EXTENSIONS
 iptables can use extended packet matching modules with the -m or --match options, followe
...
 iprange
 This matches on a given arbitrary range of IP addresses.
 [!] --src-range from[-to]
 Match source IP in the specified range.
 [!] --dst-range from[-to]
 Match destination IP in the specified range.
...
```

```
TARGET EXTENSIONS
 iptables can use extended target modules: the following are included in the standard dist
...
REJECT (IPv4-specific)
 This is used to send back an error packet in response to the matched packet: otherwise it
 --reject-with type
 The type given can be icmp-net-unreachable, icmp-host-unreachable, icmp-port-unreachable
won't accept your mail otherwise).
```

3 Connection tracking

O seguimento de conexións permite a Netfilter saber en que estado se atopa unha conexión determinada; entendendo neste contexto por conexión, comunicacións tcp, udp ou icmp. Dependendo do protocolo ó que pertencen, os paquetes de comunicacións poden estar en diferentes estados dentro do núcleo; polo que, para simplificar o tratamento dos estados dos paquetes en netfilter defínense 5 estados xenéricos que engloban os estados particulares de todos os protocolos e ós que poderemos referirnos con iptables, usando a condición `--state`, para filtrar paquetes baseándonos no seu estado:

- **NEW:** fai referencia ó primeiro paquete dunha conexión.
- **ESTABLISHED**: sinala unha conexión xa establecida. O único requisito para chegar a este estado é que tras o envío dun paquete o sistema recibise unha resposta do destinatario. O estado NEW pasará a ESTABLISHED no momento en que chegue un paquete de resposta ó firewall.
- **RELATED**: unha conexión considérase RELATED cando está relacionada cunha conexión xa establecida (en estado ESTABLISHED). Isto é, primeiro é necesario ter unha conexión xa establecida que lanza unha nova conexión, que é considerada como RELATED. Un exemplo de conexións RELATED son as conexións de datos FTP que lanzan as conexións de control FTP para a transferencia de arquivos, ou as mensaxes de erro ICMP.
- **INVALID**: Cando un paquete non pode ser identificado ou non se encontra en ningún estado considéraselle en estado INVALID. Recoméndase descartar todos os paquetes que cheguen neste estado ao devasas.
- **UNTRACKED**: un paquete encóntrase neste estado cando se deshabilitou o seguimento de conexións sobre el.

Traballar con estados é moi interesante; xa que, ademais de aumentar o control sobre o tráfico, permite facer *rulesets* máis sinxelos. Por exemplo, supoñer que detrás dun firewall de rede hai un servidor ssh e quérese que o firewall só acepte conexións tcp ó porto 22. Chegaría con usar dúas

regras: unha primeira regra que acepte todos os paquetes tcp en estado NEW ó porto 22, e unha segunda que acepte os paquetes de tipo ESTABLISHED.

A continuación veranse algúns exemplos de comandos iptables onde se traballa co estado do paquete:

- `$ sudo iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT`
    - Este comando engade unha regra para permitir a saída de paquetes dende o propio equipo pertencentes a conexións autorizadas previamente por outras regras ou paquetes asociados a esas conexións autorizadas.
        - `-A OUTPUT` engade a regra ó final da cadea OUTPUT (onde están as regras que se aplican ós paquetes creados pola propia máquina).
        - `-m state --state ESTABLISHED,RELATED` para indicar que se aplicará ós paquetes en estado ESTABLISHED e RELATED. É dicir, paquetes asociados a conexións establecidas e autorizadas previamente.
        - `-j ACCEPT` para indicar que os paquetes que cumpran todas as condición serán aceptados e continuarán procesándose (enviaranse á táboa de encamiñamento e despois POSTROUTING).
- `$ sudoiptables -A OUTPUT -p udp --dport 53 -d 80.58.32.97 -m state --state NEW -j ACCEPT`
    - Este comando engade unha regra para permitir lanzar consultas dns ó servidor DNS con IP 80.58.32.97
        - `-A OUTPUT` engade a regra ó final da cadea OUTPUT.
        - `-p udp` para indicar que a regra se aplica sobre paquetes que a nivel de transporte viaxan sobre o protocolo udp.
        - `--dport 53` para indicar o porto destino dos paquetes.
        - `-d 80.58.32.97` para indicar que a IP destino dos paquetes é 80.58.32.97.
        - `-m state --state NEW` para autorizar o primeiro paquete dunha conexión.
        - `-j ACCEPT` para indicar que os paquetes que cumpran todas as condición serán aceptados e continuarán procesándose.

Nalgunhas distribución Linux no ficheiro `/proc/net/ip_conntrack` mantense unha táboa con todas as conexións abiertas:

```
udp 17 27 src=192.168.0.205 dst=80.58.32.97 sport=45217 dport=53 src=80.58.32.97 dst=192.168.0.205 sport=53 dport=45217 mark=0 use=2
```

```
tcp 6 431999 ESTABLISHED src=192.168.2.201 dst=192.168.2.205 sport=54329 dport=22
src=192.168.2.205 dst=192.168.2.201 sport=22 dport=54329 [ASSURED] mark=0 use=2
udp 17 27 src=192.168.0.205 dst=80.58.32.97 sport=59454 dport=53 src=80.58.32.97
dst=192.168.0.205 sport=53 dport=59454 mark=0 use=2
tcp 6 299 ESTABLISHED src=192.168.0.205 dst=150.214.5.135 sport=41043 dport=80
src=150.214.5.135 dst=192.168.0.205 sport=80 dport=41043 [ASSURED] mark=0 use=2
udp 17 27 src=192.168.0.205 dst=80.58.32.97 sport=34723 dport=53 src=80.58.32.97
dst=192.168.0.205 sport=53 dport=34723 mark=0 use=2
tcp 6 431999 ESTABLISHED src=192.168.0.205 dst=91.189.92.190 sport=32978 dport=80
src=91.189.92.190 dst=192.168.0.205 sport=80 dport=32978 [ASSURED] mark=0 use=2
tcp 6 431999 ESTABLISHED src=192.168.0.205 dst=91.189.92.191 sport=38221 dport=80
src=91.189.92.191 dst=192.168.0.205 sport=80 dport=38221 [ASSURED] mark=0 use=2
tcp 6 431999 ESTABLISHED src=192.168.0.205 dst=91.189.88.33 sport=58345 dport=80
src=91.189.88.33 dst=192.168.0.205 sport=80 dport=58345 [ASSURED] mark=0 use=2
```

En Ubuntu 14.04 e posteriores o arquivo `/proc/net/ip_conntrack` desapareceu e para acceder á información relativa ás conexións hai que empregar a ferramenta `conntrack`:

```
uadmin@ubuntu:~$ sudo apt-get update
uadmin@ubuntu:~$ sudo apt-get install conntrack
uadmin@ubuntu:~$ sudo conntrack -L
unknown 2 599 src=192.168.1.1 dst=224.0.0.1 [UNREPLIED] src=224.0.0.1 dst=192.168.1.1
mark=0 use=1
tcp 6 431996 ESTABLISHED src=192.168.56.1 dst=192.168.56.254 sport=46621 dport=22
src=192.168.56.254 dst=192.168.56.1 sport=22 dport=46621 [ASSURED] mark=0 use=1
tcp 6 73 TIME_WAIT src=192.168.1.2 dst=192.168.1.254 sport=53705 dport=22
src=192.168.56.253 dst=192.168.1.2 sport=22 dport=53705 [ASSURED] mark=0 use=1
tcp 6 431999 ESTABLISHED src=192.168.56.1 dst=192.168.56.254 sport=49370 dport=22
src=192.168.56.254 dst=192.168.56.1 sport=22 dport=49370 [ASSURED] mark=0 use=1
udp 17 176 src=192.168.56.253 dst=8.8.8.8 sport=47920 dport=53 src=8.8.8.8
dst=192.168.1.254 sport=53 dport=47920 [ASSURED] mark=0 use=1
tcp 6 116 TIME_WAIT src=192.168.56.253 dst=104.83.9.145 sport=40273 dport=80
src=104.83.9.145 dst=192.168.1.254 sport=80 dport=40273 [ASSURED] mark=0 use=1
conntrack v1.4.1 (conntrack-tools): 6 flow entries have been shown.
```

Nos seguintes escenarios comezaremos a configurar os nosos firewalls netfilter/iptables usando regras con estados e poderemos usar `conntrack` para ver as conexións.

NOTA:

É posible crear regras iptables con estados usando o match `-m conntrack --ctstate lista_de_estados`. Facer regras con `-m state --state` e con `-m conntrack --ctstate` ven sendo igual; xa que, o match `conntrack` é o reemprazo de `state`. Sendo un

pouco máis estrictos, `state` é considerado un 'subconxunto' do `conntrack`, que permite máis estados. Polo tanto, as seguintes regras son equivalentes:

```
$ sudo iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$ sudoiptables -A OUTPUT -p udp --dport 53 -d 80.58.32.97 -m state --state NEW
-j ACCEPT

$ sudo iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
$ sudoiptables -A OUTPUT -p udp --dport 53 -d 80.58.32.97 -m conntrack
--ctstate NEW -j ACCEPT
```

4 Escenario 2B: Firewall de host con netfilter/iptables (regras con estados persistentes)

Nesta práctica configurarase un firewall de host nunha máquina Linux Ubuntu Server para permitir únicamente o tráfico autorizado, denegando por defecto o resto do tráfico. A diferenza do visto no Escenario 2A; neste caso, traballarase con **estados** (**seguimento de conexións**) e faranse as **regras de filtrado persistentes** (non se perden ó reiniciar a máquina).

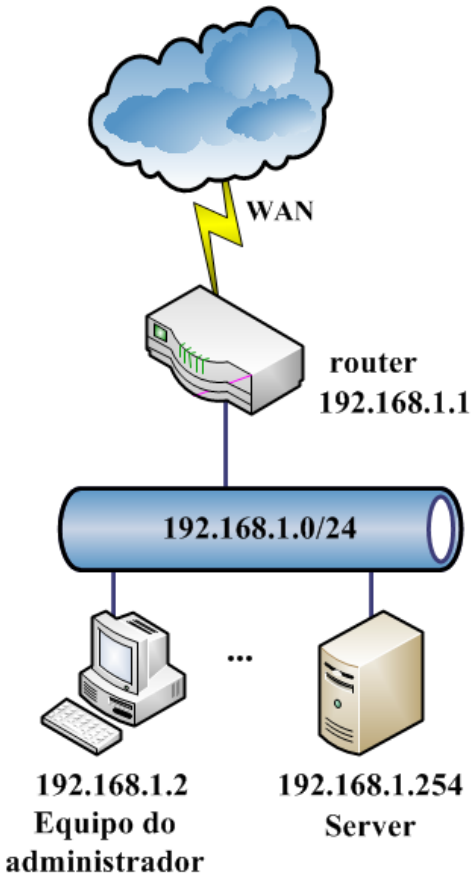
Obxectivo

Configurar o firewall a nivel de host no equipo Server permitindo únicamente o tráfico autorizado

Tráfico autorizado:

- Poderá realizar consultas DNS ós servidores 8.8.8.8 y 8.8.4.4.
- Poderá visitar sitios web (http/https).
- Pode ser xestionado dende calquera equipo a través dun servidor ssh correndo no porto por defecto (tcp/22).

4.1 Configuración do escenario



Configuración do server para o escenario 2B:

Traballarase sobre o server do escenario 2A. Neste momento a situación concreta do equipo pode variar lixeiramente en función das accións realizadas e para partir todos do mesmo estado revisaranse as dúas situacións máis probables:

Situación#1: Apagouse o server e volveuse a arrincar

As regras iptables creadas no escenario 2A non son persistentes; polo que, ó apagar o equipo perdéronse. Pode comprobarse facendo o listado das regras;

```
uadmin@server:~$ sudo iptables -L -n
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
uadmin@server:~$
```

Para preparar o server para iniciar o Escenario 2B, definirase a política de denegar por defecto:

```
uadmin@server:~$ sudo iptables -P INPUT DROP
uadmin@server:~$ sudo iptables -P OUTPUT DROP
uadmin@server:~$ sudo iptables -P FORWARD DROP
uadmin@server:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target prot opt source destination
Chain FORWARD (policy DROP)
target prot opt source destination
Chain OUTPUT (policy DROP)
target prot opt source destination
uadmin@server:~$
```

Lembrade que se estamos accedendo por ssh, estas regras de CleanUp deben crearse despois de autorizar as conexións por ssh.

Situación#2: Non se apagou o server tras rematar o Escenario 2A

Neste caso, as regras non se perderon e son as introducidas no Escenario 2A. Para preparar o server para iniciar o Escenario 2B, borraranse as regras conservando a política de denegar por defecto. Para borrar todas as regras dunha cadea úsase a opción -F:

```
iptables [-t <táboa>] -F [<cadea>]
```

- `-F` borra a lista de regras que hai nunha cadea dunha táboa.
- No caso de omitir a cadea, o comando actúa sobre todas.
- Non afecta á policy (acción por defecto da cadea).

```
uadmin@server:~$ sudo iptables -F
uadmin@server:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target prot opt source destination
Chain FORWARD (policy DROP)
target prot opt source destination
Chain OUTPUT (policy DROP)
target prot opt source destination
uadmin@server:~$
```

4.2 Regras con estados

Os comandos iptables vistos ata o momento son correctos; pero como xa se explicou no tema, netfilter permite traballar con estados. A seguinte regra acepta todos os paquetes con destino o server que pertencen a unha conexión xa autorizada e establecida:

```
uadmin@server:~$ sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Faise o mesmo para permitir a saída do server de todos os paquetes que pertencen a unha conexión xa autorizada e establecida:

```
uadmin@server:~$ sudo iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Agora o que falta é autorizar o establecemento de conexións aceptando o primeiro paquete dunha conexión. A seguinte regra permite que o server lance consultas dns ó servidor 8.8.8.8.

```
uadmin@server:~$ sudo iptables -A OUTPUT -p udp --dport 53 -d 8.8.8.8 -m state --state NEW -j ACCEPT
```

Unha vez autorizado o primeiro paquete, calquera paquete da conexión será aceptado grazas ás dúas regras anteriores:

- A regra "`OUTPUT -m state --state ESTABLISHED,RELATED`" permitirá a saída dos paquetes necesarios para que o equipo fale co servidor 8.8.8.8.
- A regra "`INPUT -m state --state ESTABLISHED,RELATED`" permitirá a entrada das mensaxes de resposta do servidor 8.8.8.8 á consulta efectuada polo equipo.

A continuación están as regras para autorizar a resolución DNS e a navegación web:

```
uadmin@server:~$ sudo iptables -A OUTPUT -p udp --dport 53 -d 8.8.4.4 -m state --state NEW -j ACCEPT
uadmin@server:~$ sudo iptables -A OUTPUT -p tcp -m multiport --dports 80,443 -m state --state NEW -j ACCEPT
```

Para autorizar as conexión por ssh ó server:

```
uadmin@server:~$ sudo iptables -A INPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT
```

O ruleset quedaría:

```
uadmin@server:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target prot opt source destination
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
Chain FORWARD (policy DROP)
target prot opt source destination
Chain OUTPUT (policy DROP)
target prot opt source destination
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
ACCEPT udp -- 0.0.0.0/0 8.8.8.8 udp dpt:53 state NEW
ACCEPT udp -- 0.0.0.0/0 8.8.4.4 udp dpt:53 state NEW
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
uadmin@server:~$
```

A continuación realízanse algunhas probas para verificar o funcionamento do firewall:

```
uadmin@server:~$ host www.edu.xunta.es 8.8.4.4
Using domain server:
Name: 8.8.4.4
Address: 8.8.4.4#53
Aliases:
www.edu.xunta.es has address 85.91.64.102

uadmin@server:~$ ping -c 2 8.8.4.4
PING 8.8.4.4 (8.8.4.4) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
--- 8.8.4.4 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 999ms

uadmin@server:~$ wget http://www.edu.xunta.es
--2015-11-04 11:47:08-- http://www.edu.xunta.es/
Resolviendo www.edu.xunta.es (www.edu.xunta.es)... 85.91.64.102
Conectando con www.edu.xunta.es (www.edu.xunta.es)[85.91.64.102]:80...
conectado.
Petición HTTP enviada, esperando respuesta... 302 Found
Ubicación: http://www.edu.xunta.es/portal [siguiente]
--2015-11-04 11:47:09-- http://www.edu.xunta.es/portal
Reutilizando la conexión con www.edu.xunta.es:80.
```

```
Petición HTTP enviada, esperando respuesta... 302 Found
Ubicación: http://www.edu.xunta.es/portal/ [siguiente]
--2015-11-04 11:47:09-- http://www.edu.xunta.es/portal/
Reutilizando la conexión con www.edu.xunta.es:80.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 81997 (80K) [text/html]
Grabando a: “index.html.2”
100%[=====================================>] 81.997 240KB/s en 0,3s
2015-11-04 11:47:09 (240 KB/s) - “index.html.2” guardado [81997/81997]
uadmin@server:~$
```

Dende o equipo do administrador:

```
manuel@lubuntu:~$ ping -c 2 192.168.1.254
PING 192.168.1.254 (192.168.1.254) 56(84) bytes of data.
--- 192.168.1.254 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1008ms

manuel@lubuntu:~$ ssh uadmin@192.168.1.254
uadmin@192.168.1.254's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-30-generic x86_64)
* Documentation: https://help.ubuntu.com/
 System information as of Wed Nov 4 11:21:06 CET 2015
 System load: 0.0 Processes: 67
 Usage of /: 7.7% of 19.07GB Users logged in: 1
 Memory usage: 10% IP address for eth0: 192.168.1.254
 Swap usage: 0%
 Graph this data and manage this system at:
 https://landscape.canonical.com/
Last login: Wed Nov 4 11:21:06 2015
uadmin@server:~$
```

4.3 Regras persistentes

As regras creadas con iptables son efímeras; polo que, pérdense ó apagar o equipo. Pódese comprobar fácilmente:

```
uadmin@server:~$ sudo reboot
```

E tras arricar verifícase que se perderon todas as regras incluíndo a política de denegar todo:

```
uadmin@server:~$ sudo iptables -L -n
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
uadmin@server:~$
```

As regras creadas con iptables **son efímeras**; polo tanto, pérdense ó apagar o equipo. Para non perdelas pódense empregar varios métodos:

- Crear un script cos comandos iptables correspondentes que se carga ó inicio do sistema (usando, por exemplo, FWBuilder).
- Usar o servizo `iptables-persistent` que se encarga de recuperar as regras iptables configuradas nos reinicios.

iptables-persistent

Neste escenario empregarase `iptables-persistent` para facer as regras persistentes. O primeiro paso será recuperar as regras borradas:

```
uadmin@server:~$ sudo iptables -P INPUT DROP
uadmin@server:~$ sudo iptables -P OUTPUT DROP
uadmin@server:~$ sudo iptables -P FORWARD DROP
uadmin@server:~$ sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j
ACCEPT
uadmin@server:~$ sudo iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j
ACCEPT
uadmin@server:~$ sudo iptables -A OUTPUT -p udp --dport 53 -d 8.8.8.8 -m state
--state NEW -j ACCEPT
```

```
uadmin@server:~$ sudo iptables -A OUTPUT -p udp --dport 53 -d 8.8.4.4 -m state
--state NEW -j ACCEPT
uadmin@server:~$ sudo iptables -A OUTPUT -p tcp -m multiport --dports 80,443 -m
state --state NEW -j ACCEPT
uadmin@server:~$ sudo iptables -A INPUT -p tcp --dport 22 -m state --state NEW
-j ACCEPT
uadmin@server:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target prot opt source destination
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
Chain FORWARD (policy DROP)
target prot opt source destination
Chain OUTPUT (policy DROP)
target prot opt source destination
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
ACCEPT udp -- 0.0.0.0/0 8.8.8.8 udp dpt:53 state NEW
ACCEPT udp -- 0.0.0.0/0 8.8.4.4 udp dpt:53 state NEW
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
uadmin@server:~$
```

En segundo lugar instalarase o paquete iptables-persistent:

