Networking e firewalling su GNU/Linux

Daniele Iamartino
otacon22 - at - poul.org

Corsi GNU/Linux avanzati 2014 - Amministrazione di sistema

POuL
have a lot of fun...

Sezione 2

Prima di iniziare...

Se vi può interessare, esiste un ambiente di emulazione di reti (utilizza macchine virtuali basate su *user mode linux*), chiamato **NetKit**

- Non è aggiornatissimo ma può essere **utile a scopo "didattico"** per capire alcuni argomenti e fare qualche esperimento "vero"
- Per installarlo basta che **seguite le istruzioni sul sito**, scaricate i tre archivi e li estraete con i comandi dati dati da loro (assicuratevi di estrarre come files sparsi)

Sezione 3

Verifica link e negoziazione Ethernet

Quasi tutte le schede Ethernet ci permettono di essere interrogate via software per avere alcune utili informazioni sul livello fisico:

- Se c'è il **link** (cioè se il cavo Ethernet è collegato alla scheda e a qualche altro apparato), se arriva segnale sul cavo.
- Quali sono le **capacità** della scheda in termini di velocità (10, 100 o 1000 **Mbps** tipicamente)
- Quali sono le capacità del "***link partner***" (tipicamente lo switch a cui è collegato il nostro server)
- Se l'***autonegoziazione***[1] ha funzionato o no[2].

[1] Subito dopo aver collegato il cavo alla scheda e aver ricevuto il link, la scheda tenta di inviare una serie di frame di varie dimensioni per capire quali sono le velocità supportate dal partner.

[2] Problemi di negoziazione possono causare **gravi** problemi di performance della scheda di rete

Verifica link e negoziazione Ethernet

Quasi tutte le schede Ethernet ci permettono di essere interrogate via software per avere alcune utili informazioni sul livello fisico:

- Se c'è il **link** (cioè se il cavo Ethernet è collegato alla scheda e a qualche altro apparato), se arriva segnale sul cavo.
- Quali sono le **capacità** della scheda in termini di velocità (10, 100 o 1000 **Mbps** tipicamente)
- Quali sono le capacità del "***link partner***" (tipicamente lo switch a cui è collegato il nostro server)
- Se l'***autonegoziazione***[1] ha funzionato o no[2].

[1] Subito dopo aver collegato il cavo alla scheda e aver ricevuto il link, la scheda tenta di inviare una serie di frame di varie dimensioni per capire quali sono le velocità supportate dal partner.

[2] Problemi di negoziazione possono causare **gravi** problemi di performance della scheda di rete

Verifica link e negoziazione Ethernet

Quasi tutte le schede Ethernet ci permettono di essere interrogate via software per avere alcune utili informazioni sul livello fisico:

- Se c'è il **link** (cioè se il cavo Ethernet è collegato alla scheda e a qualche altro apparato), se arriva segnale sul cavo.
- Quali sono le **capacità** della scheda in termini di velocità (10, 100 o 1000 **Mbps** tipicamente)
- Quali sono le capacità del "***link partner***" (tipicamente lo switch a cui è collegato il nostro server)
- Se l'***autonegoziazione***[1] ha funzionato o no[2].

[1] Subito dopo aver collegato il cavo alla scheda e aver ricevuto il link, la scheda tenta di inviare una serie di frame di varie dimensioni per capire quali sono le velocità supportate dal partner.

[2] Problemi di negoziazione possono causare **gravi** problemi di performance della scheda di rete

Verifica link e negoziazione Ethernet

Quasi tutte le schede Ethernet ci permettono di essere interrogate via software per avere alcune utili informazioni sul livello fisico:

- Se c'è il **link** (cioè se il cavo Ethernet è collegato alla scheda e a qualche altro apparato), se arriva segnale sul cavo.
- Quali sono le **capacità** della scheda in termini di velocità (10, 100 o 1000 **Mbps** tipicamente)
- Quali sono le capacità del "***link partner***" (tipicamente lo switch a cui è collegato il nostro server)
- Se l'***autonegoziazione***[1] ha funzionato o no[2].

[1] Subito dopo aver collegato il cavo alla scheda e aver ricevuto il link, la scheda tenta di inviare una serie di frame di varie dimensioni per capire quali sono le velocità supportate dal partner.

[2] Problemi di negoziazione possono causare **gravi** problemi di performance della scheda di rete

Verifica link e negoziazione Ethernet

Quasi tutte le schede Ethernet ci permettono di essere interrogate via software per avere alcune utili informazioni sul livello fisico:

- Se c'è il **link** (cioè se il cavo Ethernet è collegato alla scheda e a qualche altro apparato), se arriva segnale sul cavo.
- Quali sono le **capacità** della scheda in termini di velocità (10, 100 o 1000 **Mbps** tipicamente)
- Quali sono le capacità del "***link partner***" (tipicamente lo switch a cui è collegato il nostro server)
- Se l'***autonegoziazione***[1] ha funzionato o no[2].

[1] Subito dopo aver collegato il cavo alla scheda e aver ricevuto il link, la scheda tenta di inviare una serie di frame di varie dimensioni per capire quali sono le velocità supportate dal partner.

[2] Problemi di negoziazione possono causare **gravi** problemi di performance della scheda di rete

mii-tool

mii-tool è un comodo tool per verificare queste informazioni[3].

- *"mii-tool eth0"* ci mostra informazioni di base sullo **stato della scheda**:

```
eth0: negotiated 1000baseTx-FD flow-control, link ok
```

- *"mii-tool eth0 -v"* ci dà qualche informazione più **dettagliata** anche sullo stato del **partner** e le **capacità** della nostra scheda:

```
eth0: negotiated 1000baseTx-FD flow-control, link ok
  product info: vendor 00:08:18, model 54 rev 6
  basic mode:autonegotiation enabled
  basic status: autonegotiation complete, link ok
  capabilities: 1000baseT-FD 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
  advertising:  1000baseT-FD 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD flow-control
  link partner: 1000baseT-FD 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
```

- Nel caso in cui la negoziazione fallisca possiamo anche **"forzare" la scheda** ad utilizzare una certa modalità[4]

```
mii-tool --force=100baseTx-FD eth0
```

[3] Esiste anche ethtool in alternativa, che fa le stesse cose

[4] A patto di "forzarlo" anche sul partner! Altrimenti vi troverete in un mare di guai

mii-tool

mii-tool è un comodo tool per verificare queste informazioni[3].

- *"mii-tool eth0"* ci mostra informazioni di base sullo **stato della scheda**:

```
eth0: negotiated 1000baseTx-FD flow-control, link ok
```

- "*mii-tool eth0 -v*" ci dà qualche informazione più **dettagliata** anche sullo stato del **partner** e le **capacità** della nostra scheda:

```
eth0: negotiated 1000baseTx-FD flow-control, link ok
  product info: vendor 00:08:18, model 54 rev 6
  basic mode:autonegotiation enabled
  basic status: autonegotiation complete, link ok
  capabilities: 1000baseT-FD 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
  advertising:  1000baseT-FD 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD flow-control
  link partner: 1000baseT-FD 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
```

- Nel caso in cui la negoziazione fallisca possiamo anche **"forzare" la scheda** ad utilizzare una certa modalità[4]

```
mii-tool --force=100baseTx-FD eth0
```

[3] Esiste anche ethtool in alternativa, che fa le stesse cose

[4] A patto di "forzarlo" anche sul partner! Altrimenti vi troverete in un mare di guai

mii-tool

mii-tool è un comodo tool per verificare queste informazioni[3].

- *"mii-tool eth0"* ci mostra informazioni di base sullo **stato della scheda**:

```
eth0: negotiated 1000baseTx-FD flow-control, link ok
```

- "*mii-tool eth0 -v*" ci dà qualche informazione più **dettagliata** anche sullo stato del **partner** e le **capacità** della nostra scheda:

```
eth0: negotiated 1000baseTx-FD flow-control, link ok
  product info: vendor 00:08:18, model 54 rev 6
  basic mode:autonegotiation enabled
  basic status: autonegotiation complete, link ok
  capabilities: 1000baseT-FD 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
  advertising:  1000baseT-FD 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD flow-control
  link partner: 1000baseT-FD 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
```

- Nel caso in cui la negoziazione fallisca possiamo anche **"forzare" la scheda** ad utilizzare una certa modalità[4]

```
mii-tool --force=100baseTx-FD eth0
```

[3] Esiste anche ethtool in alternativa, che fa le stesse cose

[4] A patto di "forzarlo" anche sul partner! Altrimenti vi troverete in un mare di guai

mii-tool

- Ricordiamoci che una scheda ethernet potrebbe risultare senza link anche se il cavo è collegato perché deve essere prima “accesa” via software. Vediamo ora come.

ip: Un tool per dominarli tutti

- Il networking su Linux è basato sulle *interfacce* di rete, come abbiamo visto.
- Le interfacce sono dei punti di contatto con il resto del mondo e possono essere di vario tipo:
  - Interfacce **fisiche** (schede ethernet o wifi)
  - Endpoint di canali **virtuali** (Tunnel o VPN)
  - Una **unione** virtuale di due o più interfacce (bridge)
  - ...

ip: Un tool per dominarli tutti

- Il networking su Linux è basato sulle *interfacce* di rete, come abbiamo visto.
- Le interfacce sono dei punti di contatto con il resto del mondo e possono essere di vario tipo:
  - Interfacce **fisiche** (schede ethernet o wifi)
  - Endpoint di canali **virtuali** (Tunnel o VPN)
  - Una **unione** virtuale di due o più interfacce (bridge)
  - ...

ip: Un tool per dominarli tutti

- Il networking su Linux è basato sulle *interfacce* di rete, come abbiamo visto.
- Le interfacce sono dei punti di contatto con il resto del mondo e possono essere di vario tipo:
  - Interfacce **fisiche** (schede ethernet o wifi)
  - Endpoint di canali **virtuali** (Tunnel o VPN)
  - Una **unione** virtuale di due o più interfacce (bridge)
  - ...

ip: Un tool per dominarli tutti

- Il networking su Linux è basato sulle *interfacce* di rete, come abbiamo visto.
- Le interfacce sono dei punti di contatto con il resto del mondo e possono essere di vario tipo:
  - Interfacce **fisiche** (schede ethernet o wifi)
  - Endpoint di canali **virtuali** (Tunnel o VPN)
  - Una **unione** virtuale di due o più interfacce (bridge)
  - ...

