[**Tres nivells**](#_t32q9z8yck66) **3**

[**Diagrames de flux**](#_lkwmg6cjedcx) **4**

[**Utilitzar iptables enlloc de firewalld**](#_kcd4es80ke51) **8**

[**Arxius i programes relacionats**](#_czy31fmlnzrf) **10**

[**Com consultar l'estat actual d' iptables**](#_alzif5f9hlo5) **11**

[**Com guardar les regles d’ iptables**](#_o7b3xrt1mdtu) **12**

[**Com exportar les regles d’un servidor per importar-les a un altre**](#_2ombsanhevk) **13**

[**Com importar les regles des d’un arxiu**](#_vvy959qe9cw4) **14**

[**Com esborrar una regla concreta d’iptables**](#_kajsne5esrbz) **15**

[**Com esborrar totes les regles**](#_rj9irqdxixnr) **16**

[**Com crear una política base restrictiva**](#_u1d8v1m6mgv8) **17**

[**Com restaurar iptables a l'estat per defecte**](#_m8q3ex4ms792) **18**

[**Configurar iptables amb eines gràfiques**](#_tbhu563jfu9) **19**

[**Com permetre que el nostre PC pugui enrutar**](#_vzt9utx0m969) **20**

[De manera temporal usando procfs](#_5ggb4b1a8qak) 20

[De manera temporal usando sysctl](#_d1lsxfq2ks96) 20

[De manera definitiva](#_jc2czw2jm6vi) 21

[**Com habilitar NAT, SNAT, MASQUERADE**](#_b7a53feoxku) **22**

[**Com habilitar DNAT**](#_4724qbhuap8j) **23**

[**Deshabiltar el ping**](#_d64lq4rc1cte) **24**

[**LOGs**](#_567kgi6vc6kp) **24**

[Deixem un LOG per cada intent de connexió al port 22](#_2lgyj4wt057w) 25

[Deixem un LOG per cada intent de resolució DNS al port 53](#_2uhwj1abch5h) 25

[**DNS de caché i reenviament i facebook.com cap a la IP de google.com**](#_pcrq7qegmnn0) **26**

[Instal·lació](#_vhr8ofqssyc0) 26

[Comprovem quins arxius i binaris té](#_xiehgjbbkhko) 26

[Comprovem el servei](#_l8x09493r9ri) 26

[Configurem el servidor DNS](#_mj74dworevho) 26

[Editem /etc/named.conf](#_ttffhydtou0w) 26

[Creem /var/named/facebook.com.zone](#_48lwe8fjjxf7) 28

[**Forçar DNS local**](#_hw1e1l5cqqxr) **29**

[**Reenviar les peticions per ssh cap al servidor amb iptables a un altre PC**](#_6smh81filqga) **30**

[**QoS**](#_mzgptg6q9g93) **30**

[**Mòdul connection tracking**](#_bx6tceb63eks) **32**

[Instal·lar el mòdul](#_gmye9rmplnhg) 32

[**Mòduls iprange/multiport i subnetting per reduïr les regles**](#_nl5crama9ujb) **34**

[Reduïr regles amb subnetting](#_t3t9nb319c03) 34

[Amb mòduls iprange i multiport](#_ylz3t4tqj2kq) 34

[**Mòdul mac (filtratge per mac address)**](#_6cumfslcpm6a) **35**

[**Mòdul time (filtratge per franja horària)**](#_flcqw4ucn8e5) **36**

[**Mòdul limit**](#_ygw8jye439m7) **37**

[Syntax and Example](#_qxgfilbut9wj) 37

[Syn-flood protection:](#_xmw90d8mzz1z) 38

[Furtive port scanner:](#_og38xi69nx1f) 38

[Ping of death:](#_5hrper9yd87q) 38

[Other Examples](#_h8p3mk5l9txt) 39

[Conclusion](#_e2vthnkakw4w) 42

[**Mòdul recent**](#_os5311yc1jfr) **43**

[**Canviar el gateway del equip que farà de client**](#_4cukbh7vpdv1) **44**

[**Software malicioso y suplantación de direcciones IP**](#_irhlwi4hs2nr) **45**

[**Blocking Port Scanning**](https://docs.google.com/document/d/1aDwbPQcg4Urkjv8J7zxQBmuX8HPCPaUq2b_drp0gnNA/edit#heading=h.yz6dx9xkg2y5)[**46**](https://docs.google.com/document/d/1aDwbPQcg4Urkjv8J7zxQBmuX8HPCPaUq2b_drp0gnNA/edit#heading=h.yz6dx9xkg2y5)

[**Comencem a provar amb filter + INPUT/OUTPUT**](#_h9bxnoe9b5ev) **47**

[Per fer proves treballarem amb ssh(port22) i nginx(web server port 80)](#_1qc5igkqh083) 47

[Com mirem quins ports hi ha oberts](#_5iag3l2104zy) 47

[Al nostre equip](#_ui0jkub62qxm) 47

[A un altre equip de la xarxa](#_ll2huxu7zu0g) 47

[Proves OUTPUT](#_o86f7zezudyk) 48

[Prova DROP de tot](#_kms4alq4huek) 48

[Prova DROP de tot excepte navegació pel port 80 (web http)](#_dxdvjwlrtvvr) 50

[Prova DROP de tot excepte l’accés a una ip de LAN](#_lb31vjzb2jdo) 52

[Exemple DROP de tots els PCs excepte els que pertanyen a una xarxa determinada](#_v17slfkpinoy) 54

[Exemple DROP de tots excepte cap a els PCs que pertanyen a una xarxa determinada](#_glbc8t3r0gjm) 55

[Prova DROP de tot excepte navegació pel port 80 a una ip de LAN](#_yvkmnswct3lq) 56

[Prova DROP de tot excepte navegació pel port 80 a una ip de LAN, només des d’una xarxa concreta](#_mlizee2kq9wd) 58

[Prova DROP de tot excepte navegació web local i per Internet](#_hgs5anb9erho) 60

[Exercicis filter + INPUT](#_x1e60dye9i6v) 62

[Exercicis NAT + MASQUERADE](#_3hpvzatqmu8m) 62

[Exercicis NAT + DNAT](#_o10g9vbp3h1s) 62

[Exercicis filter + OUTPUT](#_1ddavjcn8cn7) 62

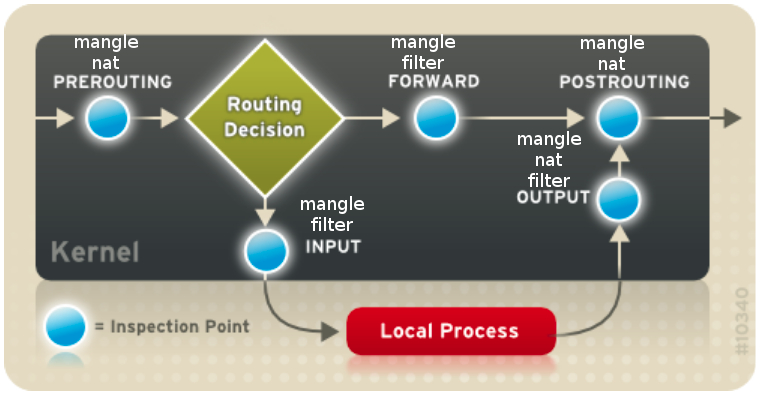
[**Pràctica a corregir als vostres PCs**](#_jjacjjm6i4p8) **63**