```
uadmin@server:~$ sudo apt-get update
uadmin@server:~$ sudo apt-get install iptables-persistent
```

Durante a instalación pregúntase se se queren gardar as regras actuais (tanto para IPv4 coma para IPv6):

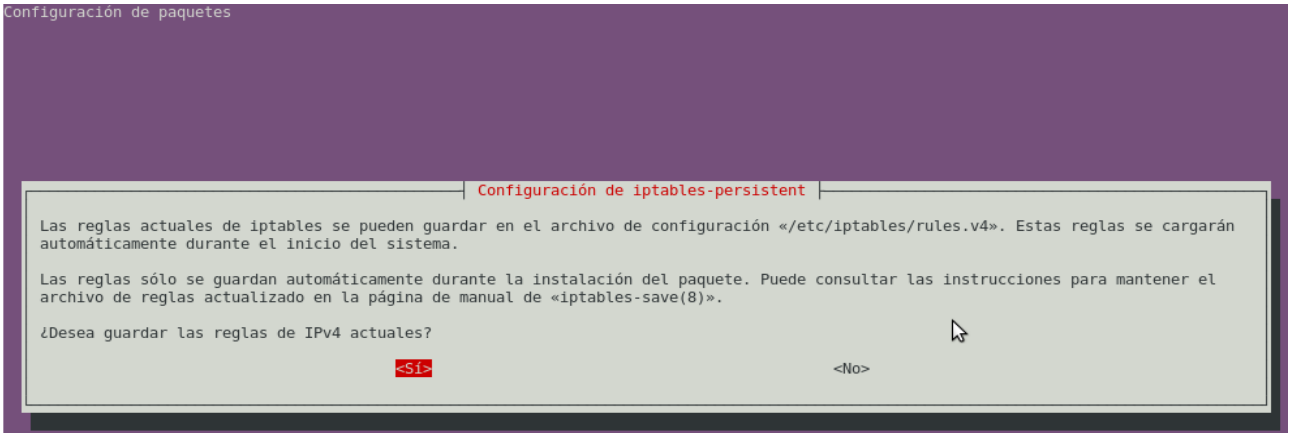


Fig. Configuración de iptables-persistent durante a súa instalación

Gardando as regras actuais e unha vez rematada a instalación, no directorio /etc/iptables aparecerán dous arquivos coas regras, un para IPv4 e outro para IPv6:

```
uadmin@server:~$ ls -lhF /etc/iptables/
total 8,0K
-rw-r--r-- 1 root root 596 nov 4 13:00 rules.v4
-rw-r--r-- 1 root root 184 nov 4 13:00 rules.v6
```

No arquivo `/etc/iptables/rules.ipv4` atoparanse as regras anteriormente introducidas e gardadas durante a instalación:

```
uadmin@server:~$ cat /etc/iptables/rules.v4
# Generated by iptables-save v1.4.21 on Wed Nov 4 13:00:42 2015
*filter
:INPUT DROP [41:5925]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -m state --state NEW -j ACCEPT
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -d 8.8.8.8/32 -p udp -m udp --dport 53 -m state --state NEW -j ACCEPT
-A OUTPUT -d 8.8.4.4/32 -p udp -m udp --dport 53 -m state --state NEW -j ACCEPT
-A OUTPUT -p tcp -m multiport --dports 80,443 -m state --state NEW -j ACCEPT
COMMIT
# Completed on Wed Nov 4 13:00:42 2015
```

Cando o equipo se reinicie, `iptables-persistent` lerá os arquivos de regras e procederá a cargalas:

```
uadmin@server:~$ sudo reboot
uadmin@server:~$ sudo iptables -L -n
[sudo] password for uadmin:
Chain INPUT (policy DROP)
target prot opt source destination
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
Chain FORWARD (policy DROP)
target prot opt source destination
Chain OUTPUT (policy DROP)
target prot opt source destination
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
```

```
ACCEPT udp -- 0.0.0.0/0 8.8.8.8 udp dpt:53 state NEW
ACCEPT udp -- 0.0.0.0/0 8.8.4.4 udp dpt:53 state NEW
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
uadmin@ubuntu:~$
```

A pregunta que xurde agora é como facer para cambiar o ruleset e que `iptables-persistent` recoñeza eses cambios. Pode facerse de varias formas:

- Editando directamente o arquivo `/etc/iptables/rules.v4`:
  - Hai que introducir as novas regras na súa posición, respectando non escribir debaixo de COMMIT.
  - Unha vez modificado o `/etc/iptables/rules.v4` hai que reiniciar o servizo con **`$ sudo service iptables-persistent restart`**
  - No caso de traballar con IPv6 sería semellante pero modificando o arquivo `/etc/iptables/rules.v6`
- Executando directamente comandos iptables:
  - Executaríanse os comandos iptables correspondentes para crear as novas regras.
  - Gárdanse os cambios no ruleset mediante: **`$ sudo sh -c "iptables-save > /etc/iptables/rules.v4".`** A próxima vez que se reinicie o sistemas (ou o servizo con **`$ sudo service iptables-persistent restart`**) cargaranse as regras gardadas no arquivo **`/etc/iptables/rules.v4`.**
  - No caso de traballar con IPv6 sería semellante pero co comando **`ip6tables-save`** e o arquivo `/etc/iptables/rules.v6`
- Executando directamente comandos iptables:
  - Executaríanse os comandos iptables correspondentes para crear as novas regras.
  - Gárdanse os cambios no ruleset mediante: **`$ sudo service iptables-persistent save`**
  - NOTA: a opción `save` non é operativa nalgunhas versións vellas de `iptables-persistent` e non garda as regras.

Apuntes para Ubuntu Server 16.04

Aínda que o proceso de instalación de **`iptables-persistent`** é igual ó visto para Ubuntu 14.04, os comandos para gardar as regras e reiniciar o servizo son diferentes. Como se pode ver en Ubuntu 16.04 usar o comando `iptables-persistent` da erros:

```
uadmin@ubuntu16:~$ sudo service iptables-persistent save
iptables-persistent: unrecognized service
uadmin@ubuntu16:~$ sudo service iptables-persistent restart
Failed to restart iptables-persistent.service: Unit iptables-persistent.service
```

```
not found.
uadmin@ubuntu16:~$
```

En Ubuntu 16.04 o comando`iptables-persistent` é sustituído por **`netfilter-persistent`** tanto para gardar as regras como para reiniciar o servizo:

```
uadmin@ubuntu16:~$ sudo service netfilter-persistent save
* Saving netfilter rules...
run-parts: executing /usr/share/netfilter-persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
uadmin@ubuntu16:~$ sudo service netfilter-persistent restart
```

Tendo en conta este punto, as diferentes formas de facer persistentes as regras vistas no punto anterior, seguen sendo válidas para Ubuntu 16.04.

A modo de exemplo crease unha nova regra:

```
uadmin@server:~$ sudo iptables -A INPUT -p tcp --dport 33000 -m state --state NEW -j ACCEPT
```

Pode comprobarse que no arquivo `/etc/iptables/rules.v4` non aparecen as entradas correspondentes:

```
uadmin@server:~$ cat /etc/iptables/rules.v4
# Generated by iptables-save v1.4.21 on Wed Nov 4 13:58:21 2015
*filter
:INPUT DROP [3:468]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -m state --state NEW -j ACCEPT
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -d 8.8.8.8/32 -p udp -m udp --dport 53 -m state --state NEW -j ACCEPT
-A OUTPUT -d 8.8.4.4/32 -p udp -m udp --dport 53 -m state --state NEW -j ACCEPT
-A OUTPUT -p tcp -m multiport --dports 80,443 -m state --state NEW -j ACCEPT
COMMIT
# Completed on Wed Nov 4 13:58:21 2015
```

Para actualizar `/etc/iptables/rules.v4` execútase:

```
uadmin@server:~$ sudo sh -c "iptables-save > /etc/iptables/rules.v4"
```

```
uadmin@server:~$ cat /etc/iptables/rules.v4
# Generated by iptables-save v1.4.21 on Wed Nov 4 14:04:08 2015
*filter
:INPUT DROP [40:5893]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -m state --state NEW -j ACCEPT
-A INPUT -p tcp -m tcp --dport 33000 -m state --state NEW -j ACCEPT
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -d 8.8.8.8/32 -p udp -m udp --dport 53 -m state --state NEW -j ACCEPT
-A OUTPUT -d 8.8.4.4/32 -p udp -m udp --dport 53 -m state --state NEW -j ACCEPT
-A OUTPUT -p tcp -m multiport --dports 80,443 -m state --state NEW -j ACCEPT
COMMIT
# Completed on Wed Nov 4 14:04:08 2015
```

Agora, a próxima vez que se reinicie o sistema (ou `iptables-persistent`) cargarase a nova regra.

4.4 Policy DROP

Como se indicou no apartado de Cadeas (*chains*) e ó falar das CleanUp rules:

- Un paquete vai pasando por cada regra dunha cadea ata que casa cunha regra; momento no que, procédese a executar a acción indicada na regra.
- Se non casa con ningunha regra, chega ata a Policy ou política da cadea, que ven sendo a acción por defecto a executar.
- A política máis segura a usar nun firewall é a de denegar por defecto, onde é permitido unicamente o tráfico requirido polas necesidades da rede e rexeitase o resto.

Estos puntos levan dun xeito natural a crear as pertinentes regras para autorizar o tráfico desexado e a aplicar unha política DROP nas cadeas a través dos comandos:

```
$sudo iptables -P INPUT DROP
$sudo iptables -P OUTPUT DROP
$sudo iptables -P FORWARD DROP
```

Todo correcto, pero ó aplicar unha política DROP poden xurdir '*problemas*' inesperados.

Problemática

Hai que entender que con Policy DROP bloqueamos todo o tráfico, agás o explícitamente autorizado; e tamén, hai que darse conta que en canto escribimos un comando iptables e prememos

Enter, a orde execútase inmediatamente. Netfilter/iptables non é como outros firewalls onde hai que gardar os cambios e despois aplicalos; aquí, a execución é inmediata.

Esto é especialmente problemático cando traballamos cunha máquina remota e polo motivo que sexa, temos que facer unha limpeza das regras do firewall. A modo de exemplo, estamos a traballar nun servidor remoto que administramos por ssh e que ten o seguinte ruleset:

```
uadmin@server:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
3 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
Chain FORWARD (policy DROP)
num target prot opt source destination
Chain OUTPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT udp -- 0.0.0.0/0 8.8.8.8 udp dpt:53 state NEW
3 ACCEPT udp -- 0.0.0.0/0 8.8.4.4 udp dpt:53 state NEW
4 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
uadmin@server:~$
```

Como pode verse, o acceso por ssh ó equipo remoto está asegurado grazas a regra 2 da cadea INPUT e as regras 1 das cadeas INPUT e OUTPUT. Supoñamos, que queremos borrar as regras e procedemos a facer un `iptables -F`:

```
uadmin@server:~$ sudo iptables -F
```

```
-F, --flush [chain]
Flush the selected chain (all the chains in the table if none is given). This is
equivalent to deleting all the rules one by one.
```

O ruleset resultante é:

```
uadmin@server:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy DROP)
num target prot opt source destination
```

```
Chain FORWARD (policy DROP)
num target prot opt source destination
Chain OUTPUT (policy DROP)
num target prot opt source destination
```

Limpamos as regras pero o ruleset resultante fai que perdamos a conexión por ssh ó server (fixarse que xa non hai regras ESTABLISHED). Non só perdemos a conexión ssh actual, tamén bloqueamos totalmente o acceso ó equipo!!!! Habería que acceder físicamente ó server para solventar o problema.

Considero moi interesante facer esta práctica co alumnado. Eu nas clases, para que se acostumen a traballar en remoto, obrígolles a traballar sempre por ssh (evitando teclear directamente na pantalla da máquina virtual en virtualbox). E cando chegamos a este tema; sen avisarlles do perigro, fago que limpen as regras do firewall como parte dunha práctica. Terminan bloqueándose a eles mesmos; e deste xeito, percátanse da importancia de pensar antes de executar unha modificación no ruleset. Eles están diante do equipo onde corren as súas máquinas virtuais e o arranxo é rápido e sinxelo; pero se estivesen nunha situación real, o erro pode ser catastrófico.

Solucións

O que está claro é que sempre, sempre, sempre, hai que ser consciente dos cambios a executar e as súas consecuencias. A política DROP non é un problema se facemos as cousas ben; sen embargo, podemos adoptar certas medidas sinxelas que nos protexan de despistes:

Protección#1

Limpar as regras do ruleset empregando a opción flush de `iptables-persistent` en vez de con `iptables -F`. Conseguimos limpar as regras e ademais as políticas de filtrado configúranse a ACCEPT; polo que, temos garantido o acceso:

```
uadmin@server:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
3 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
Chain FORWARD (policy DROP)
num target prot opt source destination
Chain OUTPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT udp -- 0.0.0.0/0 8.8.8.8 udp dpt:53 state NEW
3 ACCEPT udp -- 0.0.0.0/0 8.8.4.4 udp dpt:53 state NEW
```

```
4 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
uadmin@server:~$ sudo service iptables-persistent flush
* Flushing rules...
* IPv4...
* IPv6...
[ OK ]
uadmin@server:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination
Chain FORWARD (policy ACCEPT)
num target prot opt source destination
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
uadmin@ubuntu:~$
```

Apuntes para Ubuntu 16.04

Como xa foi explicado con anterioridade, debemos usar `netfilter-persistent` en vez de `iptables-persistent`:

```
uadmin@ubuntu16:~$ sudo service netfilter-persistent flush
* Flushing netfilter rules...
run-parts: executing /usr/share/netfilter-persistent/plugins.d/15-ip4tables flush
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables flush
uadmin@ubuntu16:~$ sudo iptables -L -n
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
uadmin@ubuntu16:~$
```

Protección#2:

Neste caso non definimos a Policy a DROP senon que:

1. Definimos a Policy da táboa filter a ACCEPT.
2. Creamos nas cadeas unha última regra de CleanUP, onde explícitamente procedemos a eliminar todo o tráfico.

Deste xeito, a política de tráfico do firewall segue sendo denegar todo por defecto (última regra); pero, se borramos as regras con `iptables -F`, como a Policy está a ACCEPT non quedamos bloqueados. A continuación podemos ver como se crean as regras finais de CleanUp pero conservando a Policy a ACCEPT e o que sucede ó facer un `iptables -F`:

```
uadmin@server:~$ sudo iptables -P INPUT ACCEPT
uadmin@server:~$ sudo iptables -P OUTPUT ACCEPT
uadmin@server:~$ sudo iptables -P FORWARD ACCEPT
uadmin@server:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
3 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
Chain FORWARD (policy ACCEPT)
num target prot opt source destination
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT udp -- 0.0.0.0/0 8.8.8.8 udp dpt:53 state NEW
3 ACCEPT udp -- 0.0.0.0/0 8.8.4.4 udp dpt:53 state NEW
4 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
uadmin@server:~$ sudo iptables -A INPUT -j DROP
uadmin@server:~$ sudo iptables -A OUTPUT -j DROP
uadmin@server:~$ sudo iptables -A FORWARD -j DROP
uadmin@server:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
3 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
4 DROP all -- 0.0.0.0/0 0.0.0.0/0
Chain FORWARD (policy ACCEPT)
num target prot opt source destination
1 DROP all -- 0.0.0.0/0 0.0.0.0/0
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT udp -- 0.0.0.0/0 8.8.8.8 udp dpt:53 state NEW
3 ACCEPT udp -- 0.0.0.0/0 8.8.4.4 udp dpt:53 state NEW
```