ip: Un tool per dominarli tutti

- In passato, tools sparsi e vecchi: ifconfig, route, netstat, ...
- Un unico nuovo comando per operare a livello datalink e network: **ip**.
  - Il tool "ip" fa parte del pacchetto *iproute2* che dovrebbe essere preinstallato su tutte le distribuzioni Linux.
- I comandi della suite *ip* hanno tutti la stessa struttura:

```
ip [options] object command
```

ip: Un tool per dominarli tutti

- In passato, tools sparsi e vecchi: ifconfig, route, netstat, ...
- Un unico nuovo comando per operare a livello datalink e network: **ip**.
  - Il tool "ip" fa parte del pacchetto *iproute2* che dovrebbe essere preinstallato su tutte le distribuzioni Linux.
- I comandi della suite *ip* hanno tutti la stessa struttura:

```
ip [options] object command
```

ip: Un tool per dominarli tutti

- In passato, tools sparsi e vecchi: ifconfig, route, netstat, ...
- Un unico nuovo comando per operare a livello datalink e network: **ip**.
  - Il tool "ip" fa parte del pacchetto *iproute2* che dovrebbe essere preinstallato su tutte le distribuzioni Linux.
- I comandi della suite *ip* hanno tutti la stessa struttura:

```
ip [options] object command
```

ip: datalink layer

- Vedere le **informazioni** a livello datalink delle interfacce, come ad esempio il MAC address e se risulta "up"

```
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
     qlen 1000 link/ether 00:de:ad:be:ef:ca brd ff:ff:ff:ff:ff:ff
```

ip: datalink layer

- Possiamo cambiare lo stato di una interfaccia e **attivarla** o disattivarla.

```
$ ip link set dev eth0 up
$ ip link set dev eth1 down
```

- Possiamo cambiare il **MAC address**[5] o altri parametri datalink dell'interfaccia:

```
$ ip link set dev eth0 address XX:XX:XX:XX:XX:XX
```

[5] Il MAC address è salvato all'interno di ciascuna scheda di rete per identificarle in modo univoco, tuttavia il sistema operativo è in grado di dire alla scheda di modificarlo (a parte rare schede che non lo permettono).

ip: datalink layer

- Possiamo cambiare lo stato di una interfaccia e **attivarla** o disattivarla.

```
$ ip link set dev eth0 up
$ ip link set dev eth1 down
```

- Possiamo cambiare il **MAC address**[5] o altri parametri datalink dell'interfaccia:

```
$ ip link set dev eth0 address XX:XX:XX:XX:XX:XX
```

[5] Il MAC address è salvato all'interno di ciascuna scheda di rete per identificarle in modo univoco, tuttavia il sistema operativo è in grado di dire alla scheda di modificarlo (a parte rare schede che non lo permettono).

ip: ARP

- Possiamo abilitare o disabiliare ARP se dovesse servire (e se sappiamo quello che stiamo facendo!)

```
$ ip link set dev eth0 arp off
$ ip link set dev eth0 arp on
```

- Possiamo visualizzare la **tabella ARP**:

```
$ ip neigh show
10.99.0.254 dev eth0 lladdr 00:11:22:33:44:55 REACHABLE
```

ip: ARP

- Possiamo abilitare o disabiliare ARP se dovesse servire (e se sappiamo quello che stiamo facendo!)

```
$ ip link set dev eth0 arp off
$ ip link set dev eth0 arp on
```

- Possiamo visualizzare la **tabella ARP**:

```
$ ip neigh show
10.99.0.254 dev eth0 lladdr 00:11:22:33:44:55 REACHABLE
```

ip: Indirizzi IP

- Una volta che abbiamo una interfaccia funzionante, possiamo **assegnare uno o più indirizzi IP**[6]:

```
$ ip address add 192.168.0.2/24 dev eth0
$ ip address add 1234:40ac:1:8d6b::1/64 dev eth0
```

- Il formato utilizzato per indicare gli indirizzi IP è **VLSM** (Variable-length subnet masking). Indichiamo quindi sempre la lunghezza della netmask. L'indirizzamento a classi IP (Classe A,B,C....) è deprecato e non viene più utilizzato dal 1993.

- Rimuovere indirizzi:

```
$ ip address del 192.168.0.2/24 dev eth0
$ ip address flush dev eth0
```

[6] Possiamo assegnare un numero arbitrario di indirizzi IP ad ogni interfaccia di rete senza problemi.

ip: Indirizzi IP

- Una volta che abbiamo una interfaccia funzionante, possiamo **assegnare uno o più indirizzi IP**[6]:

```
$ ip address add 192.168.0.2/24 dev eth0
$ ip address add 1234:40ac:1:8d6b::1/64 dev eth0
```

- Il formato utilizzato per indicare gli indirizzi IP è **VLSM** (Variable-length subnet masking). Indichiamo quindi sempre la lunghezza della netmask. L'indirizzamento a classi IP (Classe A,B,C....) è deprecato e non viene più utilizzato dal 1993.

- Rimuovere indirizzi:

```
$ ip address del 192.168.0.2/24 dev eth0
$ ip address flush dev eth0
```

[6] Possiamo assegnare un numero arbitrario di indirizzi IP ad ogni interfaccia di rete senza problemi.

ip: Indirizzi IP

- Una volta che abbiamo una interfaccia funzionante, possiamo **assegnare uno o più indirizzi IP**[6]:

```
$ ip address add 192.168.0.2/24 dev eth0
$ ip address add 1234:40ac:1:8d6b::1/64 dev eth0
```

- Il formato utilizzato per indicare gli indirizzi IP è **VLSM** (Variable-length subnet masking). Indichiamo quindi sempre la lunghezza della netmask. L'indirizzamento a classi IP (Classe A,B,C....) è deprecato e non viene più utilizzato dal 1993.

- Rimuovere indirizzi:

```
$ ip address del 192.168.0.2/24 dev eth0
$ ip address flush dev eth0
```

[6] Possiamo assegnare un numero arbitrario di indirizzi IP ad ogni interfaccia di rete senza problemi.

ip: Indirizzi IP

- Vedere gli indirizzi assegnati:

```
$ ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    link/ether 00:de:ad:be:ef:ca brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.2/24 brd 192.168.0.255 scope global eth0
    inet6 1234:40ac:1:8d6b::1/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::219:d1ff:aeae:42af/64 scope link
        valid_lft forever preferred_lft forever
```

Ripasso netmask

- Brutalizzando: per distinguere gli indirizzi IP li suddividiamo in vari gruppi chiamati *subnets*.
- Una subnet viene identificata da un IP che gli appartiene e la sua maschera
  - IP: 192.168.0.2
  - Netmask: 255.255.255.0 = 11111111.11111111.11111111.00000000 (= /24)
- Conoscendo la subnet possiamo calcolare l'intervallo di indirizzi validi
- Nel dubbio esiste un comodo tool[7], **ipcalc** che ci permette di fare i conti:

```
$ ipcalc 192.168.0.2/24
```

[7] purtroppo non preinstallato

Ripasso netmask

- Brutalizzando: per distinguere gli indirizzi IP li suddividiamo in vari gruppi chiamati *subnets*.
- Una subnet viene identificata da un IP che gli appartiene e la sua maschera
  - IP: 192.168.0.2
  - Netmask: 255.255.255.0 = 11111111.11111111.11111111.00000000 (= /24)
- Conoscendo la subnet possiamo calcolare l'intervallo di indirizzi validi
- Nel dubbio esiste un comodo tool[7], **ipcalc** che ci permette di fare i conti:

```
$ ipcalc 192.168.0.2/24
```

[7] purtroppo non preinstallato

ip: Route

- Possiamo visualizzare le rotte configurate dal sistema

```
$ ip route show
default via 192.168.0.254 dev eth0
192.168.0.0/24 dev eth0  proto kernel  scope link  src 192.168.0.2  metric 2
```

- E aggiungerne/rimuoverne

```
$ ip route add 1.2.3.0/24 via 192.168.0.254 dev eth0
$ ip route del 1.2.3.4 via 192.168.0.254 dev eth0
```

- Per poter "collegarci" ad Internet dobbiamo avere una route di default, delegata ad un gateway presente sulla nostra rete locale.

```
$ ip route add default via 192.168.0.254 dev eth0
```

ip: Route

- Possiamo visualizzare le rotte configurate dal sistema

```
$ ip route show
default via 192.168.0.254 dev eth0
192.168.0.0/24 dev eth0  proto kernel  scope link  src 192.168.0.2  metric 2
```

- E aggiungerne/rimuoverne

```
$ ip route add 1.2.3.0/24 via 192.168.0.254 dev eth0
$ ip route del 1.2.3.4 via 192.168.0.254 dev eth0
```

- Per poter "collegarci" ad Internet dobbiamo avere una route di default, delegata ad un gateway presente sulla nostra rete locale.

```
$ ip route add default via 192.168.0.254 dev eth0
```

ip: Route

- Possiamo visualizzare le rotte configurate dal sistema

```
$ ip route show
default via 192.168.0.254 dev eth0
192.168.0.0/24 dev eth0  proto kernel  scope link  src 192.168.0.2  metric 2
```

- E aggiungerne/rimuoverne

```
$ ip route add 1.2.3.0/24 via 192.168.0.254 dev eth0
$ ip route del 1.2.3.4 via 192.168.0.254 dev eth0
```

- Per poter "collegarci" ad Internet dobbiamo avere una route di default, delegata ad un gateway presente sulla nostra rete locale.

```
$ ip route add default via 192.168.0.254 dev eth0
```

dhclient

- Nel caso in cui non vogliamo assegnare un indirizzo statico, possiamo utilizzare un client DHCP come ad esempio ***dhclient***, per ottenere un indirizzo dal server DHCP sulla rete locale (solitamente installato sul gateway di default).

```
$ dhclient eth0
Internet Systems Consortium DHCP Client V3.1.1
Copyright 2004-2008 Internet Systems Consortium.
All rights reserved. For info, please visit http://www.isc.org/sw/dhcp/
Listening on LPF/eth0/6e:5f:98:37:0c:07
Sending on   LPF/eth0/6e:5f:98:37:0c:07
Sending on   Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 4
DHCPOFFER from 10.5.5.1
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 10.5.5.1
bound to 10.5.5.26 -- renewal in 297 seconds.
```

dhclient

- Nel caso in cui non vogliamo assegnare un indirizzo statico, possiamo utilizzare un client DHCP come ad esempio ***dhclient***, per ottenere un indirizzo dal server DHCP sulla rete locale (solitamente installato sul gateway di default).

```
$ dhclient eth0
Internet Systems Consortium DHCP Client V3.1.1
Copyright 2004-2008 Internet Systems Consortium.
All rights reserved. For info, please visit http://www.isc.org/sw/dhcp/
Listening on LPF/eth0/6e:5f:98:37:0c:07
Sending on   LPF/eth0/6e:5f:98:37:0c:07
Sending on   Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 4
DHCPOFFER from 10.5.5.1
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 10.5.5.1
bound to 10.5.5.26 -- renewal in 297 seconds.
```

Configurazione su Debian/Ubuntu

- Su Debian e Ubuntu possiamo impostare le interfacce di rete in modo permanente tramite il file */etc/network/interfaces*, che verrà letto e utilizzato all'avvio del sistema per configurare le interfacce.

```
# The loopback network interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
address 192.168.0.2
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255
gateway 192.168.0.254
```

- Se utilizziamo DHCP è più semplicemente:

```
# The loopback network interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp
```

Configurazione su Debian/Ubuntu

- Su Debian e Ubuntu possiamo impostare le interfacce di rete in modo permanente tramite il file */etc/network/interfaces*, che verrà letto e utilizzato all'avvio del sistema per configurare le interfacce.

```
# The loopback network interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
address 192.168.0.2
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255
gateway 192.168.0.254
```