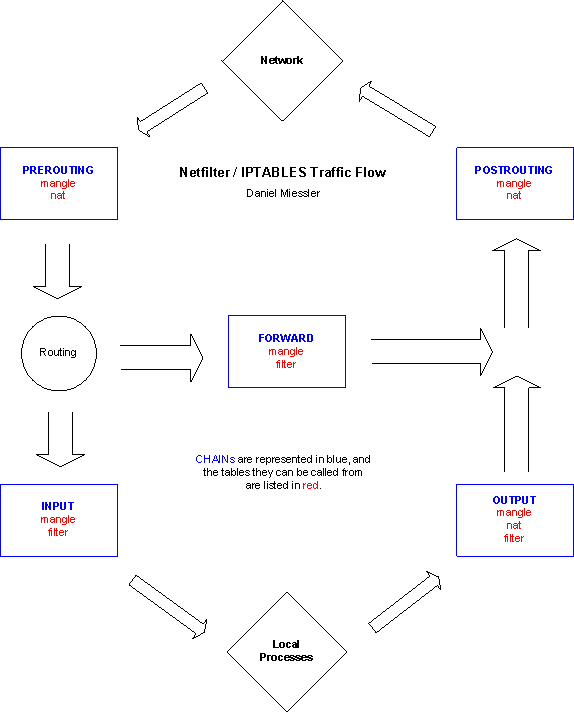
# Tres nivells

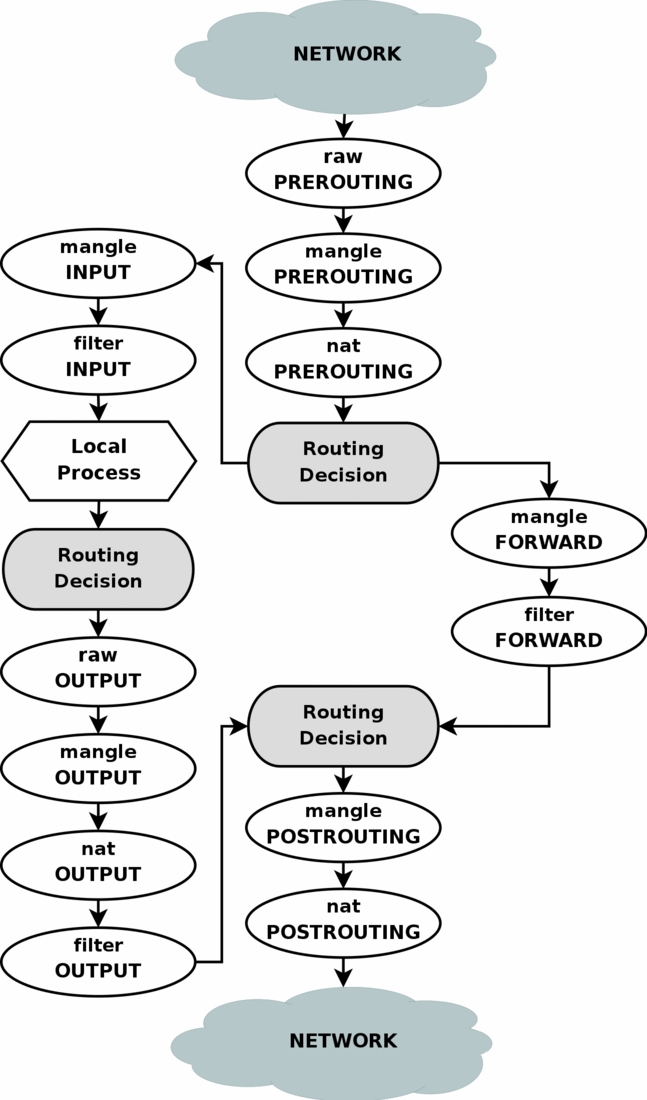
* **Taules**
* **Cadenes**
  + Pre-definidas (built-in)
  + Definidas por el usuario
* **Regles:** Dicen que hacer con un paquete que coincide

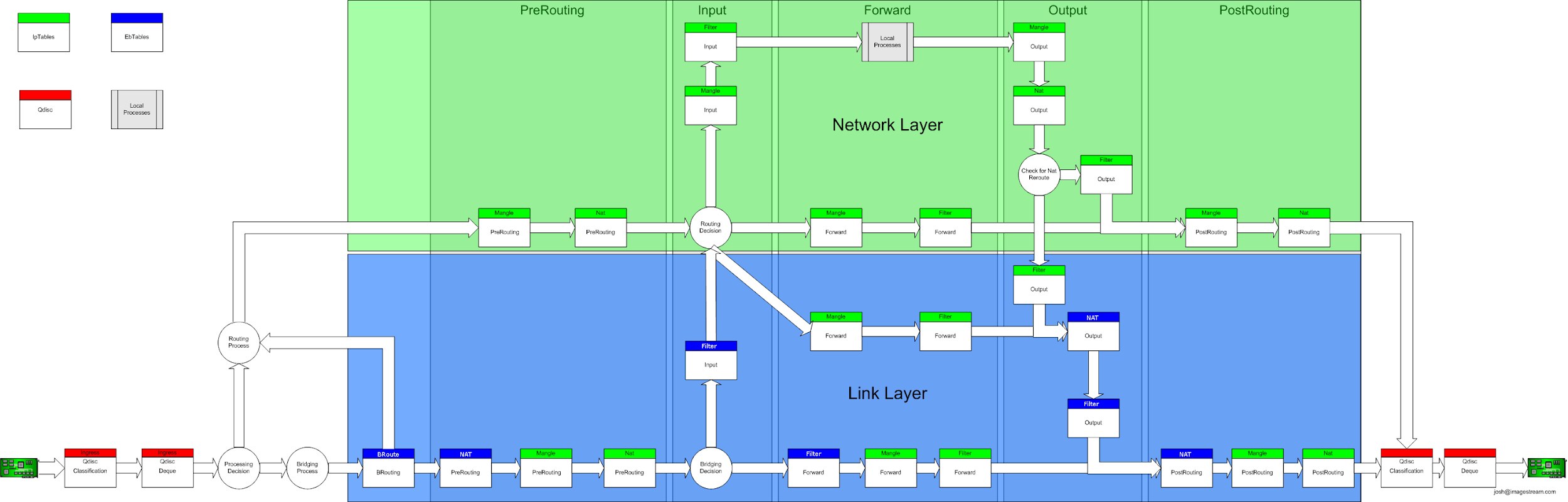
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TAULES** | **CADENES** | | | | | **ACCIONS** |
|  | **INPUT** | **OUTPUT** | **FORWARD** | **PREROUTING** | **POSTROUTING** |  |
| **filter**  Filtra paquets | X | X | X |  |  | **DROP**: llença, descarta, no informa del rebuig  **REJECT**: rebutja i informa del rebuig  **ACCEPT**: deixa passar |
| **NAT**  Permet canviar adreces o ports del paquet |  | X |  | X | X | **MASQUERADE**: enmascara una ip interna, permite compartir una ip pública entre varias ips internas  **DNAT**: destination NAT. Redirige los puertos. Modifica  ip y puerto destino  **SNAT**: como MASQUERADE pero con más opciones (obliga a posar IPs internes) |
| **mangle**  Per marcar paquets o fer log’s | X | X | X | X | X | **MARK**: marca el paquete para poderlo usar en otras reglas o  para policy routing o QoS  **TOS**: modifica el campo TOS (Type of Service) de la cabecera IP,  que se puede usar en otros routers para QoS o Policy Routing  **LOG**: genera un log del sistema cuando un paquete cumple esta regla |