```
4 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
5 DROP all -- 0.0.0.0/0 0.0.0.0/0
uadmin@server:~$ sudo iptables -F
uadmin@server:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination
Chain FORWARD (policy ACCEPT)
num target prot opt source destination
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
uadmin@server:~$
```

4.5 DROP vs REJECT

En netfilter/iptables cando desexamos decartar paquetes podemos optar por DROP ou REJECT:

- **DROP** proporciona un **descarte silencioso**; é dicir, mata o paquete sen informar ó remitente do mesmo. Esto tradúcese en que o emisor do paquete queda á espera de recibir resposta do destinatario ata esgotar os temporizadores e descartar a conexión por timeout.
- **REJECT** proporciona un **descarte informado**; é dicir, mata o paquete informando do feito ó remitente do mesmo. A forma de informar consiste no envío dunha mensaxe de erro que permite ó emisor darse por enterado da situación e abortar a conexión sen agardar o timeout.

No caso de usar REJECT podemos especificar o tipo de mensaxe de erro usando `--reject-with type` onde `type` pode ser (ver `man iptables-extensions`):

- icmp-net-unreachable
- icmp-host-unreachable
- icmp-port-unreachable (opción por defecto)
- icmp-proto-unreachable
- icmp-net-prohibited
- icmp-host-prohibited
- icmp-admin-prohibited
- tcp-reset (opción válida soamente coa opción -p tcp)

Para ver exactamente o que acontece con DROP analizaremos a seguinte imaxe, sacada dunha captura co sniffer Wireshark, onde un equipo trata de conectarse a un servidor web e o paquete é descartado silenciosamente no firewall. Podemos ver que o equipo trata de conectarse retransmitindo paquetes e tras un tempo de espera duns 63 segundos para de intentalo (paquetes #1, #3, #5, #7, #9, #11 e #13).

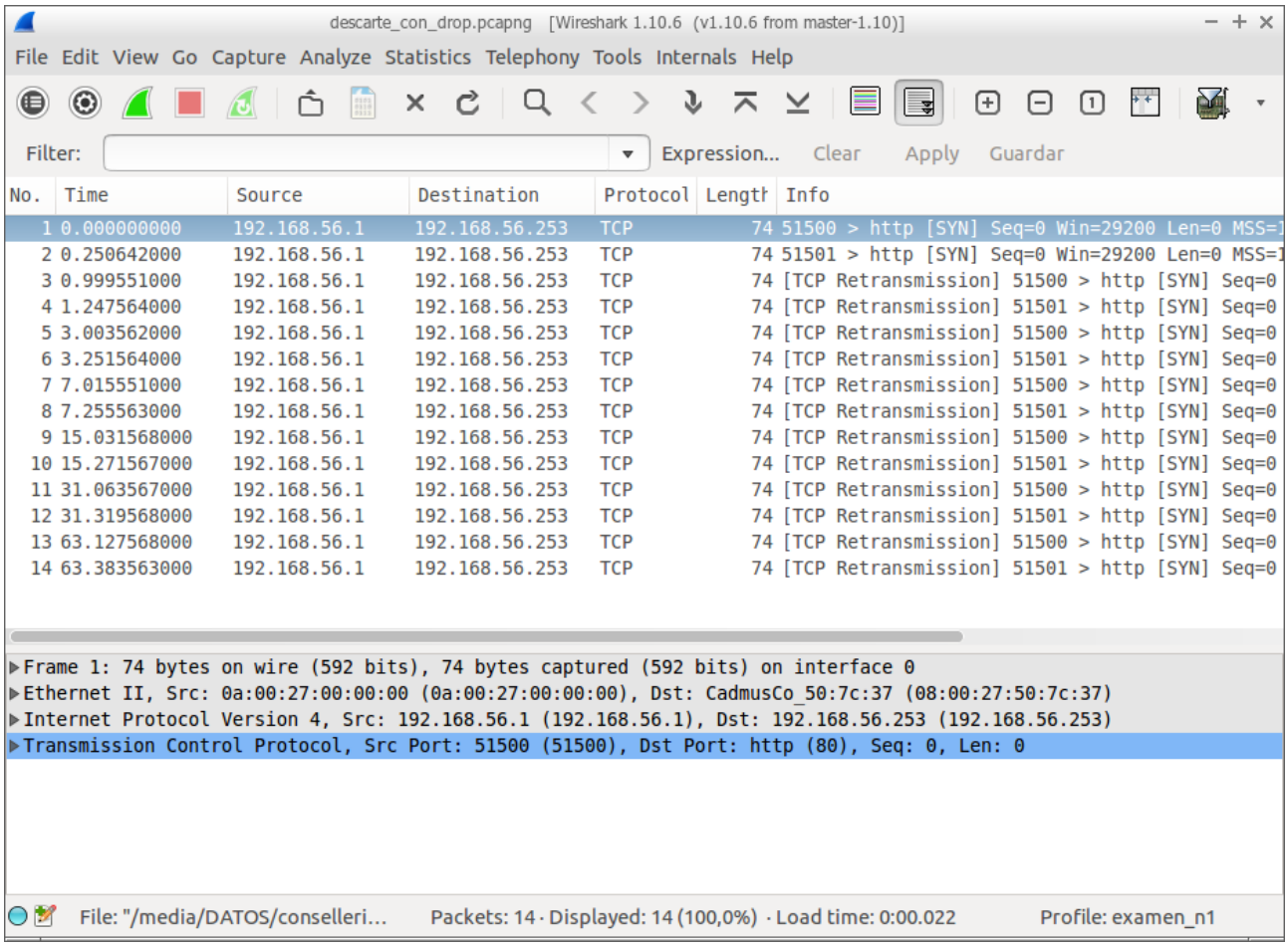


Fig. Descarte silencioso con DROP

Na seguinte imaxe podemos ver o que acontece cando o mesmo intento de conexión é abortado cun REJECT. En canto o paquete chega ó firewall é descartado e o firewall envía unha mensaxe de erro ICMP Destino Inalcanzable - Porto inalcanzable.

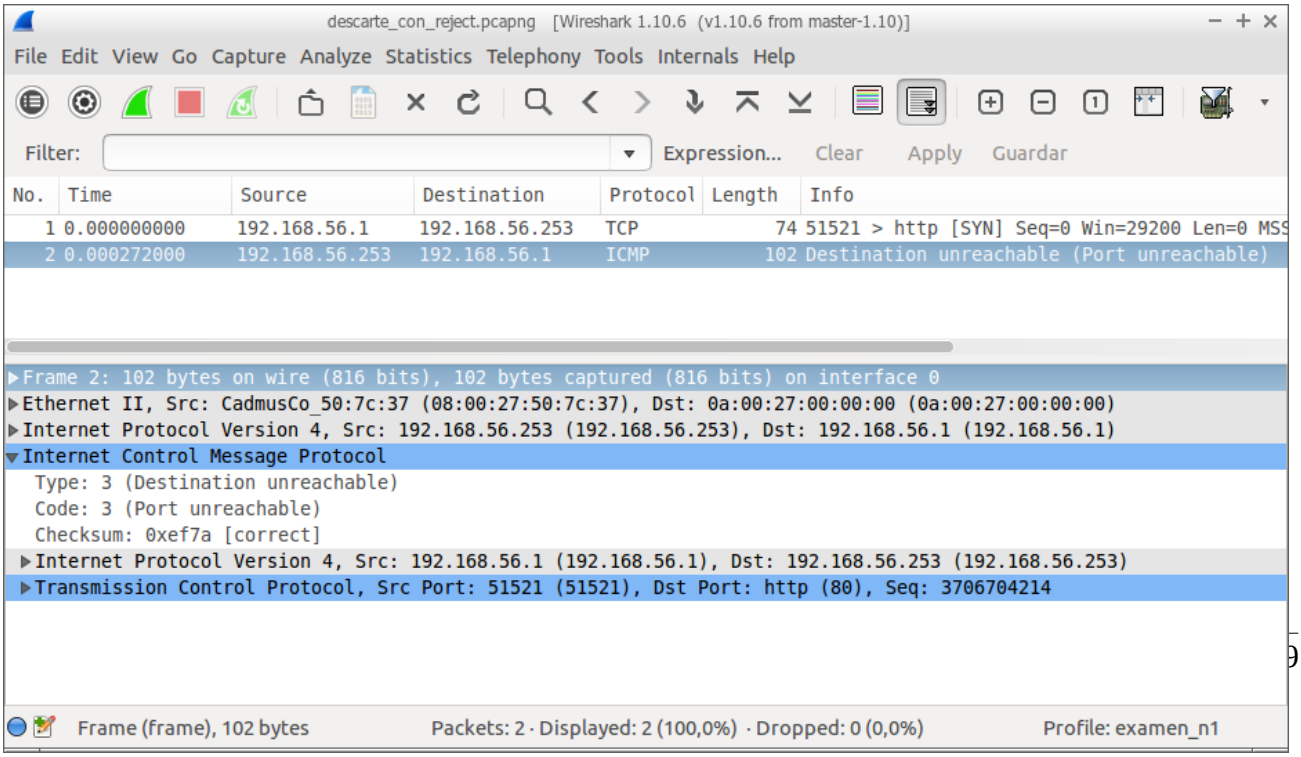


Fig. Descarte informado con REJECT

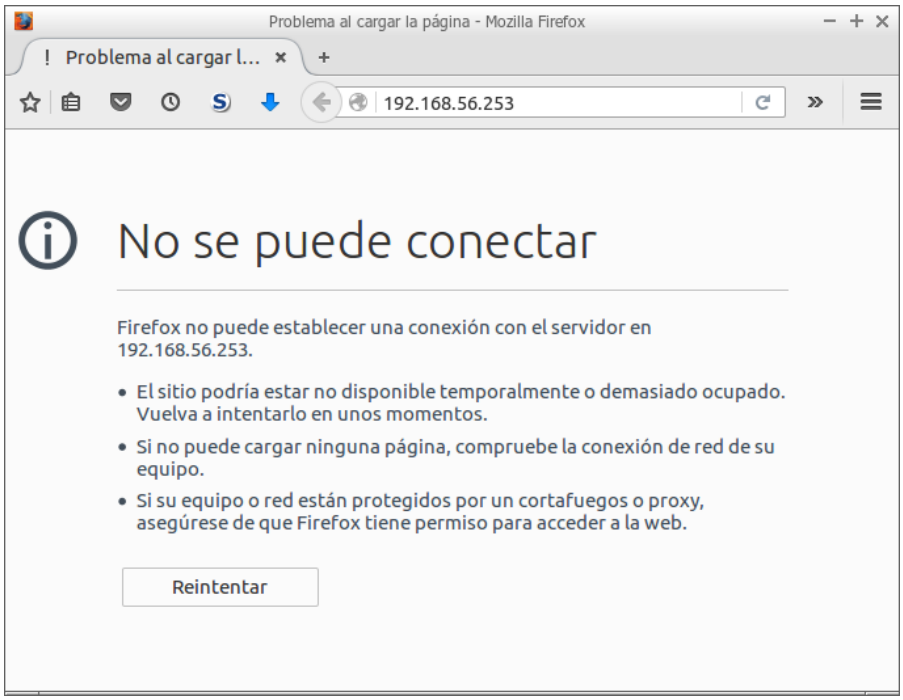


Fig. Erro no cliente

Dende o punto de vista dun cliente, DROP fai que o usuario teña que agardar certo tempo ata que a conexión dase por perdida (seguramente tras varios intentos de conexión) e con REJECT non. Usar REJECT durante a fase de probas do ruleset dun firewall aforra tempo ó non facer que nos desesperemos agardando os timeout; e despois de verificado o funcionamento, sempre se pode cambiar a DROP se o desexamos.

Un punto a ter en conta é que con DROP o firewall permanece oculto e con REJECT o firewall descóbrese ó enviar unha mensaxe de erro coa súa IP. Non por estar agochado un equipo vai ser máis seguro, pero tampouco hai ir anunciando a súa presenza.

Eu considero que o alumnado debe saber traballar con certa soltura con sniffers para capturar paquetes (tanto en contornos gráficos -Wireshark- coma liña de comandos -tcpdump-) e para interpretalos (en contorno gráfico, con Wireshark por exemplo); xa que, nalgunhas ocasións é unha fonte de información importantísima (ás veces a única) para saber que ocorre e proceder a buscar solucións. Mandar facer e analizar capturas como as expostas, serve para comprender mellor o funcionamento de netfilter/iptables e de paso mellorar as destrezas no uso de sniffers (escoller interfaces de captura, facer filtros de captura e/ou visualización para quedarse co tráfico de interese, etc.).

5 Escenario 2C: Modificación do ruleset dun firewall de host baseado en netfilter/iptables

Nesta práctica modificarase a configuración dun firewall de host nunha máquina Linux Ubuntu Server. Partindo do Escenario 2B engadiranse e borraranse regras para permitir únicamente o tráfico autorizado. Traballarase con regras con estados (seguimento de conexións) e faranse persistentes.

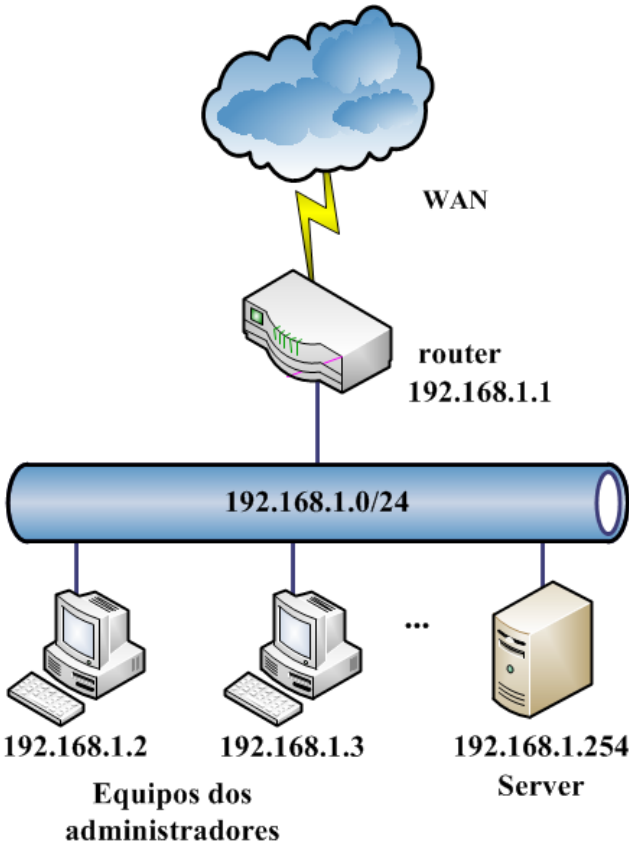
Obxectivo

Modificar o firewall no Server permitindo únicamente o tráfico autorizado e facer persistentes as regras.

Tráfico autorizado:

- Poderá realizar consultas DNS ós servidores 8.8.8.8 y 8.8.4.4.
- Poderá visitar sitios web (http/https).
- Xestionable dende os equipos dos admins a través dun servidor ssh correndo no porto tcp/22000.
- O servidor web http/https correndo en Server accesible para todo o mundo.
- Dende Server poden facer probas tipo ping/traceroute.

5.1 Configuración do escenario



Configuración do server para o escenario 2C:

Con respecto ás regras do firewall traballarase sobre o ruleset do server ó remate do escenario 2B. Para facer o escenario o máis realista posible procederase a configurar axeitadamente os servizos ssh e web:

Servizo SSH

Configurarase o servizo ssh para escoitar no porto tcp/22000:

- En `/etc/ssh/sshd_config` cambiar a directiva `Port 22` por `Port 22000`
- Reiniciar o servizo: **`$ sudo service ssh restart`**
- Verificar que o servidor ssh corre no novo porto con `netstat`. Debería aparecer unha liña semellante a:

```
$ sudo netstat -putan
Proto Recib Enviad Dirección local Dirección remota Estado PID/Program name
tcp 0 0 0.0.0.0:22000 0.0.0.0:* ESCUCHAR 1800/sshd
```

Servizo Web

Instalarase e configurarase o servidor web Apache para traballar tanto con http coma con https. Usarase a páxina web de inicio por defecto e para https o certificado 'snake oil' crea-do polo servidor:

- Instalación do servidor web Apache:

```
uadmin@server:~$ sudo apt-get update
uadmin@server:~$sudo apt-get install apache2
```

- Activación do módulo para ter soporte de https:

```
uadmin@server:~$sudo a2enmod ssl
uadmin@server:~$sudo service apache2 restart
```

- Activación do sitio web https:

```
uadmin@server:~$sudo a2ensite default-ssl.conf
uadmin@server:~$sudo service apache2 reload
```

- Verificar que o servidor Apache corre nos portos http/https con `netstat`:

```
uadmin@server:~$ sudo netstat -putan
Conexiones activas de Internet (servidores y establecidos)
Proto Recib Enviad Dirección local Dirección remota Estado PID/Program name
tcp 0 0 0.0.0.0:22000 0.0.0.0:* ESCUCHAR 1800/sshd
tcp6 0 0 :::80 :::* ESCUCHAR 2664/apache2
tcp6 0 0 :::22000 :::* ESCUCHAR 1800/sshd
tcp6 0 0 :::443 :::* ESCUCHAR 2664/apache2
```

5.2 Modificando o ruleset

Tomamos como punto de partida o ruleset do server ó remate do Escenario 2B:

```
uadmin@server:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
3 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:33000 state NEW
Chain FORWARD (policy DROP)
num target prot opt source destination
Chain OUTPUT (policy DROP)
```

```
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT udp -- 0.0.0.0/0 8.8.8.8 udp dpt:53 state NEW
3 ACCEPT udp -- 0.0.0.0/0 8.8.4.4 udp dpt:53 state NEW
4 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
uadmin@server:~$
```

Fixarse a opción **`--line-numbers`** fai que aparezan as regras numeradas, facilitando así a tarefa de modificar o ruleset. A continuación engadimos as regras para garantir o acceso por ssh dende os equipos dos admins:

```
uadmin@server:~$ sudo iptables -A INPUT -p tcp --dport 22000 -s 192.168.1.2 -m state --state NEW -j ACCEPT
uadmin@server:~$ sudo iptables -A INPUT -p tcp --dport 22000 -s 192.168.1.3 -m state --state NEW -j ACCEPT
```

Engádense as regras para garantir o acceso por http/https ó servidor web correndo en server:

```
uadmin@server:~$ sudo iptables -A INPUT -p tcp -m multiport --dports 80,443 -m state --state NEW -j ACCEPT
```

Borrar regras

Revisando as regras da cadea INPUT vese que hai que borrar as regras #2 e #3:

```
uadmin@server:~$ sudo iptables -L INPUT -n --line-numbers
Chain INPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
3 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:33000 state NEW
4 ACCEPT tcp -- 192.168.1.2 0.0.0.0/0 tcp dpt:22000 state NEW
5 ACCEPT tcp -- 192.168.1.3 0.0.0.0/0 tcp dpt:22000 state NEW
6 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
uadmin@server:~$
```

Para borrar regras emprégase a opción **`-D`**:

```
iptables [-t <táboa>] -D <cadea> <condición> <acción>
iptables [-t <táboa>] -D <cadea> <número_da_regra>
```

No caso de optar polo borrado usando o número da regra, hai que ter en conta que ó borrar unha regra pode cambiar o número das outras:

```
uadmin@server:~$ sudo iptables -D INPUT 2
uadmin@server:~$ sudo iptables -L INPUT -n --line-numbers
Chain INPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:33000 state NEW
3 ACCEPT tcp -- 192.168.1.2 0.0.0.0/0 tcp dpt:22000 state NEW
4 ACCEPT tcp -- 192.168.1.3 0.0.0.0/0 tcp dpt:22000 state NEW
5 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
uadmin@server:~$ sudo iptables -D INPUT 2
uadmin@server:~$ sudo iptables -L INPUT -n --line-numbers
Chain INPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT tcp -- 192.168.1.2 0.0.0.0/0 tcp dpt:22000 state NEW
3 ACCEPT tcp -- 192.168.1.3 0.0.0.0/0 tcp dpt:22000 state NEW
4 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
uadmin@server:~$
```

Se non se queren borrar as regras indicando o número, hai que usar opción `-D` e a regra completa. Por exemplo, o seguinte comando borraría a regra que permite o acceso por ssh a través do porto 22:

```
uadmin@server:~$ sudo iptables -D INPUT -p tcp --dport 22 -m state --state NEW
-j ACCEPT
```

Engadirase unha regra ó final da cadea OUTPUT para permitir o envío de mensaxes ICMP Echo Request para as probas ping/traceroute:

```
uadmin@server:~$ sudo iptables -A OUTPUT -p icmp --icmp-type 8 -j ACCEPT
```

O ruleset queda da seguinte forma:

```
uadmin@server:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT tcp -- 192.168.1.2 0.0.0.0/0 tcp dpt:22000 state NEW
3 ACCEPT tcp -- 192.168.1.3 0.0.0.0/0 tcp dpt:22000 state NEW
```

```
4 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
Chain FORWARD (policy DROP)
num target prot opt source destination
Chain OUTPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT udp -- 0.0.0.0/0 8.8.8.8 udp dpt:53 state NEW
3 ACCEPT udp -- 0.0.0.0/0 8.8.4.4 udp dpt:53 state NEW
4 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
5 ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmptype 8
uadmin@ubuntu:~$
```

Insertar regras

Supoñer que é preciso autorizar outro equipo dos admins ó acceso por ssh e quérese que esa nova regra sexa a número 4. Pódense engadir regras nunha determinada posición usando a opción `-I`:

`iptables [-t <táboa>] -I <cadea> <numero_de_regra> <condición> <acción>`

- engade unha regra na posicion numero_de_regra nunha cadea dunha táboa .

```
uadmin@server:~$ sudo iptables -I INPUT 4 -p tcp --dport 22000 -s 192.168.1.4 -m state --state NEW -j ACCEPT
uadmin@server:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT tcp -- 192.168.1.2 0.0.0.0/0 tcp dpt:22000 state NEW
3 ACCEPT tcp -- 192.168.1.3 0.0.0.0/0 tcp dpt:22000 state NEW
4 ACCEPT tcp -- 192.168.1.4 0.0.0.0/0 tcp dpt:22000 state NEW
5 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
Chain FORWARD (policy DROP)
num target prot opt source destination
Chain OUTPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 ACCEPT udp -- 0.0.0.0/0 8.8.8.8 udp dpt:53 state NEW
3 ACCEPT udp -- 0.0.0.0/0 8.8.4.4 udp dpt:53 state NEW
4 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
5 ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmptype 8
```

Por úiltimo, fanse persistentes as regras:

```
uadmin@server:~$ sudo sh -c "iptables-save > /etc/iptables/rules.v4"
uadmin@server:~$ cat /etc/iptables/rules.v4
# Generated by iptables-save v1.4.21 on Thu Nov 5 01:14:31 2015
*filter
:INPUT DROP [45:6250]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -s 192.168.1.2/32 -p tcp -m tcp --dport 22000 -m state --state NEW -j
ACCEPT
-A INPUT -s 192.168.1.3/32 -p tcp -m tcp --dport 22000 -m state --state NEW -j
ACCEPT
-A INPUT -s 192.168.1.4/32 -p tcp -m tcp --dport 22000 -m state --state NEW -j
ACCEPT
-A INPUT -p tcp -m multiport --dports 80,443 -m state --state NEW -j ACCEPT
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -d 8.8.8.8/32 -p udp -m udp --dport 53 -m state --state NEW -j ACCEPT
-A OUTPUT -d 8.8.4.4/32 -p udp -m udp --dport 53 -m state --state NEW -j ACCEPT
-A OUTPUT -p tcp -m multiport --dports 80,443 -m state --state NEW -j ACCEPT
-A OUTPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT
COMMIT
# Completed on Thu Nov 5 01:14:31 2015
uadmin@server:~$
```

Este é un moi bo momento para facer a Tarefa de Av. II - A.

6 NAT con netfilter/iptables

Netfilter permite facer NAT (Network Address Translation) e NAPT ou PAT (Network Address Port Translation). A táboa nat é a encargada de realizar NAT/NAPT en diferentes escenarios:

- **Un-a-un (1:1)**: tradúcese unha dirección IP privada nunha dirección IP pública.
- **Un-a-moitos (1:Many)**: unha dirección IP privada tradúcese en varias direccións IP públicas. Por cada conexión que o sistema privado establece cun sistema de internet, o router NAT escolle unha dirección IP pública das dispoñibles para ser usada.
- **Moitos-a-un (Many:1)**: varias direccións IP privadas son traducidas nunha soa dirección IP pública. Se a dirección IP pública é a do encamiñador tamén chámaselle enmascaramento (**masquerading**).
- **Moitos-a-moitos (Many:Many)**: varias direccións privadas son traducidas usando un rango de direccións públicas.

En netfilter/iptables emprégase a terminoloxía: SNAT e DNAT.

SNAT (Source Network Address Translation)

SNAT consiste na tradución das direccións IP orixe dos paquetes que atravesan os firewalls nunha ou varias direccións IP públicas. Os sistemas orixe conseguen deste modo acceder a internet sen ter unha dirección IP pública propia e ademais conseguen a protección extra de non ser accesibles dende internet. Masquerade ou enmascaramento é exactamente o mesmo que SNAT, pero usando sempre como dirección pública a dirección pública do firewall.

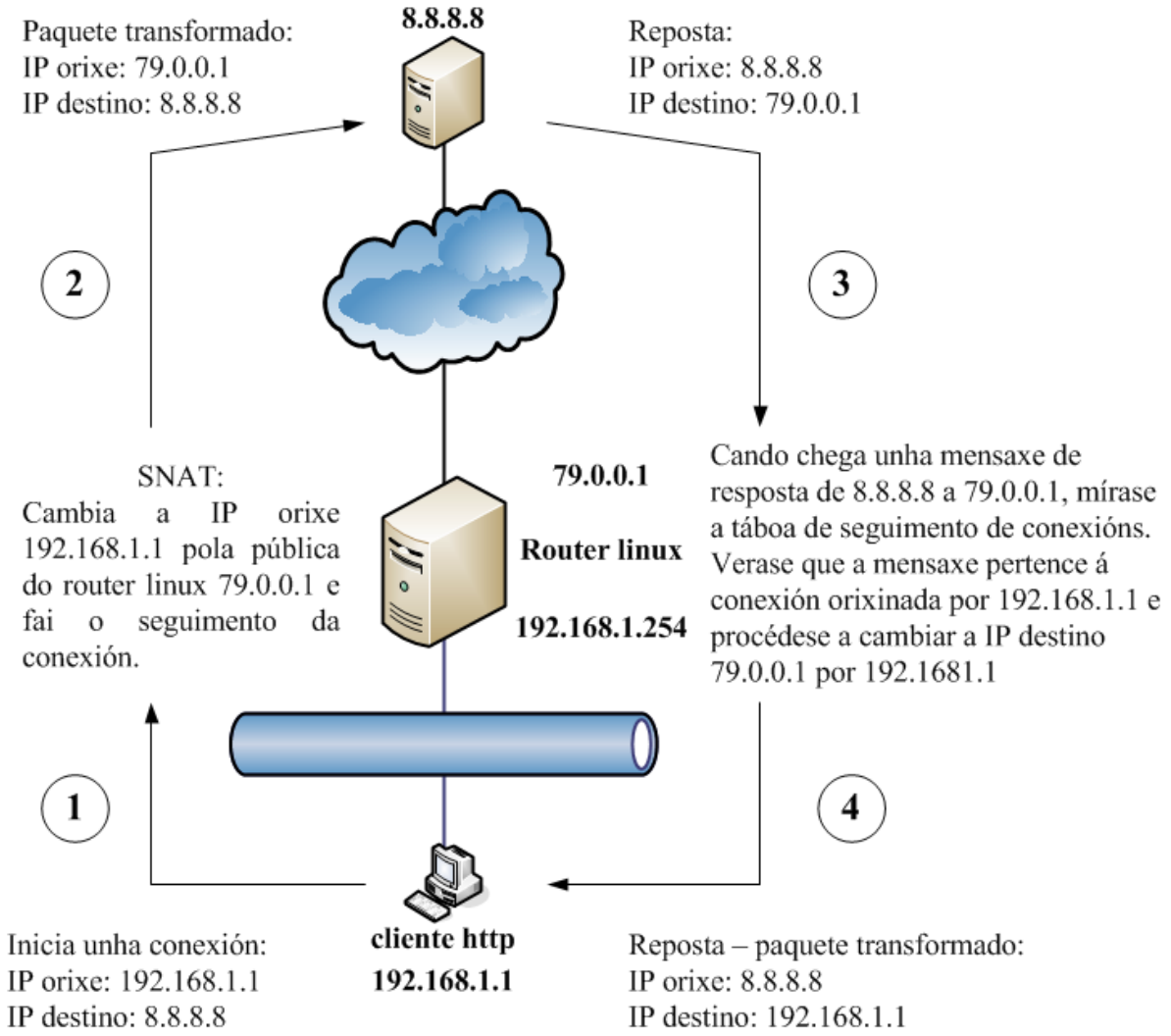


Fig. Exemplo de funcionamento de SNAT

Para comprender a figura anterior, onde se amosa o funcionamento do SNAT, hai que ter presente que as direccións privadas non son válidas para viaxar por Internet; polo que, calquera paquete cunha IP privada non chegará ó destinatario ó ser descartado nos routers de Internet.

1. O cliente http ten a dirección privada 192.168.1.1 e quere conectarse coa máquina 8.8.8.8, polo que envía un paquete con IP orixe 192.168.1.1 e IP destino 8.8.8.8. Como o equipo destino non está na mesma rede encamiña o paquete hacia o router.
2. O router recibe o paquete e cambia a IP orixe 192.168.1.1 (privada) pola IP 79.0.0.1 (pública), gardando a correspondencia entre IP privada-IP pública nunha táboa (esta información será usada para facer a transformación inversa cando chegue a mensaxe de resposta). Agora que o router ten un paquete con IPs públicas (válidas en Internet) tanto no campo orixe como destino, procede a encamiñalo hacia ó destino.
3. O destinatario recibe a petición do cliente e procede a xerar unha mensaxe de resposta. A IP orixe é a do equipo (8.8.8.8) e a IP destino é a que aparecía como orixe na mensaxe recibida (a IP pública 79.0.0.1). Unha vez creada a mensaxe de resposta procede a enviar o paquete.

4. O paquete é encamiñado e chega o router-linux que tras consultar a súa táboa de transformacións, procede a cambiar a IP destino 79.0.0.1 por 192.168.1.1, a IP do cliente que iniciou todo o proceso.

Usando SNAT ou Masquerade, o equipo 192.168.1.1 pode iniciar unha conexión cara a 8.8.8.8; non obstante, o equipo 8.8.8.8 non pode iniciar unha conexión cara a 192.168.1.1, ó ser unha dirección IP privada non alcanzable en Internet. Non confundir iniciar conexións con respostar mediante paquetes a conexións iniciadas polo outro equipo.

En netfilter, SNAT ou Masquerade faise na cadea POSTROUTING e permite especificar que dirección IP orixe debe poñerse (tamén pódese indicar o porto/s):

```
# A IP orixe dos paquetes procedentes de equipos da rede 192.168.1.0/24 será reemprazada por 79.0.0.1
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -j SNAT --to-source 79.0.0.1
# A IP orixe dos paquetes procedentes de equipos da rede 192.168.1.0/24 será reemprazada pola IP pública do router
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -j MASQUERADE
# A IP orixe dos paquetes procedentes de equipos da rede 192.168.1.0/24 será reemprazada por 79.0.0.1 e o porto orixe tcp/udp cambiarase por un do rango 100-1100
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -j SNAT --to-source 79.0.0.1:100-1100
# A IP orixe dos paquetes procedentes de equipos da rede 192.168.1.0/24 será reemprazada por unha IP do rango 79.0.0.1-79.0.0.6
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -j SNAT --to-source 79.0.0.1-79.0.0.6
# A IP orixe dos paquetes procedentes de equipos da rede 192.168.1.0/24 será reemprazada por unha IP dos rangos 79.0.0.1-79.0.0.6 e 80.0.0.1
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -j SNAT --to-source 79.0.0.1-79.0.0.6 --to-source 80.0.0.1
```

Con SAME conséguese SNAT pero ademais trátase de usar sempre a mesma IP pública coa mesma IP privada. Isto é, se un sistema abre unha conexión que é traducida cunha dirección externa, nas próximas conexións que abra o devasas intentará asignarlle a mesma dirección IP externa:

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -j SAME --to 79.0.0.1-79.0.0.6
```

DNAT (Destination Network Address Translation)

DNAT consiste en traducir unha dirección pública IP nunha dirección privada IP, sendo o proceso contrario a SNAT. Úsase habitualmente cando se teñen varios servidores detrás do firewall, de

modo que unha única dirección pública se traduce en varias direccións privadas sobre a base de diferentes portos e protocolos (tamén coñecido como Port Forwarding).

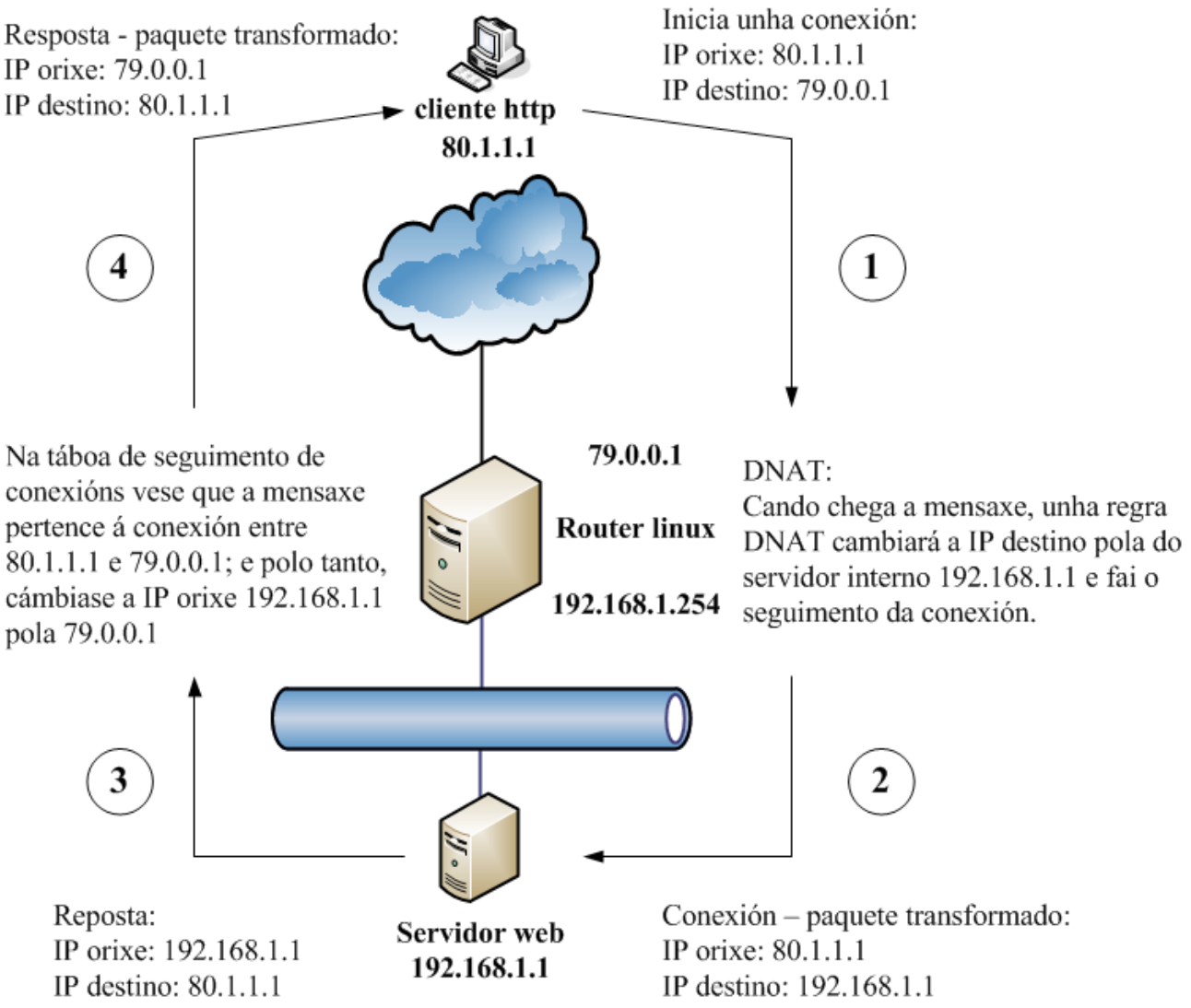


Fig. Exemplo de funcionamento de DNAT

Se DNAT está configurado, pero SNAT non, o equipo 80.1.1.1 poderá establecer conexións con 192.168.1.1 usando 79.0.0.1 como dirección IP destino; non obstante, o equipo 192.168.1.1 non poderá iniciar conexións a 80.1.1.1. Novamente, non confundir iniciar conexións con respostar mediante paquetes a conexións iniciadas polo outro equipo.

En netfilter, DNAT faise nas cadeas PREROUTING e OUTPUT:

```
# A IP destino dos paquetes destinados a 79.0.0.1 será reemprazada por 192.168.1.1
iptables -t nat -A PREROUTING -d 79.0.0.1 -j DNAT --to-destination 192.168.1.1
```

```
# A IP destino dos paquetes tcp destinados a 79.0.0.1 porto 80 será reemprazada por 192.168.1.1
iptables -t nat -A PREROUTING -d 79.0.0.1 -p tcp --dport 80 -j DNAT --to-destination 192.168.1.1
# A IP destino dos paquetes tcp destinados a 79.0.0.1 e a calquera dos portos 22, 80 ou 443 será reemprazada por 192.168.1.1
iptables -t nat -A PREROUTING -d 79.0.0.1 -p tcp -m multiport --dports 22,80,443 -j DNAT --to-destination 192.168.1.1
# Os paquetes tcp destinados a 79.0.0.1 porto 22000 serán transformados para ter IP destino 192.168.1.1 e porto destino tcp/22
iptables -t nat -A PREROUTING -d 79.0.0.1 -p tcp --dport 22000 -j DNAT --to-destination 192.168.1.1:22
# Os paquetes tcp con orixe o equipo 80.0.0.1 e destino 79.0.0.1 porto 22000 serán transformados para ter IP destino 192.168.1.1 e porto destino tcp/22
iptables -t nat -A PREROUTING -s 80.0.0.1 -d 79.0.0.1 -p tcp --dport 22000 -j DNAT --to-destination 192.168.1.1:22
```

Cando se fai SNAT dunha dirección privada nunha pública e DNAT da mesma dirección pública na mesma dirección privada, prodúcese un mapeo dunha dirección privada nunha pública en ambas as dúas direccións, coñecido como tradución total (Full NAT). Pódese facer NAT Total (Full NAT) con:

```
iptables -t nat -A PREROUTING -s 192.168.1.0/24 -j NETMAP --to 79.0.0.0/24
```

7 Escenario 2D: Firewall da rede baseado en netfilter/iptables

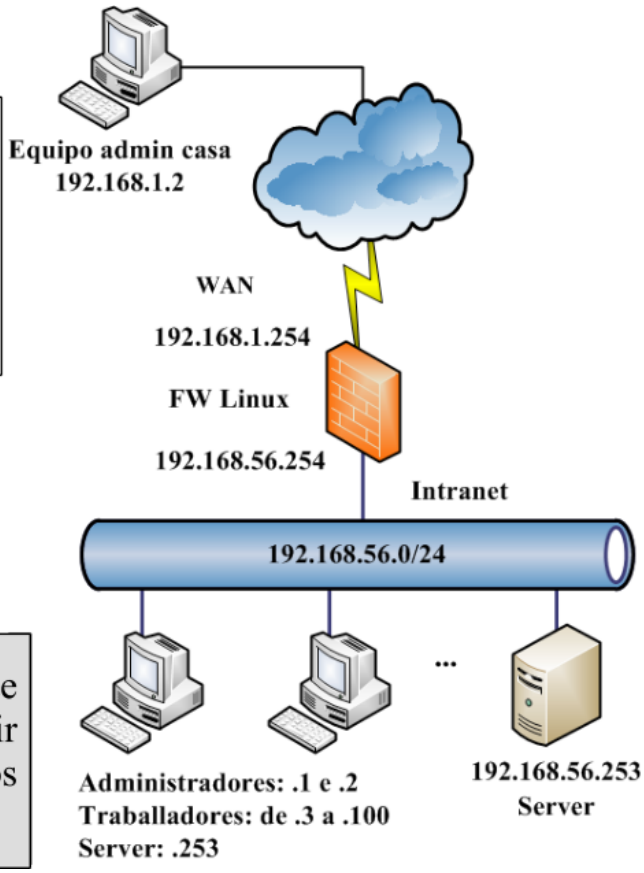
Nesta práctica configuraremos un firewall de rede nunha máquina Linux Ubuntu Server. O firewall de rede terá que procesar o tráfico entrante e saínte da organización; tanto o tráfico destinado/xerado por él, coma o tráfico xerado/destinado a outros equipos. Ademais, simularemos que posúe unha IP pública e encargarase de facer NAT para permitir a saída a Internet dos equipos internos e tamén para facer accesible o servidor web interno.

Neste escenario aprenderemos a activar o enrutamento en equipos Linux, facer SNAT e DNAT, crear cadeas de usuario para xestionar certos tipos de tráfico, empregar novas condicións (*matches*) e rexistrar incidencias nun arquivo de log propio de iptables.

Hai que indicar que a infraestrutura escollida para este escenario non é a máis desexable ó estar un servidor público (accesible dende Internet) directamente conectado na Intranet; sen embargo, servirá para traballar novos conceptos sen introducir novas dificultades. No seguinte tema abordarase o estudo da ubicación dos servidores, as redes DMZs e as regras de filtrado recomendadas para este tipo de equipos.

Obxectivo

Configurar un firewall de rede para controlar todo o tráfico entrante/saínte de Internet mediante comandos iptables, facendo persistentes as regras.



Procedemento

Configurar netfilter mediante comandos iptables para permitir únicamente o tráfico autorizado nos intervalos temporais indicados.

As condicións a cumprir son:

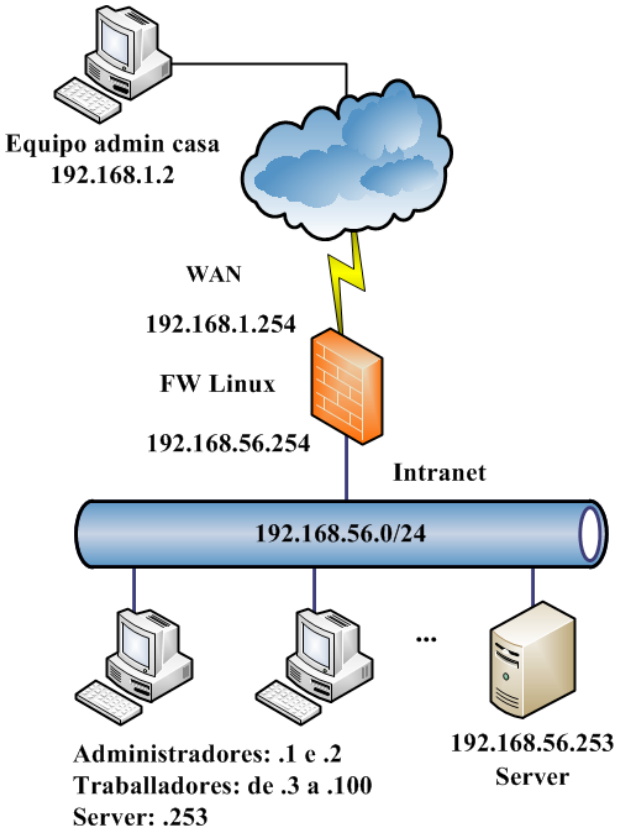
FW Linux:

- Pode facer resolución DNS usando os servidores 8.8.8.8 e 8.8.4.4.
- Pode acceder ós repositorios oficiais de Ubuntu para actualizacións/instalación de software.
- Administrable por ssh dende os equipos dos administradores (tanto dende a Intranet coma dende a casa). Os intentos de acceso por ssh dende equipos diferentes ós autorizados quedarán rexistrados.
- Todo o tráfico/equipos non autorizados será descartado.

Intranet:

- Os equipos da Intranet poden facer resolucións DNS usando os servidores 8.8.8.8 e 8.8.4.4 e navegar por Internet (tráfico http/https).
- Os traballadores únicamente poden saír a Internet de luns a venres das 07:00 ás 15:00 horas. Os administradores e o server non teñen restriccións horarias.
- O servizo web http/http correndo en server é accesible dende Internet.
- O server é administrable por ssh dende os equipos dos administradores (tanto dende a Intranet coma dende a casa).

7.1 Configuración do escenario (virtualbox)



Configuración do FW-Linux para o escenario 2D:

No caso de empregar Virtualbox para realizar o escenario 2D, o FW Linux corre nunha máquina virtual en VirtualBox coas seguintes características:

- Sistema Operativo Ubuntu Server 14.04. No caso de usar Ubuntu 16.04, revisar a explicación dada no apartado de Configuración do Escenario 2A.
- Memoria RAM: 512 MBytes
- Rede:
  - Apaptador de rede 1 configurado en modo ponte (bridge) para dar saída a Internet e recoñecido pola máquina virtual como `eth0`.
  - Adaptador de rede 2 configurado como sólo anfitrión (host-only) para conectarse á rede LAN e recoñecido pola máquina virtual como `eth1`.