- Se utilizziamo DHCP è più semplicemente:

```
# The loopback network interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp
```

Configurazione su Debian/Ubuntu

- Su Debian e Ubuntu possiamo impostare le interfacce di rete in modo permanente tramite il file */etc/network/interfaces*, che verrà letto e utilizzato all'avvio del sistema per configurare le interfacce.

```
# The loopback network interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
address 192.168.0.2
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255
gateway 192.168.0.254
```

- Se utilizziamo DHCP è più semplicemente:

```
# The loopback network interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp
```

DNS

- Nel caso in cui **configuriamo la rete manualmente**, senza utilizzare un client DHCP; al fine di poter risolvere i nomi DNS dobbiamo **impostare un server DNS**

- Su Linux, tutte le applicazioni che devono risolvere nomi consultano il file /etc/nsswitch.conf, che a sua volta indica di consultare:
  - */etc/hosts* : Dove sono presenti eventuali assegnamenti statici di nomi che possiamo configurare manualmente
  - */etc/resolv.conf* : Dove sono presenti gli indirizzi IP dei server DNS da consultare, nel formato che segue:

```
nameserver 208.67.222.222
nameserver 8.8.8.8
```

DNS

- Nel caso in cui **configuriamo la rete manualmente**, senza utilizzare un client DHCP; al fine di poter risolvere i nomi DNS dobbiamo **impostare un server DNS**

- Su Linux, tutte le applicazioni che devono risolvere nomi consultano il file /etc/nsswitch.conf, che a sua volta indica di consultare:
  - */etc/hosts* : Dove sono presenti eventuali assegnamenti statici di nomi che possiamo configurare manualmente
  - */etc/resolv.conf* : Dove sono presenti gli indirizzi IP dei server DNS da consultare, nel formato che segue:

```
nameserver 208.67.222.222
nameserver 8.8.8.8
```

DNS

- Nel caso in cui **configuriamo la rete manualmente**, senza utilizzare un client DHCP; al fine di poter risolvere i nomi DNS dobbiamo **impostare un server DNS**

- Su Linux, tutte le applicazioni che devono risolvere nomi consultano il file /etc/nsswitch.conf, che a sua volta indica di consultare:
  - */etc/hosts* : Dove sono presenti eventuali assegnamenti statici di nomi che possiamo configurare manualmente
  - */etc/resolv.conf* : Dove sono presenti gli indirizzi IP dei server DNS da consultare, nel formato che segue:

```
nameserver 208.67.222.222
nameserver 8.8.8.8
```

ping

- Per diagnostica di rete si utilizza solitamente il comando ping

```
$ ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_req=1 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_req=2 ttl=64 time=1.22 ms
64 bytes from 192.168.0.1: icmp_req=3 ttl=64 time=1.13 ms
64 bytes from 192.168.0.1: icmp_req=4 ttl=64 time=1.14 ms
64 bytes from 192.168.0.1: icmp_req=5 ttl=64 time=1.49 ms
^C
--- 192.168.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 1.047/1.211/1.499/0.158 ms
```

- Buona pratica per il debugging di rete è di fare un ping a:
  - Localhost
  - Il gateway di default della rete locale o un'altra macchina della rete locale
  - Un host esterno alla rete locale

ping

- Per diagnostica di rete si utilizza solitamente il comando ping

```
$ ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_req=1 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_req=2 ttl=64 time=1.22 ms
64 bytes from 192.168.0.1: icmp_req=3 ttl=64 time=1.13 ms
64 bytes from 192.168.0.1: icmp_req=4 ttl=64 time=1.14 ms
64 bytes from 192.168.0.1: icmp_req=5 ttl=64 time=1.49 ms
^C
--- 192.168.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 1.047/1.211/1.499/0.158 ms
```

- Buona pratica per il debugging di rete è di fare un ping a:
  - Localhost
  - Il gateway di default della rete locale o un'altra macchina della rete locale
  - Un host esterno alla rete locale

ping

- Per diagnostica di rete si utilizza solitamente il comando ping

```
$ ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_req=1 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_req=2 ttl=64 time=1.22 ms
64 bytes from 192.168.0.1: icmp_req=3 ttl=64 time=1.13 ms
64 bytes from 192.168.0.1: icmp_req=4 ttl=64 time=1.14 ms
64 bytes from 192.168.0.1: icmp_req=5 ttl=64 time=1.49 ms
^C
--- 192.168.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 1.047/1.211/1.499/0.158 ms
```

- Buona pratica per il debugging di rete è di fare un ping a:
  - Localhost
  - Il gateway di default della rete locale o un'altra macchina della rete locale
  - Un host esterno alla rete locale

ping

- Per diagnostica di rete si utilizza solitamente il comando ping

```
$ ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_req=1 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_req=2 ttl=64 time=1.22 ms
64 bytes from 192.168.0.1: icmp_req=3 ttl=64 time=1.13 ms
64 bytes from 192.168.0.1: icmp_req=4 ttl=64 time=1.14 ms
64 bytes from 192.168.0.1: icmp_req=5 ttl=64 time=1.49 ms
^C
--- 192.168.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 1.047/1.211/1.499/0.158 ms
```

- Buona pratica per il debugging di rete è di fare un ping a:
  - Localhost
  - Il gateway di default della rete locale o un'altra macchina della rete locale
  - Un host esterno alla rete locale

ping

- Per diagnostica di rete si utilizza solitamente il comando ping

```
$ ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_req=1 ttl=64 time=1.04 ms
64 bytes from 192.168.0.1: icmp_req=2 ttl=64 time=1.22 ms
64 bytes from 192.168.0.1: icmp_req=3 ttl=64 time=1.13 ms
64 bytes from 192.168.0.1: icmp_req=4 ttl=64 time=1.14 ms
64 bytes from 192.168.0.1: icmp_req=5 ttl=64 time=1.49 ms
^C
--- 192.168.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 1.047/1.211/1.499/0.158 ms
```

- Buona pratica per il debugging di rete è di fare un ping a:
  - Localhost
  - Il gateway di default della rete locale o un'altra macchina della rete locale
  - Un host esterno alla rete locale

host

- Possiamo controllare la risoluzione dei nomi DNS utilizzando il comando “host”

```
$ host poul.org
poul.org has address 91.121.182.81
poul.org has IPv6 address 2001:41d0:1:f751::1
poul.org mail is handled by 1 poul.org.

$ host -t MX poul.org
poul.org mail is handled by 1 poul.org.

$ host 91.121.182.81
81.182.121.91.in-addr.arpa domain name pointer poul.org.
```

host

- Possiamo controllare la risoluzione dei nomi DNS utilizzando il comando “host”

```
$ host poul.org
poul.org has address 91.121.182.81
poul.org has IPv6 address 2001:41d0:1:f751::1
poul.org mail is handled by 1 poul.org.

$ host -t MX poul.org
poul.org mail is handled by 1 poul.org.

$ host 91.121.182.81
81.182.121.91.in-addr.arpa domain name pointer poul.org.
```

host

- Possiamo controllare la risoluzione dei nomi DNS utilizzando il comando “host”

```
$ host poul.org
poul.org has address 91.121.182.81
poul.org has IPv6 address 2001:41d0:1:f751::1
poul.org mail is handled by 1 poul.org.

$ host -t MX poul.org
poul.org mail is handled by 1 poul.org.

$ host 91.121.182.81
81.182.121.91.in-addr.arpa domain name pointer poul.org.
```

host

- Possiamo controllare la risoluzione dei nomi DNS utilizzando il comando “host”

```
$ host poul.org
poul.org has address 91.121.182.81
poul.org has IPv6 address 2001:41d0:1:f751::1
poul.org mail is handled by 1 poul.org.

$ host -t MX poul.org
poul.org mail is handled by 1 poul.org.

$ host 91.121.182.81
81.182.121.91.in-addr.arpa domain name pointer poul.org.
```

NetCat

- Uno dei tool principali per fare scripting di rete e debugging è netcat.
- Netcat è molto semplicemente un client/server tcp/udp da linea di comando. Una sorta di "telnet client" avanzato.

```
$ nc [hostname] [port]
```

```
$ nc -l -p [port]
```
[8]

- *-l -p [porta]* per restare in ascolto per connessioni sulla porta data
- *-u* per indicare l'utilizzo di UDP invece di TCP
- *-v* per avere informazioni aggiuntive durante la connessione

[8] In altre versioni di netcat non è necessaria l'opzione -p

NetCat

- Uno dei tool principali per fare scripting di rete e debugging è netcat.
- Netcat è molto semplicemente un client/server tcp/udp da linea di comando. Una sorta di "telnet client" avanzato.

```
$ nc [hostname] [port]
```

```
$ nc -l -p [port]
```
[8]

- *-l -p [porta]* per restare in ascolto per connessioni sulla porta data
- *-u* per indicare l'utilizzo di UDP invece di TCP
- *-v* per avere informazioni aggiuntive durante la connessione

[8] In altre versioni di netcat non è necessaria l'opzione -p

NetCat

- Uno dei tool principali per fare scripting di rete e debugging è netcat.
- Netcat è molto semplicemente un client/server tcp/udp da linea di comando. Una sorta di "telnet client" avanzato.

```
$ nc [hostname] [port]
```

```
$ nc -l -p [port]
```
[8]

- *-l -p [porta]* per restare in ascolto per connessioni sulla porta data
- *-u* per indicare l'utilizzo di UDP invece di TCP
- *-v* per avere informazioni aggiuntive durante la connessione

[8] In altre versioni di netcat non è necessaria l'opzione -p

NetCat

- Uno dei tool principali per fare scripting di rete e debugging è netcat.
- Netcat è molto semplicemente un client/server tcp/udp da linea di comando. Una sorta di "telnet client" avanzato.

```
$ nc [hostname] [port]
```

```
$ nc -l -p [port]
```
[8]

- *-l -p [porta]* per restare in ascolto per connessioni sulla porta data
- *-u* per indicare l'utilizzo di UDP invece di TCP
- *-v* per avere informazioni aggiuntive durante la connessione

[8] In altre versioni di netcat non è necessaria l'opzione -p

NetCat

- Uno dei tool principali per fare scripting di rete e debugging è netcat.
- Netcat è molto semplicemente un client/server tcp/udp da linea di comando. Una sorta di "telnet client" avanzato.

```
$ nc [hostname] [port]
```

```
$ nc -l -p [port]
```
[8]

- *-l -p [porta]* per restare in ascolto per connessioni sulla porta data
- *-u* per indicare l'utilizzo di UDP invece di TCP
- *-v* per avere informazioni aggiuntive durante la connessione

[8] In altre versioni di netcat non è necessaria l'opzione -p

NetCat

- Uno dei tool principali per fare scripting di rete e debugging è netcat.
- Netcat è molto semplicemente un client/server tcp/udp da linea di comando. Una sorta di "telnet client" avanzato.

```
$ nc [hostname] [port]
```

```
$ nc -l -p [port]
```
[8]

- *-l -p [porta]* per restare in ascolto per connessioni sulla porta data
- *-u* per indicare l'utilizzo di UDP invece di TCP
- *-v* per avere informazioni aggiuntive durante la connessione