# Diagrames de flux



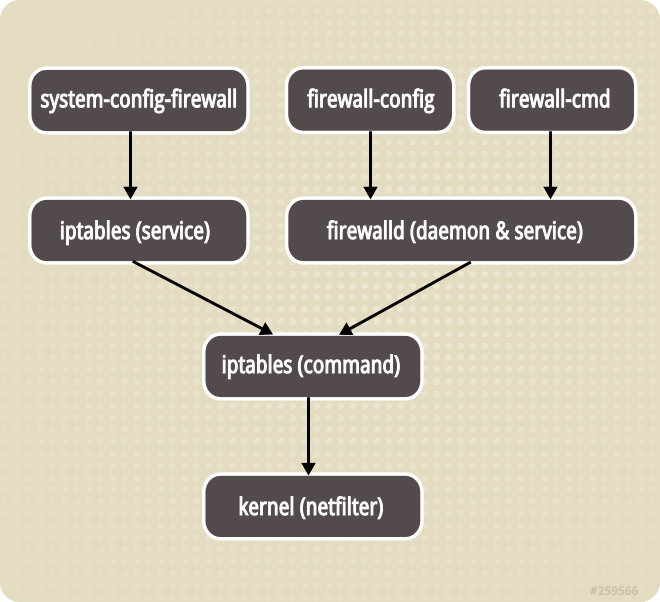


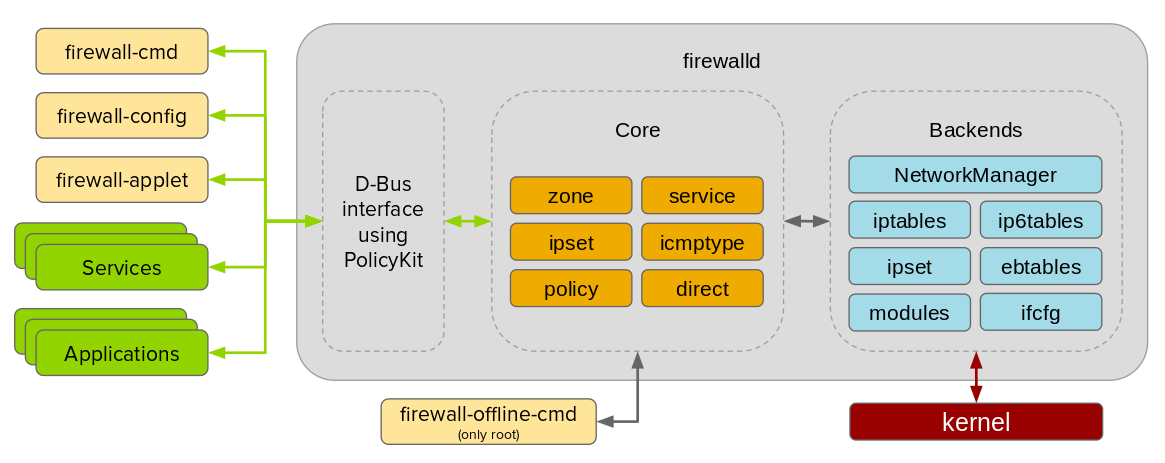




# Utilitzar iptables enlloc de firewalld

A partir del Fedora 18 **iptables** ha estat substituït per **firewalld**.

****

****

**Utilitzar iptables enlloc de firewalld**

Deshabilitem i aturem el servei **firewalld**

**# systemctl disable firewalld  
# systemctl stop firewalld**

Instal·lem el servei **iptables-services** (tant per ipv4 com per ipv6)

**# dnf -y install iptables-services**

Creem l’arxiu de configuració

**# touch /etc/sysconfig/iptables**

Iniciem els serveis

**# systemctl start iptables  
# systemctl start ip6tables**

Si volem que s’iniciïn desprès de cada reinici del servidor

**# systemctl enable iptables  
# systemctl enable ip6tables**

<https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/sec-Using_Firewalls.html>

<https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/sec-Using_Firewalls.html#sec-Using_iptables>

<https://www.atlantic.net/community/howto/fedora-20-use-iptables-instead-of-firewalld/>

# Arxius i programes relacionats

**/etc/sysconfig/iptables Conté les regles**

**/etc/sysconfig/iptables-config Conté la configuració de iptables (no de les regles)**

**iptables Comanda principal**

**iptables-save Eina per fer backups (i veure les regles actuals)**

**iptables-restore Eina per recuperar backups**

**system-config-firewall-tui Eina gràfica en mode text de configuració**

**system-config-firewall Eina gràfica de configuració**

# Com consultar l'estat actual d' iptables

Si quieres ver las reglas en el mismo formato en el cual se han de introducir

hay que utilizar la orden iptables-save (a veces no viene instalada por

defecto en la distribución). Si no contesta nada es que no hay ninguna

regla de firewall activa.

**iptables-save**

Si queremos listar las reglas en un formato con columnas y nº de línea usamos:

**iptables -L**

**iptables -t filter -n -L -v --line-numbers**

**iptables -t nat -n -L -v --line-numbers**

**iptables -t mangle -n -L -v --line-numbers**

Hay opciones que vale la pena conocer para listar las reglas:

-L: listamos las reglas,

-n: nos muestra las ips y los puertos en formato numérico

-v: nos lista los contadores

de paquetes y bytes que han cumplido esa regla (muy interesante para

ver si nuestra regla está siendo efectiva)

--line-numbers: muestra la numeración de cada regla, es útil si

queremos eliminar una regla en concreto.

Si queremos limitar el listado a una cadena, después de -L ponemos el

nombre de la cadena

# Com guardar les regles d’ iptables

Per tal que es tornin a aplicar després de reiniciar el servidor. Per defecte no es guarden.

Es guarden a:

**/etc/sysconfig/iptables**

Per guardar les actuals:

**iptables-save > /etc/sysconfig/iptables**

**service save iptables**

Si canviem aquestes línies, cada cop que aturem/reiniciem el servei es guardaran automàticament:

**/etc/sysconfig/iptables-config**

**IPTABLES\_SAVE\_ON\_STOP="yes"**

**IPTABLES\_SAVE\_ON\_RESTART="yes"**

<https://access.redhat.com/documentation/es-ES/Red_Hat_Enterprise_Linux/6/html/Security_Guide/sect-Security_Guide-IPTables-Saving_IPTables_Rules.html>

# Com exportar les regles d’un servidor per importar-les a un altre

**iptables-save > *arxiu.regles.iptables***

# 

# Com importar les regles des d’un arxiu

**iptables-restore < *arxiu.regles.iptables***

# 

# Com esborrar una regla concreta d’iptables

Primer les mostrem amb nº de línia

**iptables -t filter -n -L -v --line-numbers**

**iptables -t nat -n -L -v --line-numbers**

**iptables -t mangle -n -L -v --line-numbers**

Esborrar la 4

**iptables -D INPUT 4**

# 

# Com esborrar totes les regles

iptables -t filter -X

iptables -t filter -F

iptables -t nat -X

iptables -t nat -F

iptables -t mangle -X

iptables -t mangle -F

# Com crear una política base restrictiva

Cal executar les següent comandes:

$ sudo iptables -F  
$ sudo iptables -X  
$ sudo iptables -Z  
$ sudo iptables -t nat -F  
$ sudo iptables -P INPUT DROP  
$ sudo iptables -P OUTPUT DROP  
$ sudo iptables -P FORWARD DROP  
$ sudo iptables -t nat -P PREROUTING DROP  
$ sudo iptables -t nat -P POSTROUTING DROP

# 

# Com restaurar iptables a l'estat per defecte

Normalment l'estat per defecte és l'estat permissiu (cap restricció).

ptables -F  
iptables -F -t mangle  
iptables -X  
iptables -Z  
iptables -t nat -F  
iptables -P INPUT ACCEPT  
iptables -P OUTPUT ACCEPT  
iptables -P FORWARD ACCEPT  
iptables -t nat -P PREROUTING ACCEPT  
iptables -t nat -P POSTROUTING ACCEPT  
iptables -t mangle -P PREROUTING ACCEPT  
iptables -t mangle -P POSTROUTING ACCEPT

# Configurar iptables amb eines gràfiques

Vigileu que esborren totes les regles fetes en línia de comanda.

Utilitzar només per fer proves.

Mode text (per consola)

**system-config-firewall-tui**

Mode gràfic

**system-config-firewall**

# Com permetre que el nostre PC pugui enrutar

Per defecte només permet INPUT i OUTPUT

Així també podrem fer FORWARD (taula **filter** cadena **FORWARD**)

## De manera temporal usando procfs

Este es probablemente el metodo más usado, pero el cambio es temporal, y deberás realizarlo luego de cada que reinicies el servidor.

**echo 1 > /proc/sys/net/ipv4/ip\_forward**  
**cat /proc/sys/net/ipv4/ip\_forward**

## De manera temporal usando sysctl

sysctl, permite cambiar valores en el kernel en tiempo real, usando esta herramienta, se puede cambiar el comportamiento del kernel, y con ello habilitar el IP forward o no.

Primero veamos si esta activo o no. Ejecuta este comando como root

**sysctl -a | grep net.ipv4.ip\_forward**

Ahora, puedes colocar su valor en 1, para encender el ip forwarding

**sysctl -w net.ipv4.ip\_forward=1**

## 

## 

## De manera definitiva

Este es también un cambio temporal, si quieres que sea permanente, debes editar el archivo **/etc/sysctl.conf**

Y modificar/añadir esta línea:

**net.ipv4.ip\_forward = 1**

De este modo, esto será activado en cada momento que reinicies el servidor.