Aínda que se traballa con dirección IP privadas tanto na interface LAN como na WAN, a IP asignada a WAN a consideraremos como unha IP pública, directamente accesible dende Internet. Ó estar `eth0` en modo bridge o FW terá unha configuración válida para a rede local real e `eth1` terá unha IP da sede sólo anfitrión. No exemplo, a configuración escollida é:

- IP`eth0` : 192.168.1.254/255.255.255.0
- Default gateway: 192.168.1.1
- IP `eth1`: 192.168.56.254
- Servidores DNS: 8.8.8.8 e 8.8.4.4

Para configurar a rede hai que editar o arquivo `/etc/network/interfaces`.

```
uadmin@fw-linux:~$ sudo nano /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
auto eth0
iface eth0 inet static
 address 192.168.1.254
 netmask 255.255.255.0
 gateway 192.168.1.1
 dns-nameservers 8.8.8.8 8.8.4.4
# The secondary network interface
auto eth1
iface eth1 inet static
 address 192.168.56.254
```

```
 netmask 255.255.255.0
```

Unha vez modificado o arquivo hai que aplicar esta configuración. En Ubuntu Server 14.04 pódese reiniciar a máquina ou aplicar os cambios sen reiniciar co comando:

```
uadmin@fw-linux:~$ sudo sh -c "ifdown eth0 && ifup eth0"
uadmin@fw-linux:~$ sudo sh -c "ifdown eth1 && ifup eth1"
```

Pode verificarse que a nova configuración foi aplicada:

```
uadmin@fw-linux:~$ sudo ifconfig -a ; netstat -nr
eth0      Link encap:Ethernet  direcciónHW 08:00:27:6c:34:33
          Direc. inet:192.168.1.254  Difus.:192.168.1.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:fe6c:3433/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:670 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:299 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:65510 (65.5 KB)  TX bytes:38408 (38.4 KB)
eth1      Link encap:Ethernet  direcciónHW 08:00:27:7e:cc:c5
          Direc. inet:192.168.56.254  Difus.:192.168.56.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:fe7e:ccc5/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:2 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:16 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:523 (523.0 B)  TX bytes:1296 (1.2 KB)
lo        Link encap:Bucle local
          Direc. inet:127.0.0.1  Másc:255.0.0.0
          Dirección inet6: ::1/128 Alcance:Anfitrión
          ACTIVO BUCLE FUNCIONANDO  MTU:65536  Métrica:1
          Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:0 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:0
          Bytes RX:0 (0.0 B)  TX bytes:0 (0.0 B)
Tabla de rutas IP del núcleo
Destino        Pasarela        Genmask         Indic   MSS Ventana irtt Interfaz
0.0.0.0         192.168.1.1     0.0.0.0         UG        0 0          0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0
192.168.56.0    0.0.0.0         255.255.255.0   U         0 0          0 eth1
uadmin@fw-linux:~$
```

Para instalar o servidor openssh e verificar que o servizo está levantado execútanse os seguintes comandos:

```
uadmin@fw-linux:~$ sudo apt-get update
uadmin@fw-linux:~$ sudo apt-get install openssh-server
uadmin@fw-linux:~$ sudo netstat -putan
Conexiones activas de Internet (servidores y establecidos)
Proto  Recib Enviad Dirección local  Dirección remota   Estado  PID/Program name
tcp        0          0 0.0.0.0:22       0.0.0.0:*     ESCUCHAR    1566/sshd
```

Configuración do Server para o escenario 2D:

Para representar o papel do Server deste escenario pode usarse o server do escenario 2C cambiando a IP da 192.168.1.254 á 192.168.56.253 ou configurar un novo servidor instalando os servidores ssh e Apache e despois configurando a rede para adaptala á rede do escenario.

Como o server do Escenario 2C xa ten instalado o servidor Apache correndo http/https únicamente habería que configurar correctamente a rede:

1. En primeiro lugar, hai que cambiar o modo de rede de ponte (bridge) a rede sólo anfitrión (host-only). A rede sólo anfitrión debe ser a mesma que se usou para o adaptador de rede 2 de FW Linux.
2. En segundo lugar, hai que modificar a IP e o gateway da máquina, xa que agora o server está na rede 192.168.56.0/24 e usará a FW Linux como router:

```
uadmin@server:~$ sudo nano /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
auto eth0
iface eth0 inet static
 address 192.168.56.253
 netmask 255.255.255.0
 gateway 192.168.56.254
 dns-nameservers 8.8.8.8 8.8.4.4
```

Unha vez modificado o arquivo hai que aplicar esta configuración. En Ubuntu Server 14.04 pódese reiniciar a máquina ou aplicar os cambios sen reiniciar co comando:

```
uadmin@server:~$ sudo sh -c "ifdown eth0 && ifup eth0"
```

Pode verificarse que a nova configuración foi aplicada:

```
uadmin@server:~$ sudo ifconfig -a ; netstat -nr
eth0      Link encap:Ethernet  direcciónHW 08:00:27:c1:56:ad
          Direc. inet:192.168.56.253  Difus.:192.168.56.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:fec1:56ad/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:5297 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:3176 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:6383905 (6.3 MB)  TX bytes:265264 (265.2 KB)
lo        Link encap:Bucle local
          Direc. inet:127.0.0.1  Másc:255.0.0.0
          Dirección inet6: ::1/128 Alcance:Anfitrión
          ACTIVO BUCLE FUNCIONANDO  MTU:65536  Métrica:1
          Paquetes RX:32 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:32 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:0
          Bytes RX:2480 (2.4 KB)  TX bytes:2480 (2.4 KB)
Tabla de rutas IP del núcleo
Destino         Pasarela        Genmask         Indic   MSS Ventana irtt nterfaz
0.0.0.0         192.168.56.254  0.0.0.0         UG        0 0          0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0
uadmin@server:~$
```

Por último, volvemos a poñer o servizo ssh a escoitar no porto tcp/22:

- En `/etc/ssh/sshd_config` cambiar a directiva `Port 22000` por `Port 22`
- Reiniciar o servizo: **`$ sudo service ssh restart`**

Apuntes para Ubuntu 16.04

No caso de usar Ubuntu 16.04, revisar a explicación dada no apartado de Configuración do Escenario 2A.

Configuración dos equipos administradores para o escenario 2D:

O equipo do administrador 1 terá a IP 192.168.56.1/24 e trátase do equipo real (host anfitrión) onde corre Virtualbox. A IP deste equipo pode asignarse a través do **`Menú Archivo --> Preferencias --> Red --> Redes sólo-anfitrión`** en Virtualbox:

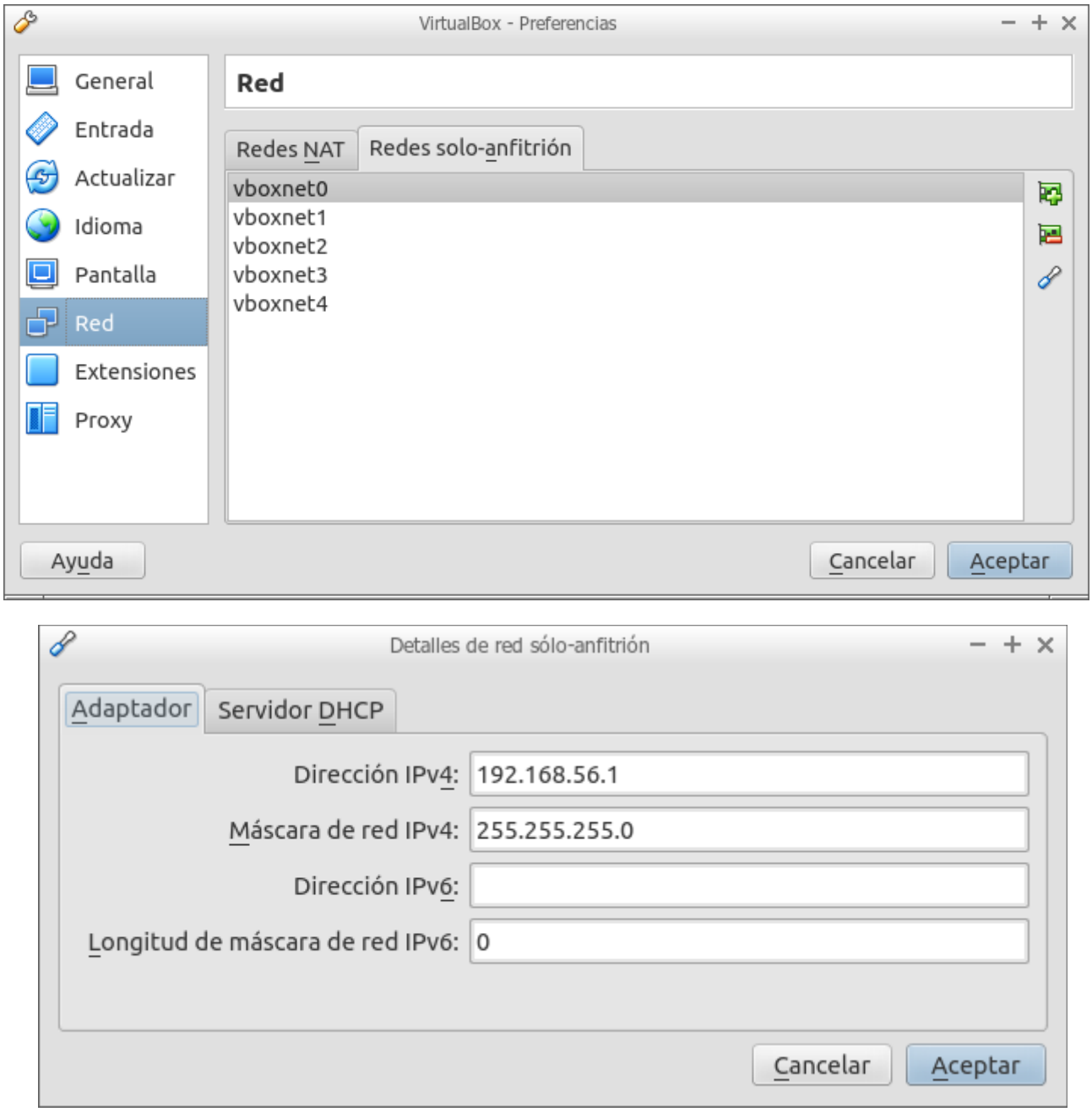


Fig. Configuración da rede sólo-anfitrión

Para o equipo do administrador 2 e o resto de equipos da LAN pode usarse calquera máquina virtual Linux ou Windows con contorno gráfico e cun adaptador de rede na rede sólo-anfitrión do escenario e configurado cunha IP da rede 192.168.56.0/24 e con pasarela predeterminada FW-Linux (192.168.56.254).

Configuración do equipo admin casa para o escenario 2D:

Ó estar o adaptador 1 de FW Linux en modo ponte (bridge) e como supoñemos que está conectado a Internet cunha 'IP pública'; dende o punto de vista de FW Linux, a IP da NIC real do equipo real está situada en Internet. Polo tanto, o equipo admin casa será o equipo real e únicamente precisamos

ter un navegador web e un cliente ssh para verificar o funcionamento de FW Linux dende Internet unha vez rematada a configuración.

Por suposto, tamén se pode usar outro equipo (real ou virtual) configurado cunha IP válida da rede á que se conecta a interface WAN de FW Linux.

7.3 Enrutamento e tráfico de FW Linux

Activar enrutamento

Para facer que un equipo Linux funcione como un router hay que asignar á variable do sistema **`net.ipv4.ip_forward`** o valor 1. Para asignarlle o valor 1 e facer o cambio persistente, resistindo reinicios da máquina, editaremos o arquivo de configuración `/etc/sysctl.conf` e descomentados a liña correspodente a `net.ipv4.ip_forward`:

```
uadmin@fw-linux:~$ sudo nano /etc/sysctl.conf
...
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
...
```

Unha vez feito o cambio e gardado o cambio, procedemos a facer efectivo o novo valor:

```
uadmin@fw-linux:~$ sudo sysctl -p
net.ipv4.ip_forward = 1
uadmin@fw-linux:~$
```

Un punto a ter claro é que a partir deste momento, o equipo FW Linux é un router; é dicir, pode reenviar paquetes. Trátase dun maneira moi rápida e sinxela de ter un router para prácticas na clase. Non obstante, reenviar paquetes non quere dicir facer NAT, para iso hai que configurar netfilter.

Tráfico de FW Linux

Neste punto abordaremos o tráfico propio de FW Linux; é dicir, tráfico xerado/destinado por/hacia aplicacións correndo na máquina. FW Linux é directamente accesible dende a Intranet e dende Internet, polo que non é preciso facer ningún tipo de NAT, sendo as cadeas INPUT e OUTPUT nas que teremos que centrarnos. Xestionaremos o tráfico de FW Linux usando cadeas de usuario para ver o seu uso.

O punto de partida é unha táboa de filtrado baleira:

```
uadmin@fw-linux:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination
```

```
Chain FORWARD (policy ACCEPT)
num target prot opt source destination
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
```

Creamos as regras para aceptar todos os paquetes con destino/orixe FW Linux que pertencen a unha conexión xa autorizada e establecida:

```
uadmin@fw-linux:~$ sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED
-j ACCEPT
uadmin@fw-linux:~$ sudo iptables -A OUTPUT -m state --state ESTABLISHED,RELATED
-j ACCEPT
```

Creamos varias cadeas de usuario para xestionar o tráfico mediante o comando `iptables -N`:

- SSH para o tráfico de administración por ssh.
- DNS para a resolución DNS.
- REPOS para o acceso ós repositorios.

```
uadmin@fw-linux:~$ sudo iptables -N SSH
uadmin@fw-linux:~$ sudo iptables -N DNS
uadmin@fw-linux:~$ sudo iptables -N REPOS
```

Facendo un listado das regras veremos como aparecen na táboa filter as novas cadeas:

```
uadmin@fw-linux:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
Chain FORWARD (policy ACCEPT)
num target prot opt source destination
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
Chain DNS (0 references)
num target prot opt source destination
Chain REPOS (0 references)
num target prot opt source destination
Chain SSH (0 references)
num target prot opt source destination
uadmin@fw-linux:~$
```

Tráfico SSH de administración de FW Linux

Creamos unha regra en INPUT para que o tráfico ssh de administración sexa redirixido á cadea SSH:

```
uadmin@fw-linux:~$ sudo iptables -A INPUT -p tcp --dport 22 -m state --state NEW -j SSH
uadmin@fw-linux:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 SSH tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
Chain FORWARD (policy ACCEPT)
num target prot opt source destination
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
Chain DNS (0 references)
num target prot opt source destination
Chain REPOS (0 references)
num target prot opt source destination
Chain SSH (1 references)
num target prot opt source destination
uadmin@fw-linux:~$
```

Neste momento todo o tráfico de ssh de administración de FW Linux é enviado á cadea SSH; e é ahí, onde crearemos as regras para permitir ós equipos dos admins e bloquear ó resto:

```
uadmin@fw-linux:~$ sudo iptables -A SSH -s 192.168.1.2 -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -A SSH -s 192.168.56.1 -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -A SSH -s 192.168.56.2 -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 SSH tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
Chain FORWARD (policy ACCEPT)
num target prot opt source destination
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
Chain DNS (0 references)
```

```
num target prot opt source destination
Chain REPOS (0 references)
num target prot opt source destination
Chain SSH (1 references)
num target prot opt source destination
1 ACCEPT all -- 192.168.1.2 0.0.0.0/0
2 ACCEPT all -- 192.168.56.1 0.0.0.0/0
3 ACCEPT all -- 192.168.56.2 0.0.0.0/0
uadmin@fw-linux:~$
```

Para rexistrar os intentos de conexión por ssh dende equipos non autorizados, crearemos unha regra a tal efecto na cadea SSH usando a acción **`-j LOG`** e **`--log-prefix "iptables: SSH Bloqueo"`** para facilitar a súa interpretación. Un punto moi importante a ter en conta é que acción de rexistro LOG non é unha acción definitiva; xa que, o paquete fará que se escriba unha entrada no arquivo de log e seguirá procesándose no ruleset do firewall.

```
uadmin@fw-linux:~$ sudo iptables -A SSH -j LOG --log-prefix "iptables: SSH
Bloqueo "
uadmin@fw-linux:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 SSH tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
Chain FORWARD (policy ACCEPT)
num target prot opt source destination
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
Chain DNS (0 references)
num target prot opt source destination
Chain REPOS (0 references)
num target prot opt source destination
Chain SSH (1 references)
num target prot opt source destination
1 ACCEPT all -- 192.168.1.2 0.0.0.0/0
2 ACCEPT all -- 192.168.56.1 0.0.0.0/0
3 ACCEPT all -- 192.168.56.2 0.0.0.0/0
4 LOG all -- 0.0.0.0/0 0.0.0.0/0 LOG flags 0 level 4 prefix "iptables: SSH
Bloqueo "
uadmin@fw-linux:~$
```

O procesamento dos intentos de conexión por ssh a FW Linux é o seguinte:

1. Cando se intenta conectar por ssh con FW Linux o paquete chega a regra#2 da cadea INPUT e é desviada á cadea definida polo usuario SSH.
2. Unha vez na cadea SSH compárase a IP orixe e se coincide con algunha das IPs dos admins (regra#1, regra#2 e regra#3) a conexión é aceptada.
3. No caso de non proceder dun dos equipos dos admins, o paquete chega a regra#4 o que provoca que se rexistre a incidencia no arquivo de logs e despois prosigue a súa marcha chegando ó final da cadea.
4. Cando o paquete chega ó final da cadea SSH aplícaselle a política por defecto que é RETURN, voltando a cadea principal INPUT onde seguirá procesándose.

Hai que indicar que as cadeas definidas polos usuarios teñen unha política implícita de RETURN que non se pode cambiar. Os paquetes que chegan ó final dunha cadea de usuario retornan á cadea de orixe de onde proceden e continúase o seu procesamento. No noso caso, os paquetes non procedentes de equipos admins queremos que sexan eliminados e podemos facelo de varias formas:

- Unha posible maneira de eliminalos é crear unha regra ó final da cadea SSH a tal efecto: $ `sudo iptables -A SSH -j DROP`
- Outra maneira de eliminalos é deixar que volten á cadea INPUT e matalos ben cunha regra de descarte ou definindo a política a DROP. Se optamos por esta solución hai que ter moi en conta que o paquete segue procesándose na cadea INPUT e se cumpre con algunha regra que o acepte, o paquete será aceptado e non chegará a descartarse.

Nos optaremos por descartalos na cadea SSH:

```
uadmin@fw-linux:~$ sudo iptables -A SSH -j DROP
uadmin@fw-linux:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 SSH tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
Chain FORWARD (policy ACCEPT)
num target prot opt source destination
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
Chain DNS (0 references)
num target prot opt source destination
Chain REPOS (0 references)
num target prot opt source destination
```

```
Chain SSH (1 references)
num target prot opt source destination
1 ACCEPT all -- 192.168.1.2 0.0.0.0/0
2 ACCEPT all -- 192.168.56.1 0.0.0.0/0
3 ACCEPT all -- 192.168.56.2 0.0.0.0/0
4 LOG all -- 0.0.0.0/0 0.0.0.0/0 LOG flags 0 level 4 prefix "iptables: SSH Bloqueo "
5 DROP all -- 0.0.0.0/0 0.0.0.0/0
uadmin@fw-linux:~$
```

Resolución DNS de FW Linux

Repetimos o proceso seguido no punto anterior:

1. Creamos unha regra na cadea OUTPUT para enviar os paquetes dns á cadea de usuario DNS.
2. Aceptamos as consultas enviadas ós servidores 8.8.8.8 e .8.8.4.4
3. Denegamos o resto.

```
uadmin@fw-linux:~$ sudo iptables -A OUTPUT -p udp --dport 53 -m state --state NEW -j DNS
uadmin@fw-linux:~$ sudo iptables -A DNS -d 8.8.8.8 -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -A DNS -d 8.8.4.4 -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -A DNS -j DROP
uadmin@fw-linux:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 SSH tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
Chain FORWARD (policy ACCEPT)
num target prot opt source destination
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2  DNS   udp -- 0.0.0.0/0        0.0.0.0/0    udp dpt:53 state NEW
Chain DNS (1 references)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 8.8.8.8
2 ACCEPT all -- 0.0.0.0/0 8.8.4.4
3 DROP all -- 0.0.0.0/0 0.0.0.0/0
Chain REPOS (0 references)
```

```
num target prot opt source destination
Chain SSH (1 references)
num target prot opt source destination
1 ACCEPT all -- 192.168.1.2 0.0.0.0/0
2 ACCEPT all -- 192.168.56.1 0.0.0.0/0
3 ACCEPT all -- 192.168.56.2 0.0.0.0/0
4 LOG all -- 0.0.0.0/0 0.0.0.0/0 LOG flags 0 level 4 prefix "iptables: SSH
Bloqueo "
5 DROP all -- 0.0.0.0/0 0.0.0.0/0
uadmin@fw-linux:~$
```