[8] In altre versioni di netcat non è necessaria l'opzione -p

NetCat

- Uno dei tool principali per fare scripting di rete e debugging è netcat.
- Netcat è molto semplicemente un client/server tcp/udp da linea di comando. Una sorta di "telnet client" avanzato.

```
$ nc [hostname] [port]
```

```
$ nc -l -p [port]
```
[8]

- *-l -p [porta]* per restare in ascolto per connessioni sulla porta data
- *-u* per indicare l'utilizzo di UDP invece di TCP
- *-v* per avere informazioni aggiuntive durante la connessione

[8] In altre versioni di netcat non è necessaria l'opzione -p

ss: connessioni del sistema

- Può tornare utile avere una lista di tutte le connessioni attive e dei programmi associati a ciascuna connessione.
- Una "*connessione*" è una quintupla composta tipicamente da: *(ip sorgente, ip destinazione, porta sorgente, porta destinazione, protocollo).*
- Il comando "*ss*" ci permette di ottenere un po' di informazioni. In realtà possiamo anche utilizzare alcune opzioni di *lsof* come abbiamo già visto in precedenti lezioni. (Nota: dobbiamo utilizzare "ss" da **root** per poter vedere i processi associati alle connessioni)

ss: connessioni del sistema

- Può tornare utile avere una lista di tutte le connessioni attive e dei programmi associati a ciascuna connessione.
- Una "*connessione*" è una quintupla composta tipicamente da: *(ip sorgente, ip destinazione, porta sorgente, porta destinazione, protocollo).*
- Il comando "*ss*" ci permette di ottenere un po' di informazioni. In realtà possiamo anche utilizzare alcune opzioni di *lsof* come abbiamo già visto in precedenti lezioni. (Nota: dobbiamo utilizzare "ss" da **root** per poter vedere i processi associati alle connessioni)

ss: connessioni del sistema

- Può tornare utile avere una lista di tutte le connessioni attive e dei programmi associati a ciascuna connessione.
- Una "*connessione*" è una quintupla composta tipicamente da: *(ip sorgente, ip destinazione, porta sorgente, porta destinazione, protocollo).*
- Il comando "*ss*" ci permette di ottenere un po' di informazioni. In realtà possiamo anche utilizzare alcune opzioni di *lsof* come abbiamo già visto in precedenti lezioni. (Nota: dobbiamo utilizzare "ss" da **root** per poter vedere i processi associati alle connessioni)

ss: connessioni del sistema

```
$ ss -ap
State       Recv-Q Send-Q   Local Address:Port     Peer Address:Port
LISTEN      0      50           127.0.0.1:mysql             *:*       users:(("mysqld",6218,11))
LISTEN      0      128                :::http            :::*       users:(("nginx",31628,19),("ngi
LISTEN      0      128                 *:http             *:*       users:(("nginx",31628,18),("ngi
LISTEN      0      32                  *:ftp              *:*       users:(("vsftpd",4941,3))
LISTEN      0      128                :::ssh             :::*       users:(("sshd",5990,4))
LISTEN      0      128                 *:ssh              *:*       users:(("sshd",5990,3))
ESTAB       0      0            10.0.0.1:56345      10.0.0.2:37597   users:(("python",29970,14))
ESTAB       0      0         91.121.182.81:ssh       1.2.3.4:1933    users:(("sshd",19373,3),("sshd"
```

Netstat

In alternativa ad “ss” c’è il più datato “netstat”

```
$ netstat -untap
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      6218/mysqld
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      19952/nginx
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN      4941/vsftpd
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      5990/sshd
tcp        0      0 10.0.0.1:56345          0.0.0.0:*               LISTEN      29970/python
tcp        0     48 91.121.182.81:22        1.2.3.4:1933            ESTABLISHED 19371/sshd: otacon
tcp6       0      0 :::80                   :::*                    LISTEN      19952/nginx
tcp6       0      0 :::22                   :::*                    LISTEN      5990/sshd
tcp6       0      0 :::443                  :::*                    LISTEN      19952/nginx
udp        0      0 0.0.0.0:5060            0.0.0.0:*                           8460/asterisk
```

Banda per connessione

Altri tool utili per vedere la quantità di banda (byte/s) utilizzati da ogni connessione:

- *iftop -i eth0*
- *nethogs eth0*

Misurazione banda

- Un comodo tool per verificare la velocità massima raggiungibile da una connessione TCP (e quindi tipicamente dalla linea sottostante) è ***iperf***. Non è preinstallato.
  - Sul client eseguiamo:

    ```
    $ iperf -c [iperf server hostname] -p [porta]
    ```

  - Sul server eseguiamo:

    ```
    $ iperf -s -p [porta]
    ```

- Per misure accurate per i valori di latenza e jitter è consigliabile usare la modalità UDP ("*-u*").

tcpdump

- Per vedere il traffico in transito su una interfaccia possiamo utilizzare tcpdump
- *tcpdump -i [interfaccia] -n*
  - Ci permette di vedere il traffico sulla interfaccia specificata, senza risolvere i *reverse-DNS* degli indirizzi che vediamo (spreca tempo e banda).
- *tcpdump -i [interfaccia] -n -w file.pcap*
  - Salva un dump di tutto il traffico nel file specificato "*file.pcap*", per una analisi successiva

tcpdump

- Per vedere il traffico in transito su una interfaccia possiamo utilizzare tcpdump
- *tcpdump -i [interfaccia] -n*
  - Ci permette di vedere il traffico sulla interfaccia specificata, senza risolvere i *reverse-DNS* degli indirizzi che vediamo (spreca tempo e banda).
- *tcpdump -i [interfaccia] -n -w file.pcap*
  - Salva un dump di tutto il traffico nel file specificato "*file.pcap*", per una analisi successiva

tcpdump

- Per vedere il traffico in transito su una interfaccia possiamo utilizzare tcpdump
- *tcpdump -i [interfaccia] -n*
  - Ci permette di vedere il traffico sulla interfaccia specificata, senza risolvere i *reverse-DNS* degli indirizzi che vediamo (spreca tempo e banda).
- *tcpdump -i [interfaccia] -n -w file.pcap*
  - Salva un dump di tutto il traffico nel file specificato "*file.pcap*", per una analisi successiva

tcpdump

```
$ sudo tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
15:39:04.224983 IP 192.168.0.81.57828 > 157.56.127.36.443:
     Flags [P.], seq 1901073964:1901074005, ack 2664128571, win 41856,
     options [nop,nop,TS val 15115199 ecr 79213980], length 41
15:39:04.265743 IP 157.56.127.36.443 > 192.168.0.81.57828:
     Flags [P.], seq 1:89, ack 41, win 64935,
     options [nop,nop,TS val 79235515 ecr 15115199], length 88
15:39:04.265785 IP 192.168.0.81.57828 > 117.56.117.36.443:
     Flags [.], ack 89, win 41856, options [nop,nop,TS val 15115209 ecr 792355
15:39:04.656254 IP 82.140.3.129.7000 > 102.168.0.81.37336:
     Flags [P.], seq 3133130585:3133130723, ack 3995030141, win 2312,
     options [nop,nop,TS val 3490043 ecr 15114909], length 138
15:39:04.656321 IP 102.168.0.81.37336 > 83.240.3.129.7000:
     Flags [.], ack 138, win 540, options [nop,nop,TS val 15115306 ecr 3490043
^C
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

tcpdump e sintassi BPF

- Possiamo utilizzare la sintassi BPF all'interno di tcpdump per specificare quali pacchetti vogliamo vedere del traffico
- *tcpdump -i eth0 -n "icmp"*
  - Vediamo solo i pacchetti icmp (ping/traceroute/...)
- *tcpdump -i eth0 -n "arp"*
  - Vediamo solo richieste e risposte ARP
- *tcpdump -i eth0 -n "tcp and src or dst 192.168.0.1"*
  - Vediamo tutti i pacchetti TCP con sorgente o destinazione 192.168.0.1
- *tcpdump -i eth0 -n "tcp port 22"*
  - Vediamo tutti i pacchetti TCP con porta sorgente o destinazione 22.

tcpdump e sintassi BPF

- Possiamo utilizzare la sintassi BPF all'interno di tcpdump per specificare quali pacchetti vogliamo vedere del traffico
- *tcpdump -i eth0 -n "icmp"*
  - Vediamo solo i pacchetti icmp (ping/traceroute/...)
- *tcpdump -i eth0 -n "arp"*
  - Vediamo solo richieste e risposte ARP
- *tcpdump -i eth0 -n "tcp and src or dst 192.168.0.1"*
  - Vediamo tutti i pacchetti TCP con sorgente o destinazione 192.168.0.1
- *tcpdump -i eth0 -n "tcp port 22"*
  - Vediamo tutti i pacchetti TCP con porta sorgente o destinazione 22.

tcpdump e sintassi BPF

- Possiamo utilizzare la sintassi BPF all'interno di tcpdump per specificare quali pacchetti vogliamo vedere del traffico
- *tcpdump -i eth0 -n "icmp"*
  - Vediamo solo i pacchetti icmp (ping/traceroute/...)
- *tcpdump -i eth0 -n "arp"*
  - Vediamo solo richieste e risposte ARP
- *tcpdump -i eth0 -n "tcp and src or dst 192.168.0.1"*
  - Vediamo tutti i pacchetti TCP con sorgente o destinazione 192.168.0.1
- *tcpdump -i eth0 -n "tcp port 22"*
  - Vediamo tutti i pacchetti TCP con porta sorgente o destinazione 22.

tcpdump e sintassi BPF

- Possiamo utilizzare la sintassi BPF all'interno di tcpdump per specificare quali pacchetti vogliamo vedere del traffico
- *tcpdump -i eth0 -n "icmp"*
  - Vediamo solo i pacchetti icmp (ping/traceroute/...)
- *tcpdump -i eth0 -n "arp"*
  - Vediamo solo richieste e risposte ARP
- *tcpdump -i eth0 -n "tcp and src or dst 192.168.0.1"*
  - Vediamo tutti i pacchetti TCP con sorgente o destinazione 192.168.0.1
- *tcpdump -i eth0 -n "tcp port 22"*
  - Vediamo tutti i pacchetti TCP con porta sorgente o destinazione 22.

tcpdump e sintassi BPF

- Possiamo utilizzare la sintassi BPF all'interno di tcpdump per specificare quali pacchetti vogliamo vedere del traffico
- *tcpdump -i eth0 -n "icmp"*
  - Vediamo solo i pacchetti icmp (ping/traceroute/...)
- *tcpdump -i eth0 -n "arp"*
  - Vediamo solo richieste e risposte ARP
- *tcpdump -i eth0 -n "tcp and src or dst 192.168.0.1"*
  - Vediamo tutti i pacchetti TCP con sorgente o destinazione 192.168.0.1
- *tcpdump -i eth0 -n "tcp port 22"*
  - Vediamo tutti i pacchetti TCP con porta sorgente o destinazione 22.

SSH

- *ssh [opzioni] username@hostname*
  - *-i /path/chiave* : Per l'autenticazione tramite chiave pubblica[9]
  - *-o PubkeyAuthentication=no* : Per forzare il login mediante password
  - *-l user* : Modo alternativo per indicare l'utente da utilizzare per il login
  - *-p [porta]* : Indica una porta (diversa dalla 22) da utilizzare per la connessione.

[9] Possiamo generare chiavi utilizzando *ssh-keygen*

SSH

- *ssh [opzioni] username@hostname*
  - *-i /path/chiave* : Per l'autenticazione tramite chiave pubblica[9]
  - *-o PubkeyAuthentication=no* : Per forzare il login mediante password
  - *-l user* : Modo alternativo per indicare l'utente da utilizzare per il login
  - *-p [porta]* : Indica una porta (diversa dalla 22) da utilizzare per la connessione.

[9] Possiamo generare chiavi utilizzando *ssh-keygen*

SSH

- *ssh [opzioni] username@hostname*
  - *-i /path/chiave* : Per l'autenticazione tramite chiave pubblica[9]
  - *-o PubkeyAuthentication=no* : Per forzare il login mediante password
  - *-l user* : Modo alternativo per indicare l'utente da utilizzare per il login
  - *-p [porta]* : Indica una porta (diversa dalla 22) da utilizzare per la connessione.

[9] Possiamo generare chiavi utilizzando *ssh-keygen*

SSH

- *ssh [opzioni] username@hostname*
  - *-i /path/chiave* : Per l'autenticazione tramite chiave pubblica[9]
  - *-o PubkeyAuthentication=no* : Per forzare il login mediante password
  - *-l user* : Modo alternativo per indicare l'utente da utilizzare per il login
  - *-p [porta]* : Indica una porta (diversa dalla 22) da utilizzare per la connessione.

[9] Possiamo generare chiavi utilizzando *ssh-keygen*

Sezione 4

Routing

- Prima di parlare di firewalling vediamo il routing.
- Ogni host GNU/Linux è anche un router.
- Per abilitare il "forwarding" di pacchetti ip dobbiamo abilitarlo con

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

- In questo modo altri host potranno utilizzare noi come router (o gateway di default e noi provvedremo ad inoltrare il traffico in base alle nostre regole di routing.

Routing

- Prima di parlare di firewalling vediamo il routing.
- Ogni host GNU/Linux è anche un router.
- Per abilitare il "forwarding" di pacchetti ip dobbiamo abilitarlo con

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

- In questo modo altri host potranno utilizzare noi come router (o gateway di default e noi provvedremo ad inoltrare il traffico in base alle nostre regole di routing.

Routing

- Prima di parlare di firewalling vediamo il routing.
- Ogni host GNU/Linux è anche un router.
- Per abilitare il "forwarding" di pacchetti ip dobbiamo abilitarlo con

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

- In questo modo altri host potranno utilizzare noi come router (o gateway di default e noi provvedremo ad inoltrare il traffico in base alle nostre regole di routing.

Routing

- Prima di parlare di firewalling vediamo il routing.
- Ogni host GNU/Linux è anche un router.
- Per abilitare il "forwarding" di pacchetti ip dobbiamo abilitarlo con