Para cargar los cambios recien realizados, sin apagar el servidor, ejecuta este comando

**sysctl -p**

# Com habilitar NAT, SNAT, MASQUERADE

NAT, també anomenat SNAT(Source NAT), o MASQUERADE

**iptables -t nat -A POSTROUTING -o enp2s0 -j MASQUERADE**

# Com habilitar DNAT

Destination NAT (NAT invers)

Si tenim un servidor web (http -80- i https -443-) a la nostra LAN i volem que es pugui accedir des de Internet

**iptables -t nat -A PREROUTING -i enp2s0 -p tcp --dport 80 -j DNAT --to****[10.10.2.10:80](http://10.10.2.10:80)**

**iptables -t nat -A PREROUTING -i enp2s0 -p tcp --dport 443 -j DNAT --to** [**10.10.2.10:443**](http://10.10.2.10:443)

# Deshabiltar el ping

Ferm que el nostre router no respongui al pings. Per evitar intents d’atac a tots els ports al veure que la IP respon, evitar atacs DoS a base de pings, …

**Opció amb iptables:**

**iptables -A INPUT -p ICMP -j DROP**

**Opció amb sys:**

Editem l’arxiu: **/proc/sys/net/ipv4/icmp\_echo\_ignore\_all**

Canviem el 0 per un 1

# LOGs

## Deixem un LOG per cada intent de connexió al port 22

# iptables -t filter -A INPUT -p tcp --dport 22 --syn -j LOG --log-prefix "iptables intent acces ssh:" --log-level 4

Per veure els logs

# journalctl -f

Donaria això: MAC origen i IP origen en blau. MAC destí i IP destí en vermell

gen 24 17:19:26 profeF2A.informatica.escoladeltreball.org kernel: iptables intent acces ssh: IN=enp2s0 OUT= MAC=74:d4:35:a1:0d:40:74:d4:35:a1:0f:89:08:00 SRC=192.168.4.3 DST=192.168.0.20 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=16820 DF PROTO=TCP SPT=33054 DPT=22 WINDOW=29200 RES=0x00 SYN URGP=0

## Deixem un LOG per cada intent de resolució DNS al port 53

# iptables -t nat -A PREROUTING -p tcp --dport 53 --syn -j LOG --log-prefix "iptables intent acces DNS:" --log-level 4  
# iptables -t nat -A PREROUTING -p udp --dport 53 -j LOG --log-prefix "iptables intent acces DNS:" --log-level 4

# DNS de caché i reenviament i facebook.com cap a la IP de google.com

## Instal·lació

**dnf -y install bind bind-utils**

## Comprovem quins arxius i binaris té

**rpm -ql bind**

## Comprovem el servei

**systemctl status named**

**systemctl start named**

**dig @127.0.0.1 google.com**

## Configurem el servidor DNS

### Editem /etc/named.conf

**Canviem la línia**

listen-on port 53 { 127.0.0.1; };

**per**

listen-on port 53 { 127.0.0.1; any; };

Així permet servir en local i per xarxa

**Canviem la línia**

allow-query { localhost; };

**per**

allow-query { localhost; any; };

Així respon peticions en local i per xarxa

**Al final de “options” afegim el que està en gris**

options {

.

.

.

forwarders {

8.8.8.8;

8.8.4.4;

};

forward only;

};

Fem un reenviament. Així les peticions que no podem resoldre localment, les resoldran en aquest cas els servidors DNS de google.

També podriem posar la ip de OPENDNS de CISCO, o similars.

**Al final de tot afegim**

zone "facebook.com" IN {

type master;

file "facebook.com.zone";

};

### 

### Creem /var/named/facebook.com.zone

cd /var/named

touch facebook.com.zone

chown root.named facebook.com.zone

**Afegim**

$TTL 86400

@ IN SOA ns1.facebook.com. root.facebook.com. (

2013042201 ;Serial

3600 ;Refresh

1800 ;Retry

604800 ;Expire

86400 ;Minimum TTL

)

; Specify our two nameservers

IN NS ns1.facebook.com.

IN NS ns2.facebook.com.

; Resolve nameserver hostnames to IP, replace with your two droplet IP addresses.

ns1 IN A 216.58.201.142

ns2 IN A 216.58.201.142

; Define hostname -> IP pairs which you wish to resolve

@ IN A 216.58.201.142

www IN A 216.58.201.142

**Nota:** El que fem es dir que facebook.com té la IP 216.58.201.14 (que és la IP de google.com)

**Per comprovar que funciona**

systemctl restart named

dig @127.0.0.1 facebook.com

dig +trace facebook.com

# Forçar DNS local

Tot i que posin els DNS de google, o altres, obliguem a utilitzar el nostre propi DNS situat a la nostra LAN.

Bó per fer un filtrat per DNS. Podem bloquejar dominis com facebook.com , twitter.com, web.whatsapp.com, … Només cal fer que el domini apunti per exemple a una ip de google. O si volem, també podriem instal·lar un servidor web a la nostra LAN y rediccionar-ho a una pàgina de “not found” o de no permesa.

**Nota:** Vigileu que si les respostes de DNS son més petites a **512 bytes** es fan per protocol **udp**, i si son de més de més de 512 bytes es fan per protocol **tcp** (p.ex. nslookup o dig).

**Manera 1 (només les peticions DNS a servidors de google)**

iptables -t nat -A PREROUTING -i enp2s0 -p udp -d 8.8.8.8 --dport 53 -j DNAT --to 192.168.1.1:53

iptables -t nat -A PREROUTING -i enp2s0 -p tcp -d 8.8.8.8 --dport 53 -j DNAT --to 192.168.1.1:53

**Manera 2 (més genèrica)**

iptables -t nat -A PREROUTING -i enp2s0 -p udp --dport 53 -j DNAT --to 192.168.1.1:53

iptables -t nat -A PREROUTING -i enp2s0 -p tcp --dport 53 -j DNAT --to 192.168.1.1:53

**Manera 3 (més genèrica i amb LOGs)**

iptables -t nat -A PREROUTING -i enp2s0 -p tcp --dport 53 --syn -j LOG --log-prefix "iptables intent acces DNS:" --log-level 4  
iptables -t nat -A PREROUTING -i enp2s0 -p udp --dport 53 -j LOG --log-prefix "iptables intent acces DNS:" --log-level 4

iptables -t nat -A PREROUTING -i enp2s0 -p udp --dport 53 -j DNAT --to 192.168.1.1:53

iptables -t nat -A PREROUTING -i enp2s0 -p tcp --dport 53 -j DNAT --to 192.168.1.1:53

# Reenviar les peticions per ssh cap al servidor amb iptables a un altre PC

Servidor amb iptables: 192.168.4.21

PC on ho reeenviem: 192.168.4.16

iptables -t nat -A PREROUTING -i enp2s0 -p tcp -d 192.168.4.21 --dport 22 --syn -j LOG --log-prefix "iptables intent acces ssh:" --log-level 4

iptables -t nat -A PREROUTING -i enp2s0 -p tcp -d 192.168.4.21 --dport 22 -j DNAT --to 192.168.4.16:22

# 

# QoS

<http://lartc.org/howto/lartc.cookbook.fullnat.intro.html>

<http://lartc.org/howto/lartc.ratelimit.single.html>

<http://lartc.org/howto/lartc.cookbook.ultimate-tc.html>

<http://etxea.net/docu/qos/qos.html>

<http://www.malditonerd.com/howto-qos-calidad-de-servicio-facil-en-linux-3/>

# Mòdul connection tracking

Connection tracking hace referencia al seguimiento de las conexiones

de capa 4 que atraviesan el router. Cuando un host cliente se comunica con un

servidor se abre una conexión que queda establecida por una combinación

de direcciones ips origen y destino y puertos (TCP o UDP) origen y destino.

Esta combinación de ips y puertos también se conoce como **socket**.

El cliente y el servidor manejan esas conexiones y el kernel lleva un

control del estado de estas conexiones que podemos consultar con la orden

netstat. Por ejemplo, con **netstat -tanp** nos lista las conexiones

activas y el proceso asociado.

Este seguimiento de conexiones es muy importante en los routers para

poder hacer un seguimiento del NAT (cambio de puertos y direcciones)

y también **para no inspeccionar cada paquete** (**con el gasto de cpu asociado** si el filtro es complejo) y analizar los filtros sólo al inicio de la conexión. Una vez la conexión se ha establecido ya se confía en ella y no es necesario analizar cada paquete de la conexión.