Acceso repositorios Ubuntu

Revisando o arquivo `/etc/apt/sources.list` podemos comprobar cales son os repositorios de paquetes que hai que autorizar:

```
uadmin@fw-linux:~$ cat /etc/apt/sources.list
# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.
deb http://es.archive.ubuntu.com/ubuntu/ trusty main restricted
deb-src http://es.archive.ubuntu.com/ubuntu/ trusty main restricted
## Major bug fix updates produced after the final release of the
## distribution.
deb http://es.archive.ubuntu.com/ubuntu/ trusty-updates main restricted
deb-src http://es.archive.ubuntu.com/ubuntu/ trusty-updates main restricted
## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team. Also, please note that software in universe WILL NOT receive any
## review or updates from the Ubuntu security team.
deb http://es.archive.ubuntu.com/ubuntu/ trusty universe
deb-src http://es.archive.ubuntu.com/ubuntu/ trusty universe
deb http://es.archive.ubuntu.com/ubuntu/ trusty-updates universe
deb-src http://es.archive.ubuntu.com/ubuntu/ trusty-updates universe
## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team, and may not be under a free licence. Please satisfy yourself as to
## your rights to use the software. Also, please note that software in
## multiverse WILL NOT receive any review or updates from the Ubuntu
## security team.
deb http://es.archive.ubuntu.com/ubuntu/ trusty multiverse
deb-src http://es.archive.ubuntu.com/ubuntu/ trusty multiverse
deb http://es.archive.ubuntu.com/ubuntu/ trusty-updates multiverse
deb-src http://es.archive.ubuntu.com/ubuntu/ trusty-updates multiverse
```

```
## N.B. software from this repository may not have been tested as
## extensively as that contained in the main release, although it includes
## newer versions of some applications which may provide useful features.
## Also, please note that software in backports WILL NOT receive any review
## or updates from the Ubuntu security team.
deb http://es.archive.ubuntu.com/ubuntu/ trusty-backports main restricted
universe multiverse
deb-src http://es.archive.ubuntu.com/ubuntu/ trusty-backports main restricted
universe multiverse
deb http://security.ubuntu.com/ubuntu trusty-security main restricted
deb-src http://security.ubuntu.com/ubuntu trusty-security main restricted
deb http://security.ubuntu.com/ubuntu trusty-security universe
deb-src http://security.ubuntu.com/ubuntu trusty-security universe
deb http://security.ubuntu.com/ubuntu trusty-security multiverse
deb-src http://security.ubuntu.com/ubuntu trusty-security multiverse
## Uncomment the following two lines to add software from Canonical's
## 'partner' repository.
## This software is not part of Ubuntu, but is offered by Canonical and the
## respective vendors as a service to Ubuntu users.
deb http://archive.canonical.com/ubuntu trusty partner
deb-src http://archive.canonical.com/ubuntu trusty partner
## Uncomment the following two lines to add software from Ubuntu's
## 'extras' repository.
## This software is not part of Ubuntu, but is offered by third-party
## developers who want to ship their latest software.
deb http://extras.ubuntu.com/ubuntu trusty main
deb-src http://extras.ubuntu.com/ubuntu trusty main
uadmin@fw-linux:~$
```

Polo tanto haberá que autorizar o acceso por http ós equipos es.archive.ubuntu.com,security.ubuntu.com,archive.canonical.com e extras.ubuntu.com:

```
uadmin@fw-linux:~$ sudo iptables -A OUTPUT -p tcp --dport 80 -m state --state
NEW -j REPOS
uadmin@fw-linux:~$ sudo iptables -A REPOS -d es.archive.ubuntu.com -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -A REPOS -d security.ubuntu.com -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -A REPOS -d archive.canonical.com -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -A REPOS -d extras.ubuntu.com -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
```

```
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 SSH tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
Chain FORWARD (policy ACCEPT)
num target prot opt source destination
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 DNS udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:53 state NEW
3  REPOS  tcp  -- 0.0.0.0/0    0.0.0.0/0    tcp dpt:80 state NEW
Chain DNS (1 references)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 8.8.8.8
2 ACCEPT all -- 0.0.0.0/0 8.8.4.4
3 DROP all -- 0.0.0.0/0 0.0.0.0/0
Chain REPOS (1 references)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 91.189.92.200
2 ACCEPT all -- 0.0.0.0/0 91.189.92.201
3 ACCEPT all -- 0.0.0.0/0 91.189.88.153
4 ACCEPT all -- 0.0.0.0/0 91.189.88.152
5 ACCEPT all -- 0.0.0.0/0 91.189.91.15
6 ACCEPT all -- 0.0.0.0/0 91.189.91.24
7 ACCEPT all -- 0.0.0.0/0 91.189.88.149
8 ACCEPT all -- 0.0.0.0/0 91.189.91.14
9 ACCEPT all -- 0.0.0.0/0 91.189.91.23
10 ACCEPT all -- 0.0.0.0/0 91.189.91.13
11 ACCEPT all -- 0.0.0.0/0 91.189.91.13
12 ACCEPT all -- 0.0.0.0/0 91.189.91.15
13 ACCEPT all -- 0.0.0.0/0 91.189.92.200
14 ACCEPT all -- 0.0.0.0/0 91.189.91.24
15 ACCEPT all -- 0.0.0.0/0 91.189.91.23
16 ACCEPT all -- 0.0.0.0/0 91.189.88.152
17 ACCEPT all -- 0.0.0.0/0 91.189.91.14
18 ACCEPT all -- 0.0.0.0/0 91.189.92.201
19 ACCEPT all -- 0.0.0.0/0 91.189.88.149
20 ACCEPT all -- 0.0.0.0/0 91.189.92.191
21 ACCEPT all -- 0.0.0.0/0 91.189.92.150
22 ACCEPT all -- 0.0.0.0/0 91.189.92.152
```

```
Chain SSH (1 references)
num target prot opt source destination
1 ACCEPT all -- 192.168.1.2 0.0.0.0/0
2 ACCEPT all -- 192.168.56.1 0.0.0.0/0
3 ACCEPT all -- 192.168.56.2 0.0.0.0/0
4 LOG all -- 0.0.0.0/0 0.0.0.0/0 LOG flags 0 level 4 prefix "iptables: SSH
Bloqueo "
5 DROP all -- 0.0.0.0/0 0.0.0.0/0
uadmin@fw-linux:~$
```

Pode observarse que aparecen IPs repetidas, producto da asignación de nomes-IPs feita polos encargados dos repositorios. Revisando e eliminando esas entradas duplicadas con `iptables -D` teríamos unha cadea REPOS máis optimizada. Igual que nos casos anteriores, podemos optar por eliminar o tráfico hacia repositorios non autorizados nesta cadea ou coa política por defecto. Para ver a diferencia coas outras solucións, optamos por denegar o tráfico coa política por defecto que estableceremos no seguinte apartado.

Denegar por defecto o tráfico orixe/destino de FW Linux

Para rematar definimos a política de denegar por defecto para as cadeas INPUT e OUPUT:

```
uadmin@fw-linux:~$sudo iptables -P INPUT DROP
uadmin@fw-linux:~$sudo iptables -P OUTPUT DROP
uadmin@fw-linux:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 SSH tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
Chain FORWARD (policy ACCEPT)
num target prot opt source destination
Chain OUTPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 DNS udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:53 state NEW
3 REPOS tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:80 state NEW
Chain DNS (1 references)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 8.8.8.8
2 ACCEPT all -- 0.0.0.0/0 8.8.4.4
3 DROP all -- 0.0.0.0/0 0.0.0.0/0
Chain REPOS (1 references)
```

```
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 91.189.92.200
2 ACCEPT all -- 0.0.0.0/0 91.189.92.201
3 ACCEPT all -- 0.0.0.0/0 91.189.88.153
4 ACCEPT all -- 0.0.0.0/0 91.189.88.152
5 ACCEPT all -- 0.0.0.0/0 91.189.91.15
6 ACCEPT all -- 0.0.0.0/0 91.189.91.24
7 ACCEPT all -- 0.0.0.0/0 91.189.88.149
8 ACCEPT all -- 0.0.0.0/0 91.189.91.14
9 ACCEPT all -- 0.0.0.0/0 91.189.91.23
10 ACCEPT all -- 0.0.0.0/0 91.189.91.13
11 ACCEPT all -- 0.0.0.0/0 91.189.92.191
12 ACCEPT all -- 0.0.0.0/0 91.189.92.150
13 ACCEPT all -- 0.0.0.0/0 91.189.92.152
Chain SSH (1 references)
num target prot opt source destination
1 ACCEPT all -- 192.168.1.2 0.0.0.0/0
2 ACCEPT all -- 192.168.56.1 0.0.0.0/0
3 ACCEPT all -- 192.168.56.2 0.0.0.0/0
4 LOG all -- 0.0.0.0/0 0.0.0.0/0 LOG flags 0 level 4 prefix "iptables: SSH
Bloqueo "
5 DROP all -- 0.0.0.0/0 0.0.0.0/0
uadmin@fw-linux:~$
```

7.4 Tráfico da Intranet

Tráfico orixinado nos equipos da Intranet

Antes de comezar a facer regras imos analizar dous puntos relativos ó tráfico iniciado dende os equipos da Intranet:

- O server é administrable por ssh dende os equipos dos administradores da Intranet: a comunicación entre os equipos dos admins da Intranet e o server ocorre sen pasar por FW Linux; xa que, trátase de tráfico dentro da mesma rede. Esto tradúcese en que non hai que facer ninguna regra en FW Linux para controlar o acceso por ssh a server dende a Intranet. No caso de querer controlar este tipo de accesos, habería que configurar o firewall no propio server.
- Os equipos da Intranet teñen IPs privadas e saen a Internet para facer resolucións dns e visitar sitios web; polo que, haberá que facer regras NAT e regras de filtrado:
    - Regras SNAT para cambiar a IP privada orixe dos paquetes pola IP pública de FW Linux.

- Regras de filtrado para autorizar os paquetes.
- SNAT faise na cadea POSTROUTING polo que na cadea FORWARD haberá que autorizar os paquetes orixinais (que aínda teñen como IP orixe a IP privada).

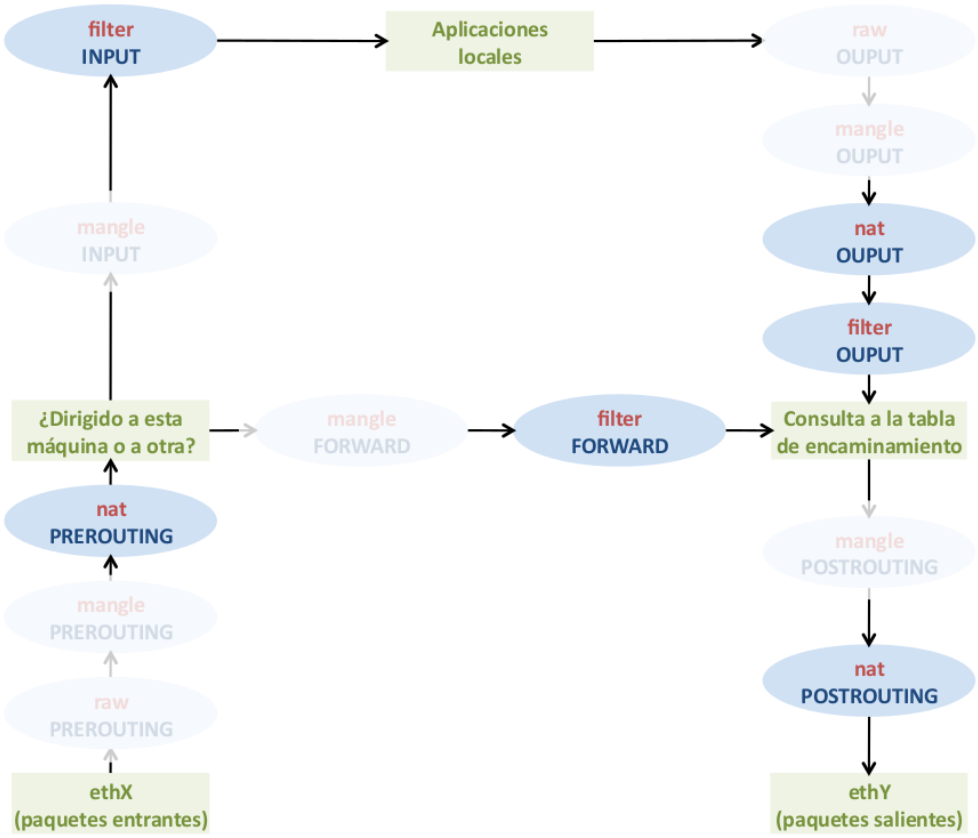


Fig. Táboas e cadeas predeterminadas máis usadas en netfilter. GSyC (CC BY-SA 2.1 ES)

Comezamos creando as regras para aceptar todos os paquetes a enrutar por FW Linux que pertencen a unha conexión xa autorizada e establecida e a establecer a política da cadea FORWARD a DROP:

```
uadmin@fw-linux:~$ sudo iptables -A FORWARD -m state --state ESTABLISHED,RELATED
-j ACCEPT
uadmin@fw-linux:~$ sudo iptables -P FORWARD DROP
uadmin@fw-linux:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 SSH tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
Chain FORWARD (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
Chain OUTPUT (policy DROP)
num target prot opt source destination
```

```
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 DNS udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:53 state NEW
3 REPOS tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:80 state NEW
...
```

Resolución DNS

Para continuar practicando o uso das cadeas definidas polos usuarios, crearemos unha cadea chamada DNS_INTRA onde enviaremos as consultas dns. Dentro de DNS_INTRA permitiremos os paquetes en base á súa orixe e intervalo horario. Para controlar o tráfico dns dos equipos dos traballadores empregaremos novas condicións:

- `-m iprange`: permite especificar un rango de direccións IP
  - `--src-range from[-to]` --> IPs orixe.
  - `--dst-range from[-to]` --> IPs destino.
- `-m time`: permite controlar se o paquete chega dentro dun intervalo temporal
  - `--timestart hh:mm[:ss]` e `--timestop hh:mm[:ss]` --> para definir controis en base á hora
  - `--weekdays day[,day...]` --> para definir controis en base ó día da semana.
  - `--kerneltz` --> para usar a hora do sistema como referencia

```
uadmin@fw-linux:~$ sudo iptables -N DNS_INTRA
uadmin@fw-linux:~$ sudo iptables -A FORWARD -p udp --dport 53 -d 8.8.8.8 -m state --state NEW -j DNS_INTRA
uadmin@fw-linux:~$ sudo iptables -A FORWARD -p udp --dport 53 -d 8.8.4.4 -m state --state NEW -j DNS_INTRA
uadmin@fw-linux:~$ sudo iptables -A DNS_INTRA -s 192.168.56.1 -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -A DNS_INTRA -s 192.168.56.2 -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -A DNS_INTRA -s 192.168.56.253 -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -A DNS_INTRA -m iprange --src-range 192.168.56.3-192.168.56.100 -m time --timestart 07:00:00 --timestop 15:00:00 --weekdays Mon,Tue,Wed,Thu,Fri --kerneltz -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -A DNS_INTRA -j DROP
uadmin@fw-linux:~$ sudo iptables -L -n --line-numbers

Chain INPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 SSH tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
Chain FORWARD (policy DROP)
num target prot opt source destination
```

```
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2  DNS_INTRA  udp  -- 0.0.0.0/0     8.8.8.8     udp dpt:53state NEW
3  DNS_INTRA  udp -- 0.0.0.0/0     8.8.4.4    udp dpt:53state NEW
Chain OUTPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 DNS udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:53 state NEW
3 REPOS tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:80 state NEW
Chain DNS (1 references)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 8.8.8.8
2 ACCEPT all -- 0.0.0.0/0 8.8.4.4
3 DROP all -- 0.0.0.0/0 0.0.0.0/0
Chain DNS_INTRA (2 references)
num target prot opt source destination
1 ACCEPT all -- 192.168.56.1 0.0.0.0/0
2 ACCEPT all -- 192.168.56.2 0.0.0.0/0
3 ACCEPT all -- 192.168.56.253 0.0.0.0/0
4 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 source IP range 192.168.56.3-192.168.56.100
TIME from 07:00:00 to 15:00:00 on Mon,Tue,Wed,Thu,Fri
5 DROP all -- 0.0.0.0/0 0.0.0.0/0
...
```

Tráfico Web

A continuación controlaremos o tráfico web iniciado dende os clientes da Intranet. Seguiremos traballando con cadeas de usuario e empregaremos a opción `-i` para indicar que os paquetes a enviar á cadea WEB_INTRA teñen que entrar no FW Linux pola interface `eth1`; é dicir, proceder de equipos da Intranet.

```
uadmin@fw-linux:~$ sudo iptables -N WEB_INTRA
uadmin@fw-linux:~$ sudo iptables -A FORWARD -i eth1 -p tcp -m multiport --dports
80,443 -m state --state NEW -j WEB_INTRA
uadmin@fw-linux:~$ sudo iptables -A WEB_INTRA -s 192.168.56.1 -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -A WEB_INTRA -s 192.168.56.2 -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -A WEB_INTRA -s 192.168.56.253 -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -A WEB_INTRA -m iprange --src-range
192.168.56.3-192.168.56.100 -m time --timestart 07:00:00 --timestop 15:00:00
--weekdays Mon,Tue,Wed,Thu,Fri --kerneltz -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -A WEB_INTRA -j DROP
```

```
uadmin@fw-linux:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 SSH tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
Chain FORWARD (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 DNS_INTRA udp -- 0.0.0.0/0 8.8.8.8 udp dpt:53 state NEW
3 DNS_INTRA udp -- 0.0.0.0/0 8.8.4.4 udp dpt:53 state NEW
4  WEB_INTRA  tcp  -- 0.0.0.0/0    0.0.0.0/0    multiport dports 80,443 state
NEW
Chain OUTPUT (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 DNS udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:53 state NEW
3 REPOS tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:80 state NEW
...
Chain WEB_INTRA (1 references)
num target prot opt source destination
1 ACCEPT all -- 192.168.56.1 0.0.0.0/0
2 ACCEPT all -- 192.168.56.2 0.0.0.0/0
3 ACCEPT all -- 192.168.56.253 0.0.0.0/0
4 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 source IP range 192.168.56.3-192.168.56.100
TIME from 07:00:00 to 15:00:00 on Mon,Tue,Wed,Thu,Fri
5 DROP all -- 0.0.0.0/0 0.0.0.0/0
```

No listado das regras non aparece a interface na regra#4 da cadea FORWARD. Se facemos o listado coa opción `-v` si aparece:

```
uadmin@fw-linux:~$ sudo iptables -L FORWARD -v -n --line-numbers
Chain FORWARD (policy DROP 0 packets, 0 bytes)
num pkts bytes target prot opt in out source destination
1 0 0 ACCEPT all -- * * 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 0 0 DNS_INTRA udp -- * * 0.0.0.0/0 8.8.8.8 udp dpt:53 state NEW
3 0 0 DNS_INTRA udp -- * * 0.0.0.0/0 8.8.4.4 udp dpt:53 state NEW
4   0   0 WEB_INTRA  tcp  -- eth1 *   0.0.0.0/0    0.0.0.0/0   multiport dports
80,443 state NEW
```

Hai que sinalar que especificar interfaces nunha regra usando `-i` ou `-o` ten certas limitacións:

- `-i` pode usarse únicamente nas cadeas INPUT, FORWARD e PREROUTING.
- `-o` pode usarse únicamente nas cadeas FORWARD, OUTPUT e POSTROUTING.

Apuntes para Ubuntu 16.04

No caso de usar Ubuntu 16.04, revisar o nome asignado ás interfaces de rede; xa que pode que non sexan eth0, eth1, ...

Regras NAT para tráfico iniciado na Intranet

É moi importante lembrar que unha cousa é transformar os paquetes (NAT) e outra é autorizalos/denegalos (filtrado). Unha regra NAT sen a súa regra de filtrado correspondente (e viceversa) non é capaz de facer que os paquetes saian a Internet. Como xa creamos as regras de filtrado que controlan o tráfico con orixe os equipos da Intranet, únicamente falta configurar o SNAT na cadea POSTROUTING para cambiar a IP orixe privada pola IP pública de FW Linux. Por comodidade, especificamos na regra SNAT coma orixe toda a Intranet sen preocuparnos de incluír máis equipos dos indicados no enunciado; xa que, o tráfico iniciado por eses equipos será bloqueado nas regras de filtrado e xa non chegará a cadea POSTROUTING.

```
uadmin@fw-linux:~$ sudo iptables -t nat -A POSTROUTING -s 192.168.56.0/24 -j
SNAT --to-source 192.168.1.254
uadmin@fw-linux:~$ sudo iptables -t nat -L -n --line-numbers
Chain PREROUTING (policy ACCEPT)
num target prot opt source destination
Chain INPUT (policy ACCEPT)
num target prot opt source destination
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
Chain POSTROUTING (policy ACCEPT)
num target prot opt source destination
1 SNAT   all -- 192.168.56.0/24    0.0.0.0/0 to:192.168.1.254
uadmin@fw-linux:~$
```

7.5 Tráfico dende Internet

Servidor Web público

Server ten unha dirección IP privada; e polo tanto, inalcanzable dende Internet. Para poder acceder dende Internet ós servizos http/https correndo nel haberá que:

- Facer DNAT (port forwarding). O obxectivo e facer un mapeado entre a IP pública de FW Linux-portos web con IP privada de server-portos web:

IP pública	Porto público	IP privada	Porto privado
192.168.1.254	tcp/80	192.168.56.253	tcp/80
192.168.1.254	tcp/443	192.168.56.253	tcp/443

- Crear unha regra de filtrado na cadea FORWARD para autorizar os paquetes.
- Como DNAT faise na cadea PREROUTING, a regra de filtrado ten que autorizar os paquetes xa transformados; é dicir, con IP destino a IP privada do server.