  ```
  echo "1" > /proc/sys/net/ipv4/ip_forward
  ```

- In questo modo altri host potranno utilizzare noi come router (o gateway di default e noi provvedremo ad inoltrare il traffico in base alle nostre regole di routing.

Firewalling

- Un firewall è un "programma" che si occupa di decidere se accettare o meno i pacchetti in transito su un host.
- Un firewall può essere utile:
  - Su un router di frontiera per discriminare l'accesso ai servizi di una rete
  - Su un server, per esporre solo determinati servizi sul singolo server
- "Blocchiamo" certi tipi di pacchetti in transito, in base a certe regole.
- Il firewall deve essere l'unico punto di contatto della rete con l'esterno[10]

[10] Per evitare il problema di generare *Single Point Of Failure* è eventualmente possibile utilizzare più firewall sincronizzati costantemente tra di loro: `http://conntrack-tools.netfilter.org/`

Firewalling

- Un firewall è un "programma" che si occupa di decidere se accettare o meno i pacchetti in transito su un host.
- Un firewall può essere utile:
  - Su un router di frontiera per discriminare l'accesso ai servizi di una rete
  - Su un server, per esporre solo determinati servizi sul singolo server
- "Blocchiamo" certi tipi di pacchetti in transito, in base a certe regole.
- Il firewall deve essere l'unico punto di contatto della rete con l'esterno[10]

[10] Per evitare il problema di generare *Single Point Of Failure* è eventualmente possibile utilizzare più firewall sincronizzati costantemente tra di loro: `http://conntrack-tools.netfilter.org/`

Firewalling

- Un firewall è un "programma" che si occupa di decidere se accettare o meno i pacchetti in transito su un host.
- Un firewall può essere utile:
  - Su un router di frontiera per discriminare l'accesso ai servizi di una rete
  - Su un server, per esporre solo determinati servizi sul singolo server
- "Blocchiamo" certi tipi di pacchetti in transito, in base a certe regole.
- Il firewall deve essere l'unico punto di contatto della rete con l'esterno[10]

[10] Per evitare il problema di generare *Single Point Of Failure* è eventualmente possibile utilizzare più firewall sincronizzati costantemente tra di loro: `http://conntrack-tools.netfilter.org/`

Firewalling

- Un firewall è un "programma" che si occupa di decidere se accettare o meno i pacchetti in transito su un host.
- Un firewall può essere utile:
  - Su un router di frontiera per discriminare l'accesso ai servizi di una rete
  - Su un server, per esporre solo determinati servizi sul singolo server
- "Blocchiamo" certi tipi di pacchetti in transito, in base a certe regole.
- Il firewall deve essere l'unico punto di contatto della rete con l'esterno[10]

[10] Per evitare il problema di generare *Single Point Of Failure* è eventualmente possibile utilizzare più firewall sincronizzati costantemente tra di loro: `http://conntrack-tools.netfilter.org/`

Firewalling

- Un firewall è un "programma" che si occupa di decidere se accettare o meno i pacchetti in transito su un host.
- Un firewall può essere utile:
  - Su un router di frontiera per discriminare l'accesso ai servizi di una rete
  - Su un server, per esporre solo determinati servizi sul singolo server
- "Blocchiamo" certi tipi di pacchetti in transito, in base a certe regole.
- Il firewall deve essere l'unico punto di contatto della rete con l'esterno[10]

[10] Per evitare il problema di generare *Single Point Of Failure* è eventualmente possibile utilizzare più firewall sincronizzati costantemente tra di loro: `http://conntrack-tools.netfilter.org/`

Firewalling

- All'interno del kernel Linux è presente una parte del kernel dedicata esclusivamente a gestire le operazioni di firewalling, chiamata ***netfilter***.
- In userspace possiamo utilizzare **iptables**/**ip6tables**/**ebtables** come strumenti per comunicare al kernel le regole da applicare[11] a livello IP(v4 e v6) e (se necessario) Ethernet/datalink.
- Netfilter è un firewall di tipo ***stateful***, cioè permette di tenere traccia delle connessioni che lo attraversano e del loro stato.
- Netfilter si occupa anche di implementare le operazioni di NAT e port forwarding spesso utilizzate.

[11] A breve si migrerà ad un nuovo unico tool, chiamato **nftables**, che tuttavia non modifica la struttura delle tabelle e hooks che vedremo tra un attimo.

Firewalling

- All'interno del kernel Linux è presente una parte del kernel dedicata esclusivamente a gestire le operazioni di firewalling, chiamata ***netfilter***.
- In userspace possiamo utilizzare **iptables**/**ip6tables**/**ebtables** come strumenti per comunicare al kernel le regole da applicare[11] a livello IP(v4 e v6) e (se necessario) Ethernet/datalink.
- Netfilter è un firewall di tipo ***stateful***, cioè permette di tenere traccia delle connessioni che lo attraversano e del loro stato.
- Netfilter si occupa anche di implementare le operazioni di NAT e port forwarding spesso utilizzate.

[11] A breve si migrerà ad un nuovo unico tool, chiamato **nftables**, che tuttavia non modifica la struttura delle tabelle e hooks che vedremo tra un attimo.

Firewalling

- All'interno del kernel Linux è presente una parte del kernel dedicata esclusivamente a gestire le operazioni di firewalling, chiamata ***netfilter***.
- In userspace possiamo utilizzare **iptables**/**ip6tables**/**ebtables** come strumenti per comunicare al kernel le regole da applicare[11] a livello IP(v4 e v6) e (se necessario) Ethernet/datalink.
- Netfilter è un firewall di tipo ***stateful***, cioè permette di tenere traccia delle connessioni che lo attraversano e del loro stato.
- Netfilter si occupa anche di implementare le operazioni di NAT e port forwarding spesso utilizzate.

[11] A breve si migrerà ad un nuovo unico tool, chiamato **nftables**, che tuttavia non modifica la struttura delle tabelle e hooks che vedremo tra un attimo.

Firewalling

- All'interno del kernel Linux è presente una parte del kernel dedicata esclusivamente a gestire le operazioni di firewalling, chiamata ***netfilter***.
- In userspace possiamo utilizzare **iptables**/**ip6tables**/**ebtables** come strumenti per comunicare al kernel le regole da applicare[11] a livello IP(v4 e v6) e (se necessario) Ethernet/datalink.
- Netfilter è un firewall di tipo ***stateful***, cioè permette di tenere traccia delle connessioni che lo attraversano e del loro stato.
- Netfilter si occupa anche di implementare le operazioni di NAT e port forwarding spesso utilizzate.

[11] A breve si migrerà ad un nuovo unico tool, chiamato **nftables**, che tuttavia non modifica la struttura delle tabelle e hooks che vedremo tra un attimo.

Netfilter

- L'infrastruttura di Netfilter è basata su 5 "hooks" presenti nel percorso dei pacchetti. Sono 5 "punti" dove i pacchetti devono per forza passare.
- In ognuno di questi "punti" è possibile:
  - Permettere il passaggio del pacchetto (ACCEPT)
  - Scartare il pacchetto e interromperlo (DROP)
  - Redirigere il pacchetto
  - Modificare alcuni campi del pacchetto (*mangle*)

Netfilter

- L'infrastruttura di Netfilter è basata su 5 "hooks" presenti nel percorso dei pacchetti. Sono 5 "punti" dove i pacchetti devono per forza passare.
- In ognuno di questi "punti" è possibile:
  - Permettere il passaggio del pacchetto (ACCEPT)
  - Scartare il pacchetto e interromperlo (DROP)
  - Redirigere il pacchetto
  - Modificare alcuni campi del pacchetto (*mangle*)

Netfilter

- L'infrastruttura di Netfilter è basata su 5 "hooks" presenti nel percorso dei pacchetti. Sono 5 "punti" dove i pacchetti devono per forza passare.
- In ognuno di questi "punti" è possibile:
  - Permettere il passaggio del pacchetto (ACCEPT)
  - Scartare il pacchetto e interromperlo (DROP)
  - Redirigere il pacchetto
  - Modificare alcuni campi del pacchetto (*mangle*)

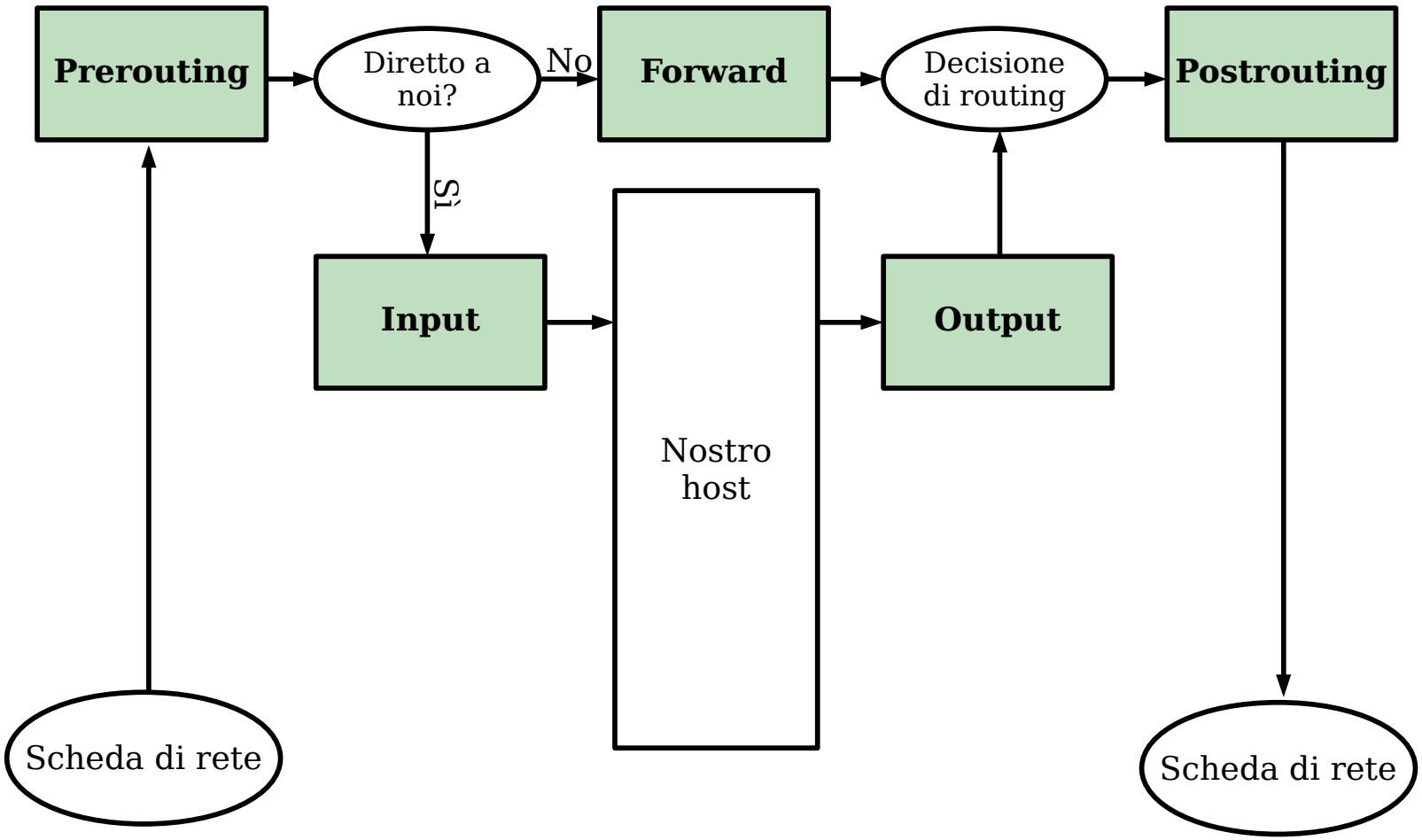
Netfilter

- L'infrastruttura di Netfilter è basata su 5 "hooks" presenti nel percorso dei pacchetti. Sono 5 "punti" dove i pacchetti devono per forza passare.