## Instal·lar el mòdul

dnf -y install conntrack-tools

Comprovem que el mòdul està actiu

modprobe nf\_conntrack\_ipv4

**Exemple**

Imaginem que només volem donar accés a un servidor web (port 80) que està a

192.168.22.10 des de la xarxa 192.168.100.0/24

**iptables -A FORWARD -m conntrack \**

**--ctstate RELATED,ESTABLISHED \**

**-j ACCEPT**

**iptables -t filter -A FORWARD**

**-d 192.168.22.10 -s 192.168.100.0/24 \**

**-p tcp --dport 80 \**

**-j ACCEPT**

**iptables -t filter -A FORWARD**

**-d 192.168.22.10 \**

**-p tcp --dport 80 \**

**-j DROP**

La primera regla analitza si la connexió s’ha establert, aleshores, no cal mirar la segona ni la tercera.

Si tenim una nova petició de connexió la primera regla no es compleix i s’avalua la segona. Si la ip es de la família 192.168.100.0/24 es deixa passar i si no pertany a aquesta família de ips actua la tercera regla y descarta el paquet (DROP)

El mòdul de “connection tracking” en aquest cas permet que només s’avaluin les regles al moment d’establir la connexió, un cop establerta la connexió ja no es tornen a avaluar les regles.

**Si volem comprovar-ho:**

conntrack -L | grep dport=80

“tcp 6 51 TIME\_WAIT src=192.168.100.32 **dst=192.168.22.10** sport=59464 **dport=80** src=192.168.22.10 dst=192.168.100.32 sport=80 dport=59464 [ASSURED] mark=0 secctx=system\_u:object\_r:unlabeled\_t:s0 use=1”

# Mòduls iprange/multiport i subnetting per reduïr les regles

Els PCs amb IP 10.0.0.101/8 i 10.0.0.102/8 només poden navegar pel port 80 (http) i 443 (https) del PC amb IP 10.10.2.10

La resta de peticions estan denegades.

## Reduïr regles amb subnetting

10.0.0.101 -> 0000 1010 . 0000 0000 . 0000 0000 . 0110 0101

10.0.0.101 -> 0000 1010 . 0000 0000 . 0000 0000 . 0110 0111

Màscara real -> 1111 1111 . 0000 0000 . 0000 0000 . 0000 0000 /8

Màscara subnetting -> 1111 1111 . 1111 1111 . 1111 1111 . 1111 1100 /30

iptables -t filter -A FORWARD -d 10.10.2.10 -s 10.0.0.100/30 -p tcp --dport 80 -j ACCEPT

iptables -t filter -A FORWARD -d 10.10.2.10 -s 10.0.0.100/30 -p tcp --dport 443 -j ACCEPT

iptables -t filter -A FORWARD -s 10.0.0.100/30 -o eth2 -j DROP

## Amb mòduls iprange i multiport

iptables -t filter -A FORWARD -d 10.10.2.10 \

-p tcp

-m iprange --src-range 10.0.0.101-10.0.0.102 \

-m multiport --destination-ports 80,443 \

-j ACCEPT

iptables -t filter -A FORWARD -m iprange --src-range 10.0.0.101-10.0.0.102 \

-o eth2 \

-j DROP

# Mòdul mac (filtratge per mac address)

iptables -t filter -A FORWARD -d 10.10.2.10 \

-p tcp

-m mac --mac-source 00:11:22:33:44:55

-j ACCEPT

# Mòdul time (filtratge per franja horària)

iptables -m time --timestart HORA1 --timestop HORA2 -j ACTION (Formato 24h HH:MM)

# Mòdul limit

Host-based firewalls are an important component in the security practitioners tool chain. Layering security mechanisms for a defense-in-depth stance provides redundancy to protect valuable assets. As I was reminded the other day, threat vectors connect threats to vulnerabilities, but ultimately exploiting a vulnerability is about access to, or destruction of an asset. Patching for known vulnerabilities is part of the constant vigilance required, but unknown vulnerabilities still remain. When one *up-stream* security mechanism fails, or is bypassed by some means, another should still remain behind it so that reaching the asset is more difficult than defeating a single security mechanism.

Of course, no network-based security measure protects assets behind application software — but this is another topic, for another day. Firewalls limit the attack surface to specific services and applications and may also help prevent some kinds of abuses.

The Linux host-based firewall, iptables, has a wealth of modules for various purposes. One particularly useful one is the *limits* module. Let’s explore how this can be used to protect against some basic attacks.

### **Syntax and Example**

First, a quote (with annotations in blue) from the Linux IP filtering HowTo:

**limit**

This module must be explicitly specified with `-m limit’ or `–match limit’. It is used to restrict the rate of matches, such as for suppressing log messages. It will only match a given number of times per second (by default 3 matches per hour, with a burst of 5). It takes two optional arguments:

**–limit**

followed by a number; specifies the maximum average number of matches to allow per second. The number can specify units explicitly, using `/second’, `/minute’, `/hour’ or `/day’, or parts of them (so `5/second’ is the same as `5/s’).

**–limit-burst**

followed by a number, indicating the maximum burst before the above limit kicks in.

This match can often be used with the LOG target to do rate-limited logging. To understand how it works, let’s look at the following rule, which logs packets with the default limit parameters:

# iptables -A FORWARD -m limit -j LOG

The first time this rule is reached, the packet will be logged; in fact, since the default burst is 5, the first five packets will be logged. After this, it will be twenty minutes before a packet will be logged from this rule (again, the default –limit is three per hour), regardless of how many packets reach it. Also, every twenty minutes which passes without matching a packet, one of the burst will be regained; if no packets hit the rule for 100 minutes, the burst will be fully recharged; back where we started.

Note: you cannot currently create a rule with a recharge time greater than about 59 hours, so if you set an average rate of one per day, then your burst rate must be less than 3.

You can also use this module to avoid various denial of service attacks (DoS) with a faster rate to increase responsiveness.

## Syn-flood protection:

# iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT

## Furtive port scanner:

# iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST \

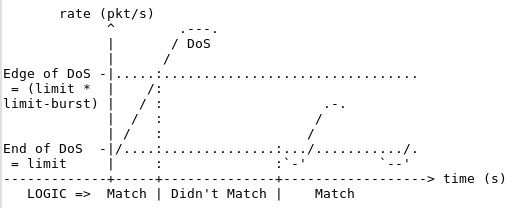
-m limit --limit 1/s -j ACCEPT

## Ping of death:

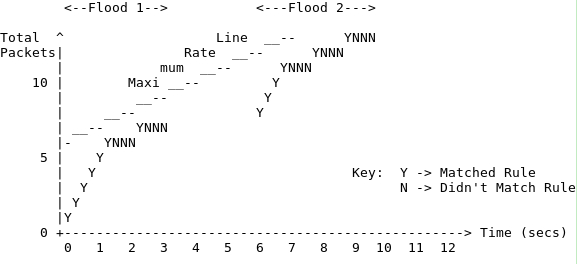
# iptables -A FORWARD -p icmp --icmp-type echo-request -m limit \

--limit 1/s -j ACCEPT

This module works like a “hysteresis door”, as shown in the graph below.



Say we say match one packet per second with a five packet burst, but packets start coming in at four per second, for three seconds, then start again in another three seconds.

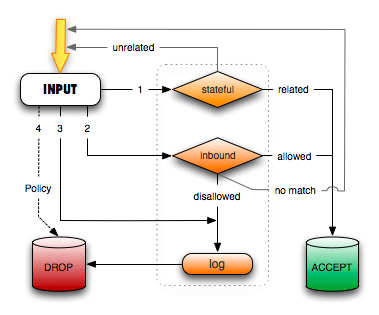


You can see that the first five packets are allowed to exceed the one packet per second, then the limiting kicks in. If there is a pause, another burst is allowed but not past the maximum rate set by the rule (1 packet per second after the burst is used).