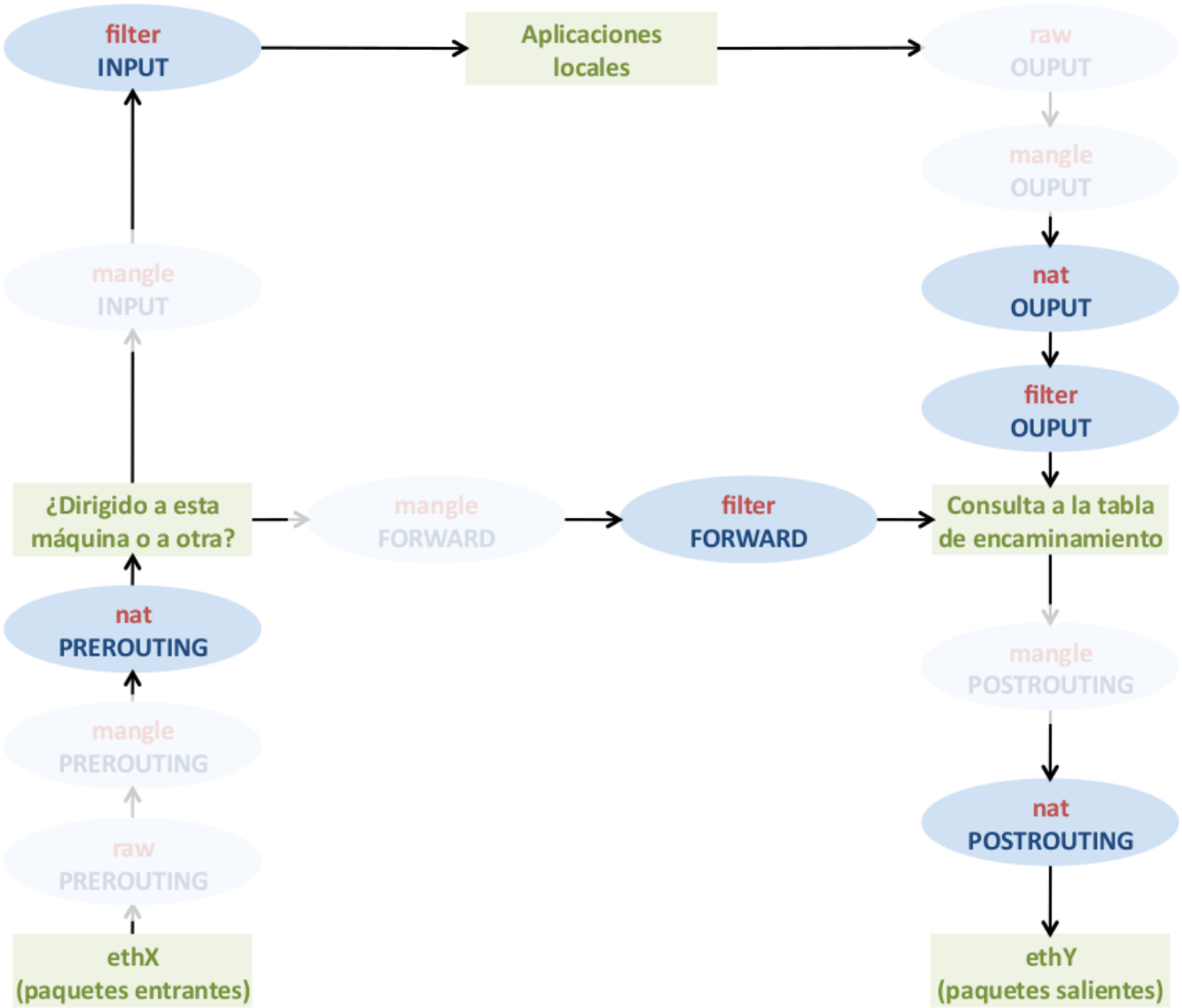


Fig. Táboas e cadeas predeterminadas máis usadas en netfilter. GSyC (CC BY-SA 2.1 ES)

Procedemos a crear a regra DNAT:

```
uadmin@fw-linux:~$ sudo iptables -t nat -A PREROUTING -p tcp -d 192.168.1.254 -m multiport --dports 80,443 -j DNAT --to-destination 192.168.56.253
uadmin@fw-linux:~$ sudo iptables -t nat -L -n --line-numbers
Chain PREROUTING (policy ACCEPT)
num target prot opt source destination
1  DNAT       tcp  --  0.0.0.0/0          192.168.1.254  multiport  dports  80,443 to:192.168.56.253
Chain INPUT (policy ACCEPT)
num target prot opt source destination
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
Chain POSTROUTING (policy ACCEPT)
num target prot opt source destination
1 SNAT all -- 192.168.56.0/24 0.0.0.0/0 to:192.168.1.254
uadmin@fw-linux:~$
```

A continuación creamos a regra de filtrado, onde a modo de exemplo empregaremos as opcións `-i` e `-o` para indicar a interface de entrada e saída do paquete:

```
uadmin@fw-linux:~$ sudo iptables -A FORWARD -i eth0 -o eth1 -p tcp -d 192.168.56.253 -m multiport --dports 80,443 -m state --state NEW -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -L FORWARD -v -n --line-numbers
Chain FORWARD (policy DROP 0 packets, 0 bytes)
num pkts bytes target prot opt in out source destination
1 9 1500 ACCEPT all -- * * 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 3 199 DNS_INTRA udp -- * * 0.0.0.0/0 8.8.8.8 udp dpt:53 state NEW
3 6 398 DNS_INTRA udp -- * * 0.0.0.0/0 8.8.4.4 udp dpt:53 state NEW
4 0 0 WEB_INTRA tcp -- eth1 * 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
5  0  0 ACCEPT  tcp  -- eth0 eth1   0.0.0.0/0    192.168.56.253    multiport dports 80,443 state NEW
```

É interesante fixarse nas regras 4 e 5 onde se traballa con paquetes web. Tal e como fixemos as regras, se non usásemos a opción `-i`, os paquetes dirixidos ós servizos http/https de server irían hacia a cadea WEB_INTRA en vez de cara ó servidor e non chegarían a él ó morrer alí. Por suposto, non hai unha única maneira de crear o ruleset do firewall, e poderíamos colocar a regra de acceso ó server como regra#2 ou incluso poderíamos deixala na posición 5 eliminando a regra DROP de WEB_INTRA; e xa desaparecería o problema mencionado.

Administración por ssh do server dende Internet

O explicado no apartado anterior é válido tamén para o acceso por ssh dende o equipo administrador da casa para administrar por ssh o server: hai que facer DNAT e autorizar os paquetes cunha regra de filtrado. O natural é facer DNAT para mapear a IP pública de FW Linux-porto tcp/22 coa IP privada de Server-porto tcp/22. Sen embargo, aquí xurde un problema: estamos asignando o mesmo socket <192.168.1.254, tcp, 22> para acceder por ssh a FW Linux e para acceder por ssh a Server. Ese conflicto queda 'resolto' pola forma de traballar de netfilter; xa que, ó chegar un paquete dende o equipo de casa do admin con destino<192.168.1.254, tcp, 22>, farase DNAT modificando o paquete e reemprazando a IP Pública destino pola IP privada de Server e remataríamos administrando o server, quedando FW Linux inaccesible por ssh dende Internet:

Fig. Conflictos de portos e funcionamento de netfilter.

Existen múltiples solucións para solucionar este conflicto de portos e poder acceder por ssh tanto a FW Linux coma ó servidor interno. A continuación indícanse algunhas delas:

- Cambiar o porto de traballo do servizo ssh en FW Linux e actualizar as regras de filtrado nas cadeas INPUT e SSH, deixando o porto tcp/22 para acceder a server.

- Cambiar o porto de traballo do servizo ssh en server, deixando o porto tcp/22 para acceder a FW Linux.
- Deixar os servizos ssh no porto tcp/22 pero usar outro porto público para acceder a server.

Empregaremos a última solución para ver como se pode facer NAT cambiando o número de porto. Mapearemos o porto público tcp/22000 co porto tcp/22 de server:

```
uadmin@fw-linux:~$ sudo iptables -t nat -A PREROUTING -p tcp -d 192.168.1.254
--dport 22000 -j DNAT --to-destination 192.168.56.253:22
uadmin@fw-linux:~$ sudo iptables -A FORWARD -p tcp -s 192.168.1.2 -d
192.168.56.253 --dport 22 -m state --state NEW -j ACCEPT
uadmin@fw-linux:~$ sudo iptables -t nat -L -n --line-numbers
Chain PREROUTING (policy ACCEPT)
num target prot opt source destination
1 DNAT tcp -- 0.0.0.0/0 192.168.1.254 multiport dports 80,443 to:192.168.56.253
2  DNAT  tcp  -- 0.0.0.0/0    192.168.1.254    tcp dpt:22000
to:192.168.56.253:22
Chain INPUT (policy ACCEPT)
num target prot opt source destination
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
Chain POSTROUTING (policy ACCEPT)
num target prot opt source destination
1 SNAT all -- 192.168.56.0/24 0.0.0.0/0 to:192.168.1.254
uadmin@fw-linux:~$ sudo iptables -L FORWARD -n --line-numbers
Chain FORWARD (policy DROP)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 DNS_INTRA udp -- 0.0.0.0/0 8.8.8.8 udp dpt:53 state NEW
3 DNS_INTRA udp -- 0.0.0.0/0 8.8.4.4 udp dpt:53 state NEW
4 WEB_INTRA tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
5 ACCEPT tcp -- 0.0.0.0/0 192.168.56.253 multiport dports 80,443 state NEW
6  ACCEPT  tcp  -- 192.168.1.2   192.168.56.253    tcp dpt:22 state NEW
uadmin@fw-linux:~$
```

Desta forma solucionamos o problema e poderemos acceder por ssh usando os seguintes comandos:

- `ssh uadmin@192.168.1.254` para acceder por ssh e administrar FW Linux. É dicir, deixamos o socket <192.168.1.254, tcp, 22> para acceder por ssh a FW Linux.
- `ssh uadmin@192.168.1.254 -p 22000` para acceder por ssh e administrar server. É dicir, usamos o socket <192.168.1.254, tcp, 22000> e DNAT para acceder por ssh ó server.

7.6 Rexistros

Uns dos requisitos do enunciado era o rexistro e bloqueo dos intentos de acceso por ssh a FW Linux dende equipos non autorizados. Para acadar estes obxectivos creouse unha regra coa acción LOG e outra coa acción DROP:

```
uadmin@fw-linux:~$ sudo iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination
1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 SSH tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
...
Chain SSH (1 references)
num target prot opt source destination
1 ACCEPT all -- 192.168.1.2 0.0.0.0/0
2 ACCEPT all -- 192.168.56.1 0.0.0.0/0
3 ACCEPT all -- 192.168.56.2 0.0.0.0/0
4 LOG all -- 0.0.0.0/0 0.0.0.0/0 LOG flags 0 level 4 prefix "iptables: SSH Bloqueo "
5 DROP all -- 0.0.0.0/0 0.0.0.0/0
```

Os intentos de acceso non autorizados quedan rexistrados no arquivo `/var/log/kern.log` e para facilitar a localización dos eventos as liñas teñen o texto "iptables: SSH Bloqueo". Podemos provocar o rexistro tratando de acceder a FW Linux por ssh dende algún equipo non autorizado. A continuación amósanse algunhas entradas do arquivo:

```
uadmin@fw-linux:~$ less /var/log/kern.log
Jan 31 08:57:04 fw-linux kernel: [ 15.732056] intel_rapl: domain package energy ctr 0:0 not working, skip
Jan 31 08:57:04 fw-linux kernel: [ 15.863076] intel_rapl: domain core energy ctr 0:0 not working, skip
Jan 31 08:57:04 fw-linux kernel: [ 15.984121] IPv6: ADDRCONF(NETDEV_UP): eth1: link is not ready
Jan 31 08:57:04 fw-linux kernel: [ 15.990003] e1000: eth1 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
Jan 31 08:57:04 fw-linux kernel: [ 15.990003] IPv6: ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready
Jan 31 08:57:04 fw-linux kernel: [ 15.993373] intel_rapl: domain uncore energy ctr 0:0 not working, skip
Jan 31 08:57:04 fw-linux kernel: [ 15.993373] intel_rapl: no valid rapl domains found in package 0
```

```
Jan 31 08:57:04 fw-linux kernel: [ 16.004432] e1000: eth0 NIC Link is Up 1000
Mbps Full Duplex, Flow Control: RX
Jan 31 08:57:04 fw-linux kernel: [ 16.200054] EXT4-fs (sda1): re-mounted. Opts:
errors=remount-ro
Jan 31 08:57:05 fw-linux kernel: [ 18.704126] ip_tables: (C) 2000-2006 Netfilter
Core Team
Jan 31 08:57:05 fw-linux kernel: [ 18.712597] nf_conntrack version 0.5.0 (3919
buckets, 15676 max)
Jan 31 08:57:06 fw-linux kernel: [ 19.552916] xt_time: kernel timezone is +0100
Jan 31 08:57:06 fw-linux kernel: [ 19.561677] ip6_tables: (C) 2000-2006
Netfilter Core Team

Jan 31 13:56:51 fw-linux kernel: [18004.839588] iptables: SSH Bloqueo IN=eth1
OUT=       MAC=08:00:27:d1:74:0e:08:00:27:50:7c:37:08:00       SRC=192.168.56.253
DST=192.168.56.254  LEN=60  TOS=0x00  PREC=0x00  TTL=64  ID=28309  DF  PROTO=TCP
SPT=37556 DPT=22 WINDOW=29200 RES=0x00 SYN URGP=0

Jan 31 14:10:55 fw-linux kernel: [18848.794862] iptables: SSH Bloqueo IN=eth1
OUT=       MAC=08:00:27:d1:74:0e:08:00:27:50:7c:37:08:00       SRC=192.168.56.253
DST=192.168.56.254  LEN=200  TOS=0x00  PREC=0x00  TTL=64  ID=28324  DF  PROTO=TCP
SPT=37556 DPT=22 WINDOW=592 RES=0x00 ACK PSH URGP=0
Jan 31 14:10:57 fw-linux kernel: [18850.155472] iptables: SSH Bloqueo IN=eth1
OUT=       MAC=08:00:27:d1:74:0e:08:00:27:50:7c:37:08:00       SRC=192.168.56.253
DST=192.168.56.254  LEN=60  TOS=0x00  PREC=0x00  TTL=64  ID=48444  DF  PROTO=TCP
SPT=37557 DPT=22 WINDOW=29200 RES=0x00 SYN URGP=0
Jan 31 14:10:58 fw-linux kernel: [18851.153796] iptables: SSH Bloqueo IN=eth1
OUT=       MAC=08:00:27:d1:74:0e:08:00:27:50:7c:37:08:00       SRC=192.168.56.253
DST=192.168.56.254  LEN=60  TOS=0x00  PREC=0x00  TTL=64  ID=48445  DF  PROTO=TCP
SPT=37557 DPT=22 WINDOW=29200 RES=0x00 SYN URGP=0
Jan 31 14:11:00 fw-linux kernel: [18853.158098] iptables: SSH Bloqueo IN=eth1
OUT=       MAC=08:00:27:d1:74:0e:08:00:27:50:7c:37:08:00       SRC=192.168.56.253
DST=192.168.56.254  LEN=60  TOS=0x00  PREC=0x00  TTL=64  ID=48446  DF  PROTO=TCP
SPT=37557 DPT=22 WINDOW=29200 RES=0x00 SYN URGP=0
```

Movendo os logs de iptables a outro arquivo con rsyslog

Pode observarse que no arquivo `kern.log` vólcanse ademais de rexistros de netfilter outro tipo de información. Pode ser desexable desviar os logs de iptables a un arquivo propio para a súa mellor revisión. En Ubuntu Linux o proceso a seguir é o seguinte:

```
uadmin@fw-linux:~$ sudo nano /etc/rsyslog.d/00-iptables.conf
```

```
:msg,contains,"iptables: " -/var/log/iptables.log
& ~
```

Unha vez creado o arquivo e indicado que as liñas que teñan a expresión "iptables: " sexan gardadas en `/var/log/iptables.log` procedemos a reiniciar o servizo rsyslog:

```
uadmin@fw-linux:~$ sudo service rsyslog restart
```

Se tratamos de acceder por ssh dende un equipo non autorizado debe aparecer no directorio `/var/log` o arquivo `iptables.log` cos rexistros da incidencia. No caso de crear a man o arquivo é importante respectar o propietario e grupo do mesmo (syslog e adm respectivamente):

```
uadmin@fw-linux:~$ ls -lahF /var/log/

total 1,9M
drwxrwxr-x 9 root syslog 4,0K feb 1 01:20 ./
drwxr-xr-x 12 root root 4,0K jun 29 2014 ../
-rw-r--r-- 1 root root 22K ene 29 23:54 alternatives.log
drwxr-xr-x 2 root root 4,0K jun 29 2014 apt/
-rw-r----- 1 syslog adm 121K feb 1 01:19 auth.log
-rw-r--r-- 1 root root 4,0K ene 31 08:57 boot.log
-rw-r--r-- 1 root root 61K abr 16 2014 bootstrap.log
-rw-rw---- 1 root utmp 768 ene 30 21:55 btmp
drwxr-xr-x 2 root root 4,0K abr 12 2014 dist-upgrade/
-rw-r----- 1 root adm 29K ene 31 08:57 dmesg
-rw-r----- 1 root adm 29K ene 31 00:42 dmesg.0
-rw-r----- 1 root adm 9,0K ene 30 21:05 dmesg.1.gz
-rw-r----- 1 root adm 9,0K ene 30 09:10 dmesg.2.gz
-rw-r----- 1 root adm 9,0K ene 30 08:51 dmesg.3.gz
-rw-r----- 1 root adm 9,0K ene 30 00:03 dmesg.4.gz
-rw-r--r-- 1 root root 577K feb 1 00:39 dpkg.log
-rw-r--r-- 1 root root 32K ene 30 00:14 faillog
drwxr-xr-x 2 root root 4,0K jun 29 2014 fsck/
drwxr-xr-x 3 root root 4,0K jun 29 2014 installer/
-rw-r----- 1 syslog adm 546 feb 1 01:20 iptables.log
-rw-r----- 1 syslog adm 357K feb 1 00:40 kern.log
drwxr-xr-x 2 landscape root 4,0K jun 29 2014 landscape/
-rw-rw-r-- 1 root utmp 286K feb 1 00:35 lastlog
-rw-r----- 1 syslog adm 396K feb 1 01:19 syslog
-rw-r--r-- 1 root root 106K ene 31 08:57 udev
drwxr-xr-x 2 root root 4,0K jun 29 2014 unattended-upgrades/
```

```
drwxr-xr-x 2 root root 4,0K ene 29 23:54 upstart/
-rw-rw-r-- 1 root utmp 53K feb 1 00:35 wtmp
```

```
uadmin@fw-linux:~$ cat /var/log/iptables.log
```

```
Feb 1 01:20:14 fw-linux kernel: [59007.979456] iptables: SSH Bloqueo IN=eth1 OUT= MAC=08:00:27:d1:74:0e:08:00:27:50:7c:37:08:00 SRC=192.168.56.253 DST=192.168.56.254 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=55357 DF PROTO=TCP SPT=37564 DPT=22 WINDOW=29200 RES=0x00 SYN URGP=0
Feb 1 01:20:15 fw-linux kernel: [59008.977908] iptables: SSH Bloqueo IN=eth1 OUT= MAC=08:00:27:d1:74:0e:08:00:27:50:7c:37:08:00 SRC=192.168.56.253 DST=192.168.56.254 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=55358 DF PROTO=TCP SPT=37564 DPT=22 WINDOW=29200 RES=0x00 SYN URGP=0
Feb 1 01:20:17 fw-linux kernel: [59010.981777] iptables: SSH Bloqueo IN=eth1 OUT= MAC=08:00:27:d1:74:0e:08:00:27:50:7c:37:08:00 SRC=192.168.56.253 DST=192.168.56.254 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=55359 DF PROTO=TCP SPT=37564 DPT=22 WINDOW=29200 RES=0x00 SYN URGP=0
Feb 1 01:20:21 fw-linux kernel: [59014.990193] iptables: SSH Bloqueo IN=eth1 OUT= MAC=08:00:27:d1:74:0e:08:00:27:50:7c:37:08:00 SRC=192.168.56.253 DST=192.168.56.254 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=55360 DF PROTO=TCP SPT=37564 DPT=22 WINDOW=29200 RES=0x00 SYN URGP=0
uadmin@fw-linux:~$
```