- In ognuno di questi "punti" è possibile:
  - Permettere il passaggio del pacchetto (ACCEPT)
  - Scartare il pacchetto e interromperlo (DROP)
  - Redirigere il pacchetto
  - Modificare alcuni campi del pacchetto (*mangle*)

Netfilter

- L'infrastruttura di Netfilter è basata su 5 "hooks" presenti nel percorso dei pacchetti. Sono 5 "punti" dove i pacchetti devono per forza passare.
- In ognuno di questi "punti" è possibile:
  - Permettere il passaggio del pacchetto (ACCEPT)
  - Scartare il pacchetto e interromperlo (DROP)
  - Redirigere il pacchetto
  - Modificare alcuni campi del pacchetto (*mangle*)

Netfilter


Prerouting
Diretto a noi?
No
Forward
Decisione di routing
Postrouting
Si`
Input
Nostro host
Output
Scheda di rete
Scheda di rete

Hooks

- **PREROUTING**: pacchetti in ingresso
- **INPUT**: pacchetti destinati all'host stesso
- **FORWARD**: pacchetti non destinati all'host stesso, che dobbiamo inoltrare
- **OUTPUT**: pacchetti in uscita dall'host stesso
- **POSTROUTING**: pacchetti in uscita

Hooks

- **PREROUTING**: pacchetti in ingresso
- **INPUT**: pacchetti destinati all'host stesso
- **FORWARD**: pacchetti non destinati all'host stesso, che dobbiamo inoltrare
- **OUTPUT**: pacchetti in uscita dall'host stesso
- **POSTROUTING**: pacchetti in uscita

Hooks

- **PREROUTING**: pacchetti in ingresso
- **INPUT**: pacchetti destinati all'host stesso
- **FORWARD**: pacchetti non destinati all'host stesso, che dobbiamo inoltrare
- **OUTPUT**: pacchetti in uscita dall'host stesso
- **POSTROUTING**: pacchetti in uscita

Hooks

- **PREROUTING**: pacchetti in ingresso
- **INPUT**: pacchetti destinati all'host stesso
- **FORWARD**: pacchetti non destinati all'host stesso, che dobbiamo inoltrare
- **OUTPUT**: pacchetti in uscita dall'host stesso
- **POSTROUTING**: pacchetti in uscita

Hooks

- **PREROUTING**: pacchetti in ingresso
- **INPUT**: pacchetti destinati all'host stesso
- **FORWARD**: pacchetti non destinati all'host stesso, che dobbiamo inoltrare
- **OUTPUT**: pacchetti in uscita dall'host stesso
- **POSTROUTING**: pacchetti in uscita

Netfilter

- Gli “hooks” non possono essere utilizzati direttamente. Non possiamo metterci delle regole dentro.
- Le regole devono essere inserite in delle **tabelle**. Ogni tabella ha dei punti di aggancio agli hook: le chain.
- Le tabelle utilizzate sono: *filter*, *mangle*, *nat* e *raw*
  - filter è la tabella con le regole principali
  - mangle è la tabella dove risiedono le regole che modificano i pacchetti
  - nat è la tabella dove avvengono le operazioni che permettono il NAT
  - *raw* (saltiamo...)

Netfilter

- Gli “hooks” non possono essere utilizzati direttamente. Non possiamo metterci delle regole dentro.
- Le regole devono essere inserite in delle **tabelle**. Ogni tabella ha dei punti di aggancio agli hook: le chain.
- Le tabelle utilizzate sono: *filter*, *mangle*, *nat* e *raw*
  - filter è la tabella con le regole principali
  - mangle è la tabella dove risiedono le regole che modificano i pacchetti
  - nat è la tabella dove avvengono le operazioni che permettono il NAT
  - *raw* (saltiamo...)

Netfilter

- Gli “hooks” non possono essere utilizzati direttamente. Non possiamo metterci delle regole dentro.
- Le regole devono essere inserite in delle **tabelle**. Ogni tabella ha dei punti di aggancio agli hook: le chain.
- Le tabelle utilizzate sono: *filter*, *mangle*, *nat* e *raw*
  - filter è la tabella con le regole principali
  - mangle è la tabella dove risiedono le regole che modificano i pacchetti
  - nat è la tabella dove avvengono le operazioni che permettono il NAT
  - *raw* (saltiamo...)

Netfilter

- Gli “hooks” non possono essere utilizzati direttamente. Non possiamo metterci delle regole dentro.
- Le regole devono essere inserite in delle **tabelle**. Ogni tabella ha dei punti di aggancio agli hook: le chain.
- Le tabelle utilizzate sono: *filter*, *mangle*, *nat* e *raw*
  - filter è la tabella con le regole principali
  - mangle è la tabella dove risiedono le regole che modificano i pacchetti
  - nat è la tabella dove avvengono le operazioni che permettono il NAT
  - *raw* (saltiamo...)

Netfilter

- Gli “hooks” non possono essere utilizzati direttamente. Non possiamo metterci delle regole dentro.
- Le regole devono essere inserite in delle **tabelle**. Ogni tabella ha dei punti di aggancio agli hook: le chain.
- Le tabelle utilizzate sono: *filter*, *mangle*, *nat* e *raw*
  - filter è la tabella con le regole principali
  - mangle è la tabella dove risiedono le regole che modificano i pacchetti
  - nat è la tabella dove avvengono le operazioni che permettono il NAT
  - *raw* (saltiamo...)

Netfilter

- Gli “hooks” non possono essere utilizzati direttamente. Non possiamo metterci delle regole dentro.
- Le regole devono essere inserite in delle **tabelle**. Ogni tabella ha dei punti di aggancio agli hook: le chain.
- Le tabelle utilizzate sono: *filter*, *mangle*, *nat* e *raw*
  - filter è la tabella con le regole principali
  - mangle è la tabella dove risiedono le regole che modificano i pacchetti
  - nat è la tabella dove avvengono le operazioni che permettono il NAT
  - *raw* (saltiamo...)

Netfilter

- Riassumendo:
  - il pacchetto segue il suo percorso
  - quando il pacchetto arriva in un hook si vanno a consultare le varie tabelle che hanno regole in quell'hook (le tabelle da verificare sono in un certo ordine)
  - il pacchetto entra nella *chain* di una tabella e lì vengono consultate tutte le regole presenti
  - se al termine delle verifiche, la decisione è "ACCEPT", allora il pacchetto prosegue il suo percorso e viene verificato nelle tabelle successive
  - poi se il pacchetto prosegue ancora passerà nell'hook successivo

Netfilter

- Riassumendo:
  - il pacchetto segue il suo percorso
  - quando il pacchetto arriva in un hook si vanno a consultare le varie tabelle che hanno regole in quell'hook (le tabelle da verificare sono in un certo ordine)
  - il pacchetto entra nella *chain* di una tabella e lì vengono consultate tutte le regole presenti
  - se al termine delle verifiche, la decisione è "ACCEPT", allora il pacchetto prosegue il suo percorso e viene verificato nelle tabelle successive
  - poi se il pacchetto prosegue ancora passerà nell'hook successivo

Netfilter

- Riassumendo:
  - il pacchetto segue il suo percorso
  - quando il pacchetto arriva in un hook si vanno a consultare le varie tabelle che hanno regole in quell'hook (le tabelle da verificare sono in un certo ordine)
  - il pacchetto entra nella *chain* di una tabella e lì vengono consultate tutte le regole presenti
  - se al termine delle verifiche, la decisione è "ACCEPT", allora il pacchetto prosegue il suo percorso e viene verificato nelle tabelle successive
  - poi se il pacchetto prosegue ancora passerà nell'hook successivo

Netfilter

- Riassumendo:
  - il pacchetto segue il suo percorso
  - quando il pacchetto arriva in un hook si vanno a consultare le varie tabelle che hanno regole in quell'hook (le tabelle da verificare sono in un certo ordine)
  - il pacchetto entra nella *chain* di una tabella e lì vengono consultate tutte le regole presenti
  - se al termine delle verifiche, la decisione è "ACCEPT", allora il pacchetto prosegue il suo percorso e viene verificato nelle tabelle successive
  - poi se il pacchetto prosegue ancora passerà nell'hook successivo

Netfilter

- Riassumendo:
  - il pacchetto segue il suo percorso
  - quando il pacchetto arriva in un hook si vanno a consultare le varie tabelle che hanno regole in quell'hook (le tabelle da verificare sono in un certo ordine)
  - il pacchetto entra nella *chain* di una tabella e lì vengono consultate tutte le regole presenti
  - se al termine delle verifiche, la decisione è "ACCEPT", allora il pacchetto prosegue il suo percorso e viene verificato nelle tabelle successive
  - poi se il pacchetto prosegue ancora passerà nell'hook successivo

iptables

- Iniziamo a guardare la tabella di default, la tabella filter:

```
$ sudo iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out     source               destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out     source               destination

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

iptables

- Nella tabella filter abbiamo 3 chains che si agganciano agli hooks di **INPUT**, **FORWARD** e **OUTPUT**.
- Durante il percorso del pacchetto, quando si arriva nella chain della tabella, si analizzano una per una le regole presenti dalla prima all'ultima. Se nessuna regola fa **match**, si applica la *policy di default* della chain.
- La policy di default può essere ACCEPT o DROP.

iptables

- Nella tabella filter abbiamo 3 chains che si agganciano agli hooks di **INPUT**, **FORWARD** e **OUTPUT**.
- Durante il percorso del pacchetto, quando si arriva nella chain della tabella, si analizzano una per una le regole presenti dalla prima all'ultima. Se nessuna regola fa **match**, si applica la *policy di default* della chain.
- La policy di default può essere ACCEPT o DROP.

iptables

- Nella tabella filter abbiamo 3 chains che si agganciano agli hooks di **INPUT**, **FORWARD** e **OUTPUT**.
- Durante il percorso del pacchetto, quando si arriva nella chain della tabella, si analizzano una per una le regole presenti dalla prima all'ultima. Se nessuna regola fa **match**, si applica la *policy di default* della chain.
- La policy di default può essere ACCEPT o DROP.

iptables

- Ogni **regola** è costituta da due parti principali:
  - *Match:* È la parte della regola che specifica quali sono le caratteristiche del pacchetto che cerchiamo
  - *Target*: È la parte della regola che specifica che operazione effettuare sul pacchetto che fa match
- Solitamente si usa il minuscono per i nomi dei match e il maiuscolo per i TARGET.

iptables

- Ogni **regola** è costituta da due parti principali:
  - *Match:* È la parte della regola che specifica quali sono le caratteristiche del pacchetto che cerchiamo
  - *Target*: È la parte della regola che specifica che operazione effettuare sul pacchetto che fa match
- Solitamente si usa il minuscono per i nomi dei match e il maiuscolo per i TARGET.

sintassi di iptables

```
iptables -A INPUT --source 192.168.0.153 -j DROP
```

- **-A** significa che vogliamo aggiungere questa regola (in coda alle altre) nella chain **INPUT** (della tabella implicita filter)
- **–source** indica la condizione che il pacchetto IPv4 deve avere come sorgente l'indirizzo 192.168.0.153
- **-j DROP** specifica che se le precedenti condizioni di match sono soddisfatte, l'operazione da effettuare è un **jump** al target DROP, cioè scartiamo il pacchetto.
  - Il pacchetto viene scartato e **non** proseguirà nelle altre regole.

sintassi di iptables

```
iptables -A INPUT --source 192.168.0.153 -j DROP
```

- **-A** significa che vogliamo aggiungere questa regola (in coda alle altre) nella chain **INPUT** (della tabella implicita filter)
- **–source** indica la condizione che il pacchetto IPv4 deve avere come sorgente l'indirizzo 192.168.0.153
- **-j DROP** specifica che se le precedenti condizioni di match sono soddisfatte, l'operazione da effettuare è un **jump** al target DROP, cioè scartiamo il pacchetto.
  - Il pacchetto viene scartato e **non** proseguirà nelle altre regole.

sintassi di iptables

```
iptables -A INPUT --source 192.168.0.153 -j DROP
```

- **-A** significa che vogliamo aggiungere questa regola (in coda alle altre) nella chain **INPUT** (della tabella implicita filter)
- **–source** indica la condizione che il pacchetto IPv4 deve avere come sorgente l'indirizzo 192.168.0.153
- **-j DROP** specifica che se le precedenti condizioni di match sono soddisfatte, l'operazione da effettuare è un **jump** al target DROP, cioè scartiamo il pacchetto.
  - Il pacchetto viene scartato e **non** proseguirà nelle altre regole.

sintassi di iptables

```
iptables -A INPUT --source 192.168.0.153 -j DROP
```

- **-A** significa che vogliamo aggiungere questa regola (in coda alle altre) nella chain **INPUT** (della tabella implicita filter)
- **–source** indica la condizione che il pacchetto IPv4 deve avere come sorgente l'indirizzo 192.168.0.153
- **-j DROP** specifica che se le precedenti condizioni di match sono soddisfatte, l'operazione da effettuare è un **jump** al target DROP, cioè scartiamo il pacchetto.
  - Il pacchetto viene scartato e **non** proseguirà nelle altre regole.

sintassi di iptables

- Nel match possiamo sempre utilizzare la negazione mettendo un punto esclamativo davanti ad una delle condizioni del match.
- Ad esempio:

```
iptables -A INPUT -s 192.168.0.0/24 ! -i eth1 -j DROP
```

  - *-s* è come *–source*, utilizzando la notazione VLSM "/24" specifichiamo il **prefisso**, non l'indirizzo esatto.
  - **-i** specifica il match sull'interfaccia di arrivo del pacchetto

sintassi di iptables

- Nel match possiamo sempre utilizzare la negazione mettendo un punto esclamativo davanti ad una delle condizioni del match.
- Ad esempio:

```
iptables -A INPUT -s 192.168.0.0/24 ! -i eth1 -j DROP
```

  - *-s* è come *–source*, utilizzando la notazione VLSM "/24" specifichiamo il **prefisso**, non l'indirizzo esatto.
  - **-i** specifica il match sull'interfaccia di arrivo del pacchetto

sintassi di iptables

- Nel match possiamo sempre utilizzare la negazione mettendo un punto esclamativo davanti ad una delle condizioni del match.
- Ad esempio:

```
iptables -A INPUT -s 192.168.0.0/24 ! -i eth1 -j DROP
```

  - *-s* è come *–source*, utilizzando la notazione VLSM "/24" specifichiamo il **prefisso**, non l'indirizzo esatto.
  - **-i** specifica il match sull'interfaccia di arrivo del pacchetto

sintassi di iptables

```
iptables -A [CHAIN] [condizioni di match] -j [target]

iptables -D [CHAIN] [condizioni di match] -j [target]

iptables -I [CHAIN] [condizioni di match] -j [target]

iptables -P [CHAIN] [ACCEPT|DROP]
```

- Policy di default della chain

```
iptables -F [CHAIN]
```

- Per fare un flush delle regole nella chain (senza cambiare la policy di default)

sintassi di iptables

```
iptables -A [CHAIN] [condizioni di match] -j [target]

iptables -D [CHAIN] [condizioni di match] -j [target]

iptables -I [CHAIN] [condizioni di match] -j [target]

iptables -P [CHAIN] [ACCEPT|DROP]
```

- Policy di default della chain

```
iptables -F [CHAIN]
```

- Per fare un flush delle regole nella chain (senza cambiare la policy di default)

sintassi di iptables

```
iptables -A [CHAIN] [condizioni di match] -j [target]

iptables -D [CHAIN] [condizioni di match] -j [target]

iptables -I [CHAIN] [condizioni di match] -j [target]
```

```
iptables -P [CHAIN] [ACCEPT|DROP]
```

- Policy di default della chain

```
iptables -F [CHAIN]
```

- Per fare un flush delle regole nella chain (senza cambiare la policy di default)

sintassi di iptables

```
iptables -A [CHAIN] [condizioni di match] -j [target]

iptables -D [CHAIN] [condizioni di match] -j [target]

iptables -I [CHAIN] [condizioni di match] -j [target]

iptables -P [CHAIN] [ACCEPT|DROP]
```

- Policy di default della chain

```
iptables -F [CHAIN]
```

- Per fare un flush delle regole nella chain (senza cambiare la policy di default)

sintassi di iptables

```
iptables -A [CHAIN] [condizioni di match] -j [target]

iptables -D [CHAIN] [condizioni di match] -j [target]

iptables -I [CHAIN] [condizioni di match] -j [target]
```

```
iptables -P [CHAIN] [ACCEPT|DROP]
```

- Policy di default della chain

```
iptables -F [CHAIN]
```

- Per fare un flush delle regole nella chain (senza cambiare la policy di default)

Cancellare una regola

- Possiamo riscriverla uguale mettendo -D al posto di -A
- Oppure possiamo elencare le regole con i numeri a fianco

```
$ iptables -nvL --line-numbers
```

- ...e cancellare in base al numero:

```
$ iptables -t [table] -D [CHAIN] [numero regola]
```

Cancellare una regola

- Possiamo riscriverla uguale mettendo -D al posto di -A
- Oppure possiamo elencare le regole con i numeri a fianco

```
$ iptables -nvL --line-numbers
```

- ...e cancellare in base al numero:

```
$ iptables -t [table] -D [CHAIN] [numero regola]
```

ACCEPT/DROP

- Abbiamo visto che con DROP il pacchetto viene fermato.
- Cosa succede con ACCEPT ?
  - Il pacchetto viene accettato e passa alla chain successiva.
  - Nel caso di INPUT non c'è una chain successiva, quindi viene passato al sistema operativo
  - Però se siamo in PREROUTING ad esempio , non è detto che il pacchetto arrivi, potrebbe essere bloccato in INPUT.

ACCEPT/DROP

- Abbiamo visto che con DROP il pacchetto viene fermato.
- Cosa succede con ACCEPT ?
  - Il pacchetto viene accettato e passa alla chain successiva.
  - Nel caso di INPUT non c'è una chain successiva, quindi viene passato al sistema operativo
  - Però se siamo in PREROUTING ad esempio , non è detto che il pacchetto arrivi, potrebbe essere bloccato in INPUT.