### **Other Examples**

While the above examples are technically correct, I prefer to explicitly specify the limit and burst rates, rather than relying on default values. One important concept to understand is rule matching. All iptables rules are examined in order, and the first match wins. However, a limits rule does **not** act as a match! You might have a limits rule, followed by one that accepts packets. This can be confusing. If you limit packets, it should be followed by a rule that drops (or logs, followed by rules that drop) additional packets of the same *“type*“. This can be implicit, i.e. you use a policy which drops packets not processed by matching rules.

One thing of note regarding the following examples… there are many ways to organize the rule chains. I prefer to denote them by the direction of flow, but you may organize them anyway that makes sense to you. Also, I like to keep the number of rules on the default chains (e.g. INPUT) as small as possible, and shunt categories of traffic that might be acceptable to custom rule chains, so that non-qualifying traffic quickly follows the policy, which is set to drop. The flow for the example INPUT chain (the chain for local traffic is not shown) is as follows:



Traffic flow through the example INPUT chain

The custom chains are in the dashed-line box in orange. All traffic eventually falls into one of two *buckets* — drop or accept.

Now let’s look at a very practical example. Let’s say that you have a mail server that should only accept TCP traffic for the SSH and SMTP protocols, along with ICMP echo requests. Additionally, you’d like to protect it against TCP SYN floods, inordinate amounts of incoming UDP traffic (if you allow any), and limit rates for ICMP traffic. We’ll put together a script you can use during system startup — you could include iptables rules as part of network startup, but I prefer a separate *“service”* because you can start/stop it for testing, making changes, and debugging any issues. You can retrieve [the entire script here](http://www.box.net/shared/p728vm9wgc).

First we set up some variables to make our script easier to understand (notice the options available for specifying single hosts and subnets; single hosts can be specified with or without CIDR notation). Most match parts of our fictitious network, but others match broadcast or multicast address spaces. Our mail server is a gateway to an internal Exchange server.

IPT="/sbin/iptables"

lan="10.1.0.0/16"

dmz="192.168.100.0/24"

dns1="192.168.100.2/32"

mail="192.168.100.10/32"

exch="10.1.0.10/32"

local="192.168.100.10"

bcast1="192.168.100.255"

bcast2="255.255.255.255"

mcast="224.0.0.0/8"

Now the standard case statement for start/stop/restart arguments to the script.

case "$1" in

start)

Now we flush any existing rules and delete all chains.

echo "Setting iptables policies..."

# Flush old rules, delete chains

$IPT -F

$IPT -X

A drop policy will force all packets not explicitly matched by a rule to be dropped.

# Construct the default policy -- drop everything

echo " Setting policies..."

$IPT -P INPUT DROP

$IPT -P OUTPUT DROP

$IPT -P FORWARD DROP

We establish five chains: one each for local traffic, stateful matches, inbound traffic, outbound traffic, and a special one for logging anything of interest:

$IPT -N local

$IPT -N stateful

$IPT -N inbound

$IPT -N outbound

$IPT -N log

Now we set up stateful inspection for TCP and UDP. We then add rules to the INPUT chain to jump here when we find appropriate traffic. If traffic does not match an existing flow, we return to the INPUT chain for further processing.

# Statefulness

echo " Setting stateful inspection..."

$IPT -A stateful -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT

$IPT -A stateful -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT

$IPT -A INPUT -p tcp -j stateful

$IPT -A INPUT -p udp -j stateful

$IPT -A OUTPUT -p tcp -j stateful

$IPT -A OUTPUT -p udp -j stateful

Now we add our rate limiting rules, first for TCP SYN packets, and then for new UDP flows. If the prescribed limits are exceeded, subsequent packets are logged and dropped. Of course, further on we’ll rate limit the logging too. Again, our order is 1) limit, 2) log, 3) drop.

# Inbound traffic to accept

echo " Setting rules for inbound traffic..."

# Rate limiting rules:

# Set rates with burst for new conns, then log/drop the excess

$IPT -A inbound -p tcp --syn -m limit --limit 4/s --limit-burst 10 -j ACCEPT

$IPT -A inbound -p tcp --syn -j LOG

$IPT -A inbound -p tcp --syn -j DROP

$IPT -A inbound -p udp -m state --state NEW -m limit \

--limit 1/s --limit-burst 5 -j ACCEPT

$IPT -A inbound -p udp -m state --state NEW -j LOG

$IPT -A inbound -p udp -m state --state NEW -j DROP

$IPT -A inbound -p tcp --dport 25 -j ACCEPT

$IPT -A inbound -p tcp -s $lan --dport 22 -j ACCEPT

$IPT -A inbound -p icmp --icmp-type 8 -m limit \

--limit 1/s --limit-burst 5 -j ACCEPT

$IPT -A inbound -p icmp --icmp-type 11 -m limit \

--limit 1/s --limit-burst 5 -j ACCEPT

We now add the appropriate rules to the INPUT chain to jump the inbound chain as appropriate.

$IPT -A INPUT -p tcp --dport 25 -j inbound

$IPT -A INPUT -p tcp -s $lan -j inbound

$IPT -A INPUT -p icmp -j inbound

# Drop these to avoid logging them  
$IPT -A INPUT -p udp -d $bcast1 -j DROP  
$IPT -A INPUT -p udp -d $bcast2 --dport 67 -j DROP  
$IPT -A INPUT -d $mcast -j DROP

Notice that since we’re not actually accepting any UDP traffic, we don’t actually add a rule to jump to the inbound chain. If we do in the future, we can add the INPUT chain rule, and appropriate inbound chain rules; we have the rate limiting rules already in place. Also notice that we chose to simply drop some kinds of broadcast traffic to avoid logging these expected packets.

Now let’s look at the logging rules (the rest of the script is left to the reader to work through and understand):

# Log all dropped packets

echo " Setting logging rules..."

$IPT -A log -j LOG -m limit --limit 4/minute --limit-burst 3 \

--log-level info --log-prefix "IPTABLES: "

$IPT -A log -j DROP

$IPT -A INPUT -j log

$IPT -A OUTPUT -j log

We’ve limited logging to four entries per minute, but with a burst of 3. Again, if we consume the burst limit, we gain back one entry every 15 seconds. If nothing is logged for 45 seconds, the burst is fully re-charged.

### **Conclusion**

Adding a host-based firewall adds to a defense-in-depth stance. It does add complexity (you will have the similar rules in several places, making maintenance, testing, and debugging more difficult), but if your risk profile warrants it, using a firewall like iptables is helpful.

# Mòdul recent

<https://we.riseup.net/stefani/iptables-recent-module-and-hit-limits>

## Anti SSH-Bruteforce

### **Your default INPUT chain policy is ACCEPT**

This is the version that is found online, typically under tags such as Brute-force.

one adds lines to ones iptables

iptables -N SSHSCAN  
  
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 22 -m state --state NEW -j SSHSCAN   
  
iptables -A SSHSCAN -m recent --set --name SSH --rsource   
iptables -A SSHSCAN -m recent --update --seconds 3600 --hitcount 5 --name SSH --rsource -j LOG --log-prefix "Anti SSH-Bruteforce: " --log-level 6   
iptables -A SSHSCAN -m recent --update --seconds 3600 --hitcount 5 --name SSH --rsource -j DROP

### **Your default INPUT chain policy is DROP**

This is a variation of the above, but takes into account the DROP policy. It is a small change, but necessary if you really want to be able to log in via ssh.

iptables -N SSHSCAN  
  
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 22 -m state --state NEW -j SSHSCAN  
  
iptables -A SSHSCAN -m recent --set --name SSH --rsource  
iptables -A SSHSCAN -m recent --update --seconds 3600 --hitcount 5 --name SSH --rsource -j LOG --log-prefix "Anti SSH-Bruteforce: " --log-level 6  
iptables -A SSHSCAN -m recent --update --seconds 3600 --hitcount 5 --name SSH --rsource -j LogDrp  
iptables -A SSHSCAN -j ACCEPT

# Canviar el gateway del equip que farà de client

Fer backup de **/etc/sysconfig/network-scripts/ifcfg-enp2s0** al vostre home

**Canviar la porta d’enllaç**

Si el client és 192.168.4.11

Si el PC que farà de router és 192.168.4.12

Eliminar la porta d’enllaç per defecte

ip r d default

Afegir la nova porta d’enllaç

ip r a default via 192.168.4.12

Retornar a les dades originals IP/MASC/GW/DNS

ip a f dev eth0

dhclient

Retornar el backup del vostre home a **/etc/sysconfig/network-scripts/ifcfg-enp2s0**

Executar

nmcli con reload

# Software malicioso y suplantación de direcciones IP

Reglas más elaboradas pueden ser creadas para que controlen el acceso a subredes específicas, o incluso para nodos específicos, dentro de la LAN. Puede también restringir ciertas aplicaciones o programas de carácter dudoso como troyanos, gusanos, y demás virus cliente/servidor, y evitar que entren en contacto con sus servidores.