Se queremos reducir o número de entradas recollidas nos logs, pódemos aplicar unha limitación a través da opción -m limit. No seguinte exemplo gárdanse 10 rexistros e despois baixase o ritmo a razón de 5 entradas por minuto. Se non hai avalancha de incidencias voltase a incrementar o número de entradas ata chegar ó especificado por limit-burst.

```
uadmin@fw-linux:~$ sudo iptables -A SSH -m limit --limit 5/m --limit-burst 10 -j LOG --log-prefix "iptables: SSH Bloqueo "
```

Usando NFLOG para redixir os logs de iptables a outro arquivo

Outro método (que si vale para os contedores LXC/LXD) para ter os logs de iptables nun arquivo diferente é usar o servizo ulogd e NFLOG nas regras iptables en vez de LOG. Para usar este sistema comezamos instalando ulogd:

```
uadmin@fw-linux:~$ sudo apt-get install ulogd2
```

A partir de agora, cando desexemos rexistrar incidencias en netfilter/iptables empregaremos NFLOG. No noso exemplo temos a cadea SSH cunha regra con LOG:

```
uadmin@fw-linux:~$ sudo iptables -L SSH -n --line-numbers
sudo: unable to resolve host fw-linux
Chain SSH (1 references)
num target prot opt source destination
1 ACCEPT all -- 192.168.1.2 0.0.0.0/0
2 ACCEPT all -- 192.168.56.1 0.0.0.0/0
3 ACCEPT all -- 192.168.56.2 0.0.0.0/0
4 LOG all -- 0.0.0.0/0 0.0.0.0/0 LOG flags 0 level 4 prefix "iptables: SSH
BLoqueo"
5 DROP all -- 0.0.0.0/0 0.0.0.0/0
uadmin@fw-linux:~$
```

Procedemos a reemplazar a regra cunha nova que use NFLOG:

```
uadmin@fw-linux:~$ sudo iptables -R SSH 4 -j NFLOG --nflog-prefix "iptables: SSH
BLoqueo"
uadmin@fw-linux:~$ sudo iptables -L SSH -n --line-numbers
Chain SSH (1 references)
num target prot opt source destination
1 ACCEPT all -- 192.168.1.2 0.0.0.0/0
2 ACCEPT all -- 192.168.56.1 0.0.0.0/0
3 ACCEPT all -- 192.168.56.2 0.0.0.0/0
4 NFLOG all -- 0.0.0.0/0 0.0.0.0/0 nflog-prefix "iptables: SSH BLoqueo"
5 DROP all -- 0.0.0.0/0 0.0.0.0/0
uadmin@fw-linux:~$
```

A partir deste momento rexistraranse as incidencias no arquivo/var/log/ulog/syslogemu.log :

```
uadmin@fw-linux:~$ ls -lahF /var/log/ulog/
total 2.5K
drwxr-xr-x 2 root root 3 Oct 15 22:07 ./
drwxrwxr-x 9 root syslog 25 Oct 15 22:07 ../
-rw-r--r-- 1 root root 0 Oct 15 22:07 syslogemu.log
uadmin@fw-linux:~$ cat /var/log/ulog/syslogemu.log
Oct 15 22:17:20 fw-linux iptables: SSH BLoqueo IN=eth0 OUT=
MAC=00:16:3e:f1:b3:36:08:00:27:48:e3:f7:08:00 SRC=192.168.1.150
```

```
DST=192.168.1.254 LEN=60 TOS=00 PREC=0x00 TTL=64 ID=53870 DF PROTO=TCP SPT=35008
DPT=22 SEQ=432581504 ACK=0 WINDOW=29200 SYN URGP=0 MARK=0
Oct 15 22:17:21 fw-linux iptables: SSH BLoqueo IN=eth0 OUT=
MAC=00:16:3e:f1:b3:36:08:00:27:48:e3:f7:08:00 SRC=192.168.1.150
DST=192.168.1.254 LEN=60 TOS=00 PREC=0x00 TTL=64 ID=53871 DF PROTO=TCP SPT=35008
DPT=22 SEQ=432581504 ACK=0 WINDOW=29200 SYN URGP=0 MARK=0
uadmin@fw-linux:~$
```

7.7 Probas de funcionamento

Probas de funcionamento

A continuación faremos unhas sinxelas probas para verificar o correcto funcionamento do sistema:

Resolución dns e acceso ós repositorios en FW Linux:

```
uadmin@fw-linux:~$ host www.xunta.es
www.xunta.es has address 85.91.64.254
uadmin@fw-linux:~$ host www.xunta.es 8.8.8.8
Using domain server:
Name: 8.8.8.8
Address: 8.8.8.8#53
Aliases:
www.xunta.es has address 85.91.64.254
uadmin@fw-linux:~$ sudo apt-get install ipcalc
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
 ipcalc
0 actualizados, 1 se instalarán, 0 para eliminar y 0 no actualizados.
Necesito descargar 24,6 kB de archivos.
Se utilizarán 109 kB de espacio de disco adicional después de esta operación.
Des:1 http://es.archive.ubuntu.com/ubuntu/ trusty/universe ipcalc all 0.41-4
[24,6 kB]
Descargados 24,6 kB en 0seg. (68,2 kB/s)
Seleccionando el paquete ipcalc previamente no seleccionado.
(Leyendo la base de datos ... 85854 ficheros o directorios instalados
actualmente.)
Preparing to unpack .../archives/ipcalc_0.41-4_all.deb ...
```

```
Unpacking ipcalc (0.41-4) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Configurando ipcalc (0.41-4) ...
uadmin@fw-linux:~$
```

Acceso por ssh dende un equipo admin da Intranet:

```
manuel@lubuntu:~$ ssh uadmin@192.168.56.254
uadmin@192.168.56.254's password:
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-76-generic x86_64)
* Documentation: https://help.ubuntu.com/
 System information as of Mon Feb 1 00:35:56 CET 2016
 System load: 0.0 Processes: 76
 Usage of /: 7.7% of 19.07GB Users logged in: 1
 Memory usage: 12% IP address for eth0: 192.168.1.254
 Swap usage: 0% IP address for eth1: 192.168.56.254
 Graph this data and manage this system at:
 https://landscape.canonical.com/
1 package can be updated.
0 updates are security updates.
Last login: Mon Feb 1 00:35:56 2016 from 192.168.56.1
uadmin@fw-linux:~$
```

Acceso por ssh dende o equipo admin de Internet:

```
manuel@lubuntu:~$ ssh uadmin@192.168.1.254
The authenticity of host '192.168.1.254 (192.168.1.254)' can't be established.
ECDSA key fingerprint is f9:8c:1c:ae:7b:d2:a0:6a:6e:94:b0:41:a3:51:b2:79.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.254' (ECDSA) to the list of known hosts.
uadmin@192.168.1.254's password:
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-76-generic x86_64)
* Documentation: https://help.ubuntu.com/
 System information as of Mon Feb 1 01:34:04 CET 2016
 System load: 0.04 Processes: 68
 Usage of /: 7.7% of 19.07GB Users logged in: 1
 Memory usage: 12% IP address for eth0: 192.168.1.254
 Swap usage: 0% IP address for eth1: 192.168.56.254
 Graph this data and manage this system at:
 https://landscape.canonical.com/
Last login: Mon Feb 1 01:34:05 2016 from 192.168.56.1
```

```
uadmin@fw-linux:~$
```

Acceso por ssh a server dende equipo admin en Internet :

```
manuel@lubuntu:~$ ssh uadmin@192.168.1.254 -p 22000
The authenticity of host '[192.168.1.254]:22000 ([192.168.1.254]:22000)' can't
be established.
ECDSA key fingerprint is 3d:36:e6:45:94:11:59:b5:8d:a4:9e:c1:fb:97:91:45.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[192.168.1.254]:22000' (ECDSA) to the list of known
hosts.
uadmin@192.168.1.254's password:
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-76-generic x86_64)
* Documentation: https://help.ubuntu.com/
 System information as of Sun Jan 31 23:47:00 CET 2016
 System load: 0.0 Processes: 74
 Usage of /: 7.7% of 19.07GB Users logged in: 1
 Memory usage: 12% IP address for eth0: 192.168.56.253
 Swap usage: 0%
 Graph this data and manage this system at:
 https://landscape.canonical.com/
Last login: Sun Jan 31 23:47:00 2016 from 192.168.1.2
uadmin@server:~$
```

Acceso a FW Linux por ssh dende un equipo non autorizado:

```
uadmin@server:~$ ssh uadmin@192.168.56.254
ssh: connect to host 192.168.56.254 port 22: Connection timed out
uadmin@server:~$
```

Acceso dende Internet ó servidor Web por https e entradas no `conntrack`:

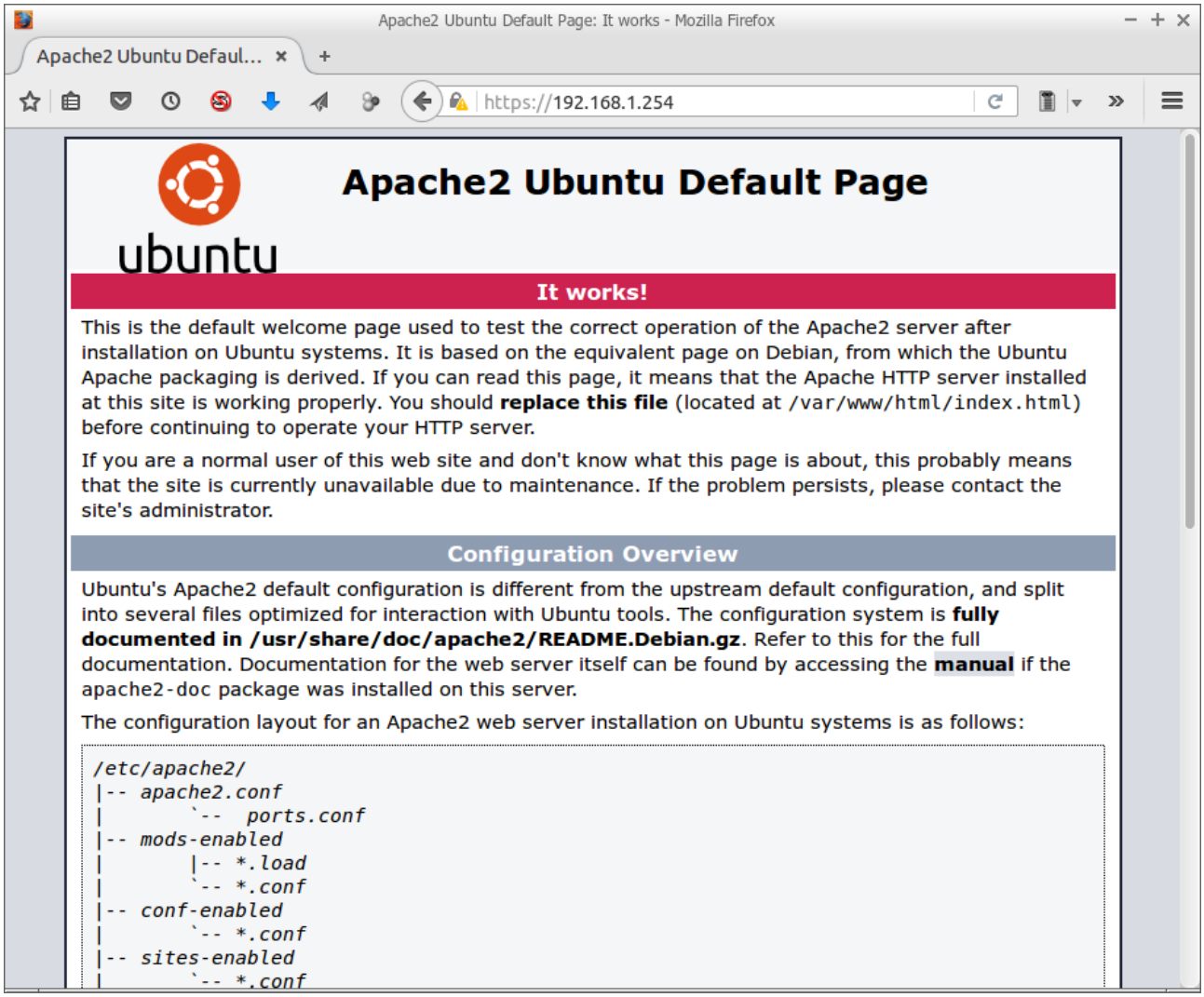


Fig. Acceso por https ó servidor web

Fixarse que na saída de conntrack aparece en primeiro lugar información da conexión no sentido no que se orixinou (192.168.1.2 --> 192.168.1.254) e despois aparece no outro sentido (192.168.56.253 --> 192.168.1.2).

```
uadmin@fw-linux:~$ sudo conntrack -L
tcp 6 431999 ESTABLISHED src=192.168.1.2 dst=192.168.1.254 sport=47514 dport=443
src=192.168.56.253 dst=192.168.1.2 sport=443 dport=47514 [ASSURED] mark=0 use=1
unknown 2 584 src=192.168.1.1 dst=224.0.0.1 [UNREPLIED] src=224.0.0.1
dst=192.168.1.1 mark=0 use=1
tcp 6 431999 ESTABLISHED src=192.168.56.1 dst=192.168.56.254 sport=46621
dport=22 src=192.168.56.254 dst=192.168.56.1 sport=22 dport=46621 [ASSURED]
mark=0 use=1
```

```
tcp 6 431627 ESTABLISHED src=192.168.1.2 dst=192.168.1.254 sport=58849
dport=22000 src=192.168.56.253 dst=192.168.1.2 sport=22 dport=58849 [ASSURED]
mark=0 use=1
conntrack v1.4.1 (conntrack-tools): 4 flow entries have been shown.
uadmin@fw-linux:~$
```

Erro dun cliente ó tratar de navegar fóra do horario permitido:

Fig. Navegación bloqueada fóra do horario habilitado

Se probamos a xerar tráfico, tanto dende Internet hacia ó server como dende a intranet hacia Internet, e usamos os opcións de filtrado -n e -g de conntrack poderemos filtrar as conexións con SNAT e DNAT respectivamente:

```
uadmin@ubuntu16:~$ sudo conntrack -L -n
tcp 6 300 ESTABLISHED src=192.168.56.253 dst=91.189.88.162 sport=60529 dport=80
src=91.189.88.162 dst=192.168.1.254 sport=80 dport=60529 [ASSURED] mark=0 use=1
tcp 6 299 ESTABLISHED src=192.168.56.253 dst=91.189.92.150 sport=33181 dport=80
src=91.189.92.150 dst=192.168.1.254 sport=80 dport=33181 [ASSURED] mark=0 use=1
udp 17 178 src=192.168.56.253 dst=8.8.8.8 sport=33883 dport=53 src=8.8.8.8
dst=192.168.1.254 sport=53 dport=33883 [ASSURED] mark=0 use=1
tcp 6 299 ESTABLISHED src=192.168.56.253 dst=91.189.92.152 sport=41276 dport=80
src=91.189.92.152 dst=192.168.1.254 sport=80 dport=41276 [ASSURED] mark=0 use=1
```

```
udp 17 178 src=192.168.56.253 dst=8.8.8.8 sport=54925 dport=53 src=8.8.8.8
dst=192.168.1.254 sport=53 dport=54925 [ASSURED] mark=0 use=1
tcp 6 431999 ESTABLISHED src=192.168.56.253 dst=91.189.91.26 sport=59111
dport=80 src=91.189.91.26 dst=192.168.1.254 sport=80 dport=59111 [ASSURED]
mark=0 use=1
udp 17 178 src=192.168.56.253 dst=8.8.8.8 sport=36460 dport=53 src=8.8.8.8
dst=192.168.1.254 sport=53 dport=36460 [ASSURED] mark=0 use=1
udp 17 172 src=192.168.56.253 dst=8.8.8.8 sport=60722 dport=53 src=8.8.8.8
dst=192.168.1.254 sport=53 dport=60722 [ASSURED] mark=0 use=1
conntrack v1.4.3 (conntrack-tools): 8 flow entries have been shown.
uadmin@ubuntu16:~$ sudo conntrack -L -g
tcp 6 110 TIME_WAIT src=192.168.1.2 dst=192.168.1.254 sport=35791 dport=22000
src=192.168.56.253 dst=192.168.1.2 sport=22 dport=35791 [ASSURED] mark=0 use=1
tcp 6 431992 ESTABLISHED src=192.168.1.2 dst=192.168.1.254 sport=35816
dport=22000 src=192.168.56.253 dst=192.168.1.2 sport=22 dport=35816 [ASSURED]
mark=0 use=1
tcp 6 107 TIME_WAIT src=192.168.1.2 dst=192.168.1.254 sport=54397 dport=80
src=192.168.56.253 dst=192.168.1.2 sport=80 dport=54397 [ASSURED] mark=0 use=1
tcp 6 104 TIME_WAIT src=192.168.1.2 dst=192.168.1.254 sport=54394 dport=80
src=192.168.56.253 dst=192.168.1.2 sport=80 dport=54394 [ASSURED] mark=0 use=1
tcp 6 112 TIME_WAIT src=192.168.1.2 dst=192.168.1.254 sport=54401 dport=80
src=192.168.56.253 dst=192.168.1.2 sport=80 dport=54401 [ASSURED] mark=0 use=1
conntrack v1.4.3 (conntrack-tools): 5 flow entries have been shown
```

Este é un moi bo momento para facer a Tarefa de Av. II -B.

7.8 Rede de lazo pechado

Esta regra está pensada para permitir o tráfico o host dirixido ó propio host; e sen ela, pode haber problemas de comunicacións entre aplicacións correndo no mesmo equipo. Para crear estas adoitase facer de calquera das seguintes formas e inmediatamente despois de crear as regras para as conexións ESTABLISHED, RELATED:

```
uadmin@server:~$ sudo iptables -A INPUT -i lo -j ACCEPT
uadmin@server:~$ sudo iptables -A OUTPUT -o lo -j ACCEPT
```

Ou

```
uadmin@server:~$ sudo iptables -A INPUT -i lo -m state --state NEW -j ACCEPT
uadmin@server:~$ sudo iptables -A OUTPUT -o lo -m state --state NEW -j ACCEPT
```

Podemos ver que especificamos como interface de entrada/saída o dispositivo lo, que ven sendo o dispositio de loopback ou lazo pechado:

```
uadmin@server:~$ ifconfig -a
eth0 Link encap:Ethernet direcciónHW 08:00:27:50:7c:37
 Direc. inet:192.168.56.253 Difus.:192.168.56.255 Másc:255.255.255.0
 Dirección inet6: fe80::a00:27ff:fe50:7c37/64 Alcance:Enlace
 ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Métrica:1
 Paquetes RX:7025 errores:0 perdidos:0 overruns:0 frame:0
 Paquetes TX:5033 errores:0 perdidos:0 overruns:0 carrier:0
 colisiones:0 long.colaTX:1000
 Bytes RX:647977 (647.9 KB) TX bytes:1059457 (1.0 MB)
lo Link encap:Bucle local
 Direc. inet:127.0.0.1 Másc:255.0.0.0
 Dirección inet6: ::1/128 Alcance:Anfitrión
 ACTIVO BUCLE FUNCIONANDO MTU:65536 Métrica:1
 Paquetes RX:89 errores:0 perdidos:0 overruns:0 frame:0
 Paquetes TX:89 errores:0 perdidos:0 overruns:0 carrier:0
 colisiones:0 long.colaTX:0
 Bytes RX:11564 (11.5 KB) TX bytes:11564 (11.5 KB)
uadmin@server:~$
```

É importante facer uso da opción `-v` no listado de regras para ter visibles as interfaces implicadas nas regras e non facer unha interpretación errónea das mesmas:

```
uadmin@server:~$ sudo iptables -L -n -v --line-numbers
Chain INPUT (policy DROP 0 packets, 0 bytes)
num pkts bytes target prot opt in out source destination
1 1913 143K ACCEPT all -- * * 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 0 0 ACCEPT all -- lo * 0.0.0.0/0 0.0.0.0/0 state NEW
3 1 60 ACCEPT tcp -- * * 0.0.0.0/0 0.0.0.0/0 tcp dpt:22 state NEW
Chain FORWARD (policy DROP 0 packets, 0 bytes)
num pkts bytes target prot opt in out source destination
Chain OUTPUT (policy DROP 0 packets, 0 bytes)
num pkts bytes target prot opt in out source destination
1 1528 174K ACCEPT all -- * * 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
2 1 60 ACCEPT all -- * lo 0.0.0.0/0 0.0.0.0/0 state NEW
3 2 116 ACCEPT udp -- * * 0.0.0.0/0 8.8.8.8        udp dpt:53 state NEW
4 0 0 ACCEPT udp -- * * 0.0.0.0/0 8.8.4.4        udp dpt:53 state NEW
5 0 0 ACCEPT tcp -- * * 0.0.0.0/0 0.0.0.0/0 multiport dports 80,443 state NEW
```