ACCEPT/DROP

- Abbiamo visto che con DROP il pacchetto viene fermato.
- Cosa succede con ACCEPT ?
  - Il pacchetto viene accettato e passa alla chain successiva.
  - Nel caso di INPUT non c'è una chain successiva, quindi viene passato al sistema operativo
  - Però se siamo in PREROUTING ad esempio , non è detto che il pacchetto arrivi, potrebbe essere bloccato in INPUT.

Match comuni

- Interfaccia di rete di *input*/*output* del pacchetto
  - ***-i [iface]***
  - ***-o [iface]***
  - Non possiamo usarli sempre questi match. Ad esempio non ha senso usare -i nella chain di OUTPUT.
- Indirizzo sorgente/destinazione
  - ***-s [address or network prefix]***
  - ***-d [address or network prefix]***
- Protocollo di livello 4 e numeri di porta destinazione
  - ***-p udp –dport 53***
  - **-p tcp –dport 22**

Match comuni

- Interfaccia di rete di *input*/*output* del pacchetto
  - ***-i [iface]***
  - ***-o [iface]***
  - Non possiamo usarli sempre questi match. Ad esempio non ha senso usare -i nella chain di OUTPUT.
- Indirizzo sorgente/destinazione
  - ***-s [address or network prefix]***
  - ***-d [address or network prefix]***
- Protocollo di livello 4 e numeri di porta destinazione
  - ***-p udp –dport 53***
  - **-p tcp –dport 22**

Match comuni

- Interfaccia di rete di *input*/*output* del pacchetto
  - ***-i [iface]***
  - ***-o [iface]***
  - Non possiamo usarli sempre questi match. Ad esempio non ha senso usare -i nella chain di OUTPUT.
- Indirizzo sorgente/destinazione
  - ***-s [address or network prefix]***
  - ***-d [address or network prefix]***
- Protocollo di livello 4 e numeri di porta destinazione
  - ***-p udp –dport 53***
  - **-p tcp –dport 22**

Match stateful

- Tutti i match che abbiamo visto fino ad ora sono "integrati" in netfilter. Molti altri match e quelli aggiuntivi necessitano quasi sempre di "***-m nome_modulo_match***", per dire a iptables di caricare il modulo relativo.
- Possiamo fare match sullo stato della connessione correlata al pacchetto utilizzando il modulo di tracking delle connessioni: **conntrack**

```
$ iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED
```

- Questa regola dice se il pacchetto è aprte di una connnessione stabilita (3-way handshake tcp completato, o nel caso di UDP è già "passato qualcosa recentemente")

Match stateful

- Tutti i match che abbiamo visto fino ad ora sono "integrati" in netfilter. Molti altri match e quelli aggiuntivi necessitano quasi sempre di "***-m nome_modulo_match***", per dire a iptables di caricare il modulo relativo.

- Possiamo fare match sullo stato della connessione correlata al pacchetto utilizzando il modulo di tracking delle connessioni: **conntrack**

```
$ iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED
```

- Questa regola dice se il pacchetto è aprte di una connnessione stabilita (3-way handshake tcp completato, o nel caso di UDP è già "passato qualcosa recentemente")

Target comuni

- ACCEPT/DROP
- REJECT: Come DROP ma genera un messaggio di errore ICMP o TCP che viene rispedito al mittente
  - Ad esempio:

    ```
    -j REJECT --reject-with tcp-reset
    ```
- LOG: Salva una riga di log relativa al pacchetto che ha fatto match nei log di sistema (dmesg)

RST o DROP?

Normalmente si preferisce non inviare nessun REJECT. Con il REJECT si spreca banda e si è leggermente vulnerabili ad attacchi DoS, oltre a rendere più facile il port scanning.

Blacklisting o whitelisting

- “Vogliamo fare una festa. Facciamo la lista di tutti gli invitati o la lista di tutti i ’non invitati’ ? “ (cit.)
- Si fa sempre *whitelisting*. Se state facendo blacklisting state sbagliando qualcosa.
- Solitamente si impostano le policy di default di INPUT e FORWARD a DROP, e poi si inseriscono una serie di regole per fare whitelisting del traffico legittimo. Di solito si lascia OUTPUT in ACCEPT, non ha molto senso bloccare anche quello che generiamo noi.

Blacklisting o whitelisting

- “Vogliamo fare una festa. Facciamo la lista di tutti gli invitati o la lista di tutti i ’non invitati’ ? “ (cit.)
- Si fa sempre *whitelisting*. Se state facendo blacklisting state sbagliando qualcosa.
- Solitamente si impostano le policy di default di INPUT e FORWARD a DROP, e poi si inseriscono una serie di regole per fare whitelisting del traffico legittimo. Di solito si lascia OUTPUT in ACCEPT, non ha molto senso bloccare anche quello che generiamo noi.

Blacklisting o whitelisting

- “Vogliamo fare una festa. Facciamo la lista di tutti gli invitati o la lista di tutti i ’non invitati’ ? “ (cit.)
- Si fa sempre *whitelisting*. Se state facendo blacklisting state sbagliando qualcosa.
- Solitamente si impostano le policy di default di INPUT e FORWARD a DROP, e poi si inseriscono una serie di regole per fare whitelisting del traffico legittimo. Di solito si lascia OUTPUT in ACCEPT, non ha molto senso bloccare anche quello che generiamo noi.

Interfaccia di loopback

- L'interfaccia di rete "lo" è quella associata all'indirizzo 127.0.0.1 e tutta la rete 127.0.0.0/8
- Molti programmi utilizzano questo indirizzamento per comunicazioni tra processi e altro.
- **Non** dobbiamo bloccare questo traffico locale.

```
$ iptables -A INPUT -i lo -j ACCEPT
```

Interfaccia di loopback

- L'interfaccia di rete "lo" è quella associata all'indirizzo 127.0.0.1 e tutta la rete 127.0.0.0/8
- Molti programmi utilizzano questo indirizzamento per comunicazioni tra processi e altro.
- **Non** dobbiamo bloccare questo traffico locale.

```
$ iptables -A INPUT -i lo -j ACCEPT
```

Interfaccia di loopback

- L'interfaccia di rete "lo" è quella associata all'indirizzo 127.0.0.1 e tutta la rete 127.0.0.0/8
- Molti programmi utilizzano questo indirizzamento per comunicazioni tra processi e altro.
- **Non** dobbiamo bloccare questo traffico locale.

```
$ iptables -A INPUT -i lo -j ACCEPT
```

ICMP

- ICMP è un protocollo di servizio che si occupa di trasmettere informazioni riguardanti malfunzionamenti, informazioni di controllo o messaggi tra i vari componenti di una rete.
- Il tipo di pacchetto ICMP più famoso è senza dubbio la "echo request", che viene inviata con il comando ping e serve a sapere se un certo host è attivo (e viene utilizzato per controllare se arrivano i pacchetti o ci sono problemi sulla rete)
- A meno che non abbiate specifiche esigenze (ad esempio nascondere il fatto che il vostro host è acceso) **è considerata pratica scorretta oltre che "maleducazione" ignorare i messaggi ICMP**
- Se volete essere gentili semplicemente accettate tutto:

```
$ iptables -A INPUT -p icmp -j ACCEPT
```

ICMP

- ICMP è un protocollo di servizio che si occupa di trasmettere informazioni riguardanti malfunzionamenti, informazioni di controllo o messaggi tra i vari componenti di una rete.
- Il tipo di pacchetto ICMP più famoso è senza dubbio la "echo request", che viene inviata con il comando ping e serve a sapere se un certo host è attivo (e viene utilizzato per controllare se arrivano i pacchetti o ci sono problemi sulla rete)
- A meno che non abbiate specifiche esigenze (ad esempio nascondere il fatto che il vostro host è acceso) **è considerata pratica scorretta oltre che "maleducazione" ignorare i messaggi ICMP**
- Se volete essere gentili semplicemente accettate tutto:

```
$ iptables -A INPUT -p icmp -j ACCEPT
```

ICMP

- ICMP è un protocollo di servizio che si occupa di trasmettere informazioni riguardanti malfunzionamenti, informazioni di controllo o messaggi tra i vari componenti di una rete.
- Il tipo di pacchetto ICMP più famoso è senza dubbio la "echo request", che viene inviata con il comando ping e serve a sapere se un certo host è attivo (e viene utilizzato per controllare se arrivano i pacchetti o ci sono problemi sulla rete)
- A meno che non abbiate specifiche esigenze (ad esempio nascondere il fatto che il vostro host è acceso) **è considerata pratica scorretta oltre che "maleducazione" ignorare i messaggi ICMP**
- Se volete essere gentili semplicemente accettate tutto:

```
$ iptables -A INPUT -p icmp -j ACCEPT
```

ICMP

- ICMP è un protocollo di servizio che si occupa di trasmettere informazioni riguardanti malfunzionamenti, informazioni di controllo o messaggi tra i vari componenti di una rete.
- Il tipo di pacchetto ICMP più famoso è senza dubbio la "echo request", che viene inviata con il comando ping e serve a sapere se un certo host è attivo (e viene utilizzato per controllare se arrivano i pacchetti o ci sono problemi sulla rete)
- A meno che non abbiate specifiche esigenze (ad esempio nascondere il fatto che il vostro host è acceso) **è considerata pratica scorretta oltre che "maleducazione" ignorare i messaggi ICMP**
- Se volete essere gentili semplicemente accettate tutto:

```
$ iptables -A INPUT -p icmp -j ACCEPT
```

ICMP

Se volete evitare di accettare propro tutto il traffico, almeno accettate i tipi di messaggio ICMP fondamentali:

```
$ iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
$ iptables -A INPUT -p icmp --icmp-type 3 -j ACCEPT
$ iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
```

- Che corrispondono a:
  - Echo request (type 8)
  - Destination Unreachable (type 3)
  - Echo reply (type 0)

Esempio di config essenziale

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 3 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW -j ACCEPT
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP   #Se non vogliamo fare da router a nessuno....
```

Chiudersi fuori di casa

- Fate molta attenzione se state configurando una macchina remota (tipo via ssh) quando impostate le policy di default!
- Caso tipico:
  - "Oh cavolo, non ho impostato nessun firewall su questa macchina...
  - beh... per prima cosa policy di default DROP!"
  - "Perché non risponde più il server?"
  - Tramite cosa stavate comunicando? (-.-')

Chiudersi fuori di casa

- Fate molta attenzione se state configurando una macchina remota (tipo via ssh) quando impostate le policy di default!
- Caso tipico:
  - "Oh cavolo, non ho impostato nessun firewall su questa macchina...
  - beh... per prima cosa policy di default DROP!"
  - "Perché non risponde più il server?"
  - Tramite cosa stavate comunicando? (-.-')

Chiudersi fuori di casa

- Fate molta attenzione se state configurando una macchina remota (tipo via ssh) quando impostate le policy di default!
- Caso tipico:
  - "Oh cavolo, non ho impostato nessun firewall su questa macchina...
  - beh... per prima cosa policy di default DROP!"
  - "Perché non risponde più il server?"
  - Tramite cosa stavate comunicando? (-.-')

Chiudersi fuori di casa