Por ejemplo, algunos **troyanos** examinan redes para ver los servicios en los puertos **31337** a **31340** (llamados los **puertos *elite*** en la terminología de craqueo).

Dado que no hay servicios legítimos que se comunican vía estos puertos no estándares, su bloqueo puede disminuir efectivamente las posibilidades de que nodos infectados en su red se comuniquen con sus servidores maestros remotos.

Las siguientes reglas eliminan todo el tráfico TCP que intenta usar el puerto 31337:

**iptables -A OUTPUT -o eth0 -p tcp --dport 31337 --sport 31337 -j DROP  
iptables -A FORWARD -o eth0 -p tcp --dport 31337 --sport 31337 -j DROP**

También se puede bloquear conexiones salientes que intenten suplantar los rangos de direcciones IP privadas para infiltrarse en su LAN.

Por ejemplo, si su red usa el rango 192.168.1.0/24, se puede diseñar una regla que instruya al dispositivo de red del lado de Internet (por ejemplo, eth0) para que descarte cualquier paquete en ese dispositivo con una dirección en el rango IP de su red local.

Dado que se recomienda rechazar paquetes reenviados como una política predeterminada, cualquier otra dirección IP mentida al dispositivo del lado externo (eth0) se rechaza automáticamente.

**iptables -A FORWARD -s 192.168.1.0/24 -i eth0 -j DROP**

**Nota**

Existe una distinción entre los destinos DROP y REJECT cuando se trabaja con reglas *agregadas*.

El destino RECHAZAR niega acceso y regresa un error de conexión denegada a los usuarios que intenten conectarse al servicio. El destino ABANDONAR, como su nombre lo indica, abandona el paquete sin previo aviso.

Los administradores pueden usar su propia discreción cuando usen estos destinos. Sin embargo, para evitar la confusión del usuario e intentos de continuar conectando, el destino REJECT es recomendado.

<https://docs.fedoraproject.org/es-ES/Fedora/13/html/Security_Guide/sect-Security_Guide-Firewalls-Malicious_Software_and_Spoofed_IP_Addresses.html>

# Comencem a provar amb filter + INPUT/OUTPUT

De moment només utilitzarem les cadenes **INPUT**(cap al nostre PC) i **OUTPUT**(surt del nostre PC).

A la següent fase farem enrutament (taula **filter** cadena **FORWARD**)

## Per fer proves treballarem amb **ssh(port22)** i **nginx(web server port 80)**

Els tindrem disabled i els iniciarem només per fer proves

**-ssh** ja està instal·lat

**systemctl disable sshd**

**systemctl start sshd**

**-nginx** cal fer:

**dnf -y install nginx**

**systemctl disable nginx**

**systemctl start nginx**

## Com mirem quins ports hi ha oberts

### Al nostre equip

**nmap localhost**

**telnet localhost port** (telnet localhost 22)

**netstat -putona**

**nc -zv localhost port** (nc -zv localhost 22)

**lsof -i -P -n**

### A un altre equip de la xarxa

**nmap nom/ip** (nmap 192.168.4.10)

**telnet nom/ip port** (telnet 192.168.4.10 22)

**nc -zv nom/ip port** (nc -zv 192.168.4.10 22)

## Proves OUTPUT

### Prova DROP de tot

**Esborrem totes les regles**

iptables -t filter -X

iptables -t filter -F

iptables -t nat -X

iptables -t nat -F

iptables -t mangle -X

iptables -t mangle -F

**Provem que podem navegar per LAN (ip del PC d’un company)**

telnet 192.168.4.10 80

Si

**Provem que podem navegar per Internet**

telnet google.com 80

Si

telnet google.com 443

Si

**Provem que podem fer ssh per LAN (ip del PC d’un company)**

telnet 192.168.4.10 22

Si

**Afegim**

iptables -t filter -A OUTPUT -j DROP

**Comprovem les regles i ordre**

iptables -n -L -v --line-numbers

**Provem que podem navegar per LAN (ip del PC d’un company)**

telnet 192.168.4.10 80

No ens deixa, perquè hem posat una regla que fa DROP per tot paquet que surt

**Provem que podem navegar per Internet**

telnet google.com 80

No

telnet google.com 443

No

**Provem que podem fer ssh per LAN (ip del PC d’un company)**

telnet 192.168.4.10 22

No

### Prova DROP de tot excepte navegació pel port 80 (web http)

**Esborrem totes les regles**

iptables -t filter -X

iptables -t filter -F

iptables -t nat -X

iptables -t nat -F

iptables -t mangle -X

iptables -t mangle -F

**Provem que podem navegar per LAN (ip del PC d’un company)**

telnet 192.168.4.10 80

Si

**Provem que podem navegar per Internet**

telnet google.com 80

Si

telnet 216.58.210.142 80 (la ip de google.com)

Si

telnet google.com 443

Si

**Provem que podem fer ssh per LAN (ip del PC d’un company)**

telnet 192.168.4.10 22

Si

**Afegim**

iptables -t filter -A OUTPUT -p tcp --dport 80 -j ACCEPT

iptables -t filter -A OUTPUT -j DROP

**Nota**: L’ordre de les regles importa, si ho fesim al revés, sempre faria un DROP i mai no arrivaria a avaluar la segona regla

**iptables -t filter -A OUTPUT -j DROP**

**iptables -t filter -A OUTPUT -p tcp --dport 80 -j ACCEPT**

**Comprovem les regles i ordre**

iptables -n -L -v --line-numbers

**Provem que podem navegar per LAN (ip del PC d’un company)**

telnet 192.168.4.10 80

Si

**Provem que podem navegar per Internet**

telnet google.com 80

No perquè falla la resolució DNS (udp port 53)

telnet 216.58.210.142 80 (la ip de google.com)

Si

telnet google.com 443

No

**Provem que podem fer ssh per LAN (ip del PC d’un company)**

telnet 192.168.4.10 22

No

### Prova DROP de tot excepte l’accés a una ip de LAN

Només podem accedir a tots els serveis del PC amb IP 192.168.4.10

**Esborrem totes les regles**

iptables -t filter -X

iptables -t filter -F

iptables -t nat -X

iptables -t nat -F

iptables -t mangle -X

iptables -t mangle -F

**Provem que podem navegar per LAN (ip del PC d’un company)**

telnet 192.168.4.10 80

Si

**Provem que podem navegar per Internet**

telnet google.com 80

Si

telnet 216.58.210.142 80 (la ip de google.com)

Si

telnet google.com 443

Si

**Provem que podem fer ssh per LAN (ip del PC d’un company)**

telnet 192.168.4.10 22

Si

**Afegim**

iptables -t filter -A OUTPUT -d 192.168.4.10 -j ACCEPT

iptables -t filter -A OUTPUT -j DROP

**Comprovem les regles i ordre**

iptables -n -L -v --line-numbers

**Provem que podem navegar per LAN (ip del PC d’un company)**

telnet 192.168.4.10 80

Si

**Provem que podem navegar per Internet**

telnet google.com 80

No

telnet 216.58.210.142 80 (la ip de google.com)

No

telnet google.com 443

No

**Provem que podem fer ssh per LAN (ip del PC d’un company)**

telnet 192.168.4.10 22

Si

### Exemple DROP de tots els PCs excepte els que pertanyen a una xarxa determinada

Cap PC pot accedir a res excepte els de la xarxa 192.168.4.0/24 que tenen accés a tot

**Afegim**

iptables -t filter -A OUTPUT -s 192.168.4.0/24 -j ACCEPT

iptables -t filter -A OUTPUT -j DROP

### Exemple DROP de tots excepte cap a els PCs que pertanyen a una xarxa determinada

Tots els PCs només poden accedir als serveis dels PCs de la xarxa 192.168.4.0/24

**Afegim**

iptables -t filter -A OUTPUT -d 192.168.4.0/24 -j ACCEPT

iptables -t filter -A OUTPUT -j DROP

### Prova DROP de tot excepte navegació pel port 80 a una ip de LAN

Només podem navegar pel port 80 de la IP del PC del nostre company (suposem 192.168.4.10)

**Esborrem totes les regles**

iptables -t filter -X

iptables -t filter -F

iptables -t nat -X

iptables -t nat -F

iptables -t mangle -X

iptables -t mangle -F

**Provem que podem navegar per LAN (ip del PC d’un company)**

telnet 192.168.4.10 80

Si

**Provem que podem navegar per Internet**

telnet google.com 80

Si

telnet 216.58.210.142 80 (la ip de google.com)

Si

telnet google.com 443

Si

**Provem que podem fer ssh per LAN (ip del PC d’un company)**

telnet 192.168.4.10 22

Si

**Afegim**

iptables -A OUTPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

iptables -t filter -A OUTPUT -d 192.168.4.10 -p tcp --dport 80 -j ACCEPT

iptables -t filter -A OUTPUT -j DROP

**Nota**: Hem afegit una regla amb “**Connection tracking**” que només controla el primer paquet (quan estableix la connexió). La resta s’avalua més ràpid.

En aquest cas pel primer paquet s’avaluen les dues primeres regles.

Per la resta de paquets del mateix PC només s’avalua la primera regla

**Si volem comprovar-ho:**

dnf -y install conntrack-tools

conntrack -L | grep dport=80

“tcp 6 51 TIME\_WAIT src=192.168.4.32 **dst=192.168.4.10** sport=59464 **dport=80** src=192.168.4.10 dst=192.168.4.32 sport=80 dport=59464 [ASSURED] mark=0 secctx=system\_u:object\_r:unlabeled\_t:s0 use=1”

**Comprovem les regles i ordre**

iptables -n -L -v --line-numbers

**Provem que podem navegar per LAN (ip del PC d’un company)**

telnet 192.168.4.10 80

Si

**Provem que podem navegar per Internet**

telnet google.com 80

No

telnet 216.58.210.142 80 (la ip de google.com)

No

telnet google.com 443

No

**Provem que podem fer ssh per LAN (ip del PC d’un company)**

telnet 192.168.4.10 22

No

### Prova DROP de tot excepte navegació pel port 80 a una ip de LAN, només des d’una xarxa concreta

Només podem navegar pel port 80 de la IP del PC del nostre company (suposem 192.168.4.10).

Els únics PCs que poden fer-ho son de la xarxa 192.168.4.0/24

**Esborrem totes les regles**

iptables -t filter -X

iptables -t filter -F

iptables -t nat -X

iptables -t nat -F

iptables -t mangle -X

iptables -t mangle -F

**Provem que podem navegar per LAN (ip del PC d’un company)**

telnet 192.168.4.10 80

Si

**Provem que podem navegar per Internet**

telnet google.com 80

Si

telnet 216.58.210.142 80 (la ip de google.com)

Si

telnet google.com 443

Si

**Provem que podem fer ssh per LAN (ip del PC d’un company)**

telnet 192.168.4.10 22

Si

**Afegim**

iptables -A OUTPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

iptables -t filter -A OUTPUT -s 192.168.4.0/24 -d 192.168.4.10 -p tcp --dport 80 -j ACCEPT

iptables -t filter -A OUTPUT -j DROP

**Nota**: Hem afegit una regla amb “**Connection tracking**” que només controla el primer paquet (quan estableix la connexió). La resta s’avalua més ràpid.

En aquest cas pel primer paquet s’avaluen les dues primeres regles.

Per la resta de paquets del mateix PC només s’avalua la primera regla

**Si volem comprovar-ho:**

dnf -y install conntrack-tools

conntrack -L | grep dport=80

“tcp 6 51 TIME\_WAIT src=192.168.4.32 **dst=192.168.4.10** sport=59464 **dport=80** src=192.168.4.10 dst=192.168.4.32 sport=80 dport=59464 [ASSURED] mark=0 secctx=system\_u:object\_r:unlabeled\_t:s0 use=1”

**Comprovem les regles i ordre**

iptables -n -L -v --line-numbers

**Provem que podem navegar per LAN (ip del PC d’un company)**

telnet 192.168.4.10 80

Si

**Provem que podem navegar per Internet**

telnet google.com 80

No

telnet 216.58.210.142 80 (la ip de google.com)

No

telnet google.com 443

No

**Provem que podem fer ssh per LAN (ip del PC d’un company)**

telnet 192.168.4.10 22

No

### Prova DROP de tot excepte navegació web local i per Internet

**Cal permetre**

Resolució DNS

port 53/udp

Navegació web

port 80/tcp per http

port 443/tcp per https

**Esborrem totes les regles**

iptables -t filter -X

iptables -t filter -F

iptables -t nat -X

iptables -t nat -F

iptables -t mangle -X

iptables -t mangle -F

**Provem que podem navegar per LAN (ip del PC d’un company)**

telnet 192.168.4.10 80

Si

**Provem que podem navegar per Internet**

telnet google.com 80

Si

telnet google.com 443

Si

**Provem que podem fer ssh per LAN (ip del PC d’un company)**

telnet 192.168.4.10 22

Si

**Afegim**

iptables -t filter -A OUTPUT -p tcp --dport 80 -j ACCEPT

iptables -t filter -A OUTPUT -p tcp --dport 443 -j ACCEPT

iptables -t filter -A OUTPUT -p udp --dport 53 -j ACCEPT

iptables -t filter -A OUTPUT -j DROP

**Comprovem les regles i ordre**

iptables -n -L -v --line-numbers

**Provem que podem navegar per LAN (ip del PC d’un company)**

telnet 192.168.4.10 80

Si

**Provem que podem navegar per Internet**

telnet google.com 80

Si

telnet google.com 443

Si

**Provem que podem fer ssh per LAN (ip del PC d’un company)**

telnet 192.168.4.10 22

No

## Exercicis filter + INPUT

-Deshabilitar el ping cap a la targeta de xarxa de la IP pública del nostre router

## Exercicis NAT + MASQUERADE

-Configureu dos PCs que facin de clients de LAN amb gateway el vostre PC que farà de router

-Activeu l’opció de poder enrutar de manera temporal

-Afegir la regla de NAT MASQUERADE al vostre PC router.

-Proveu que **accedeixen** als serveis de LAN/Internet amb el vostre PC router

-Esborreu totes les regles

-Proveu que **no poden accedir** als serveis de LAN/Internet amb el vostre PC router

## Exercicis NAT + DNAT

-Configureu dos PCs que facin de clients de LAN amb gateway el vostre PC que farà de router

-Afegir la regla de NAT DNAT, per reenviar les peticions DNS

## Exercicis filter + OUTPUT

-ACCEPT de tot menys la navegació web local i per Internet

-Volem accedir des del nostre PC només cap al servidor web que està a 192.168.4.10/16

-Treure internet a la F2A (xarxa 192.168.4.0/16):

-Per xarxa

-Per rang d’IPs (fer servir el mòdul ????)

-Reenviar les peticions a moodle.escoladeltreball.org per LAN

-Habilitar el “Connection tracking”

# Pràctica a corregir als vostres PCs

**Per treure un 5**

1.-Configurar un PC que faci de router (de manera temporal i definitva).

2.-No ha de respondre als pings en la targeta de xarxa que connecta amb Internet.

3.-Ha de tenir connection tracking.

4.-Reenviar la navegació http 80 i https 443 des de 3 IP’s de la LAN cap a un PC de LAN (192.168.4.6:80/443)

5.-Forçar el DNS per tota la xarxa cap a un PC de LAN (192.168.4.6:53)

**Per treure un 10**

6.-Reenviar el ssh per tota la xarxa cap a un PC (passarel·la) de LAN (192.168.4.6:22)

7.-Fer un parell de scripts que treguin /posin Internet(navegació web 80/443) a un rang d’IPs

8.-Fer un parell de scripts que treguin /posin Internet(navegació web 80/443) a una llista de MACs address

9.-Fer un parell de scripts que treguin /posin Internet(navegació web 80/443) segons la hora del dia (a les nits es treu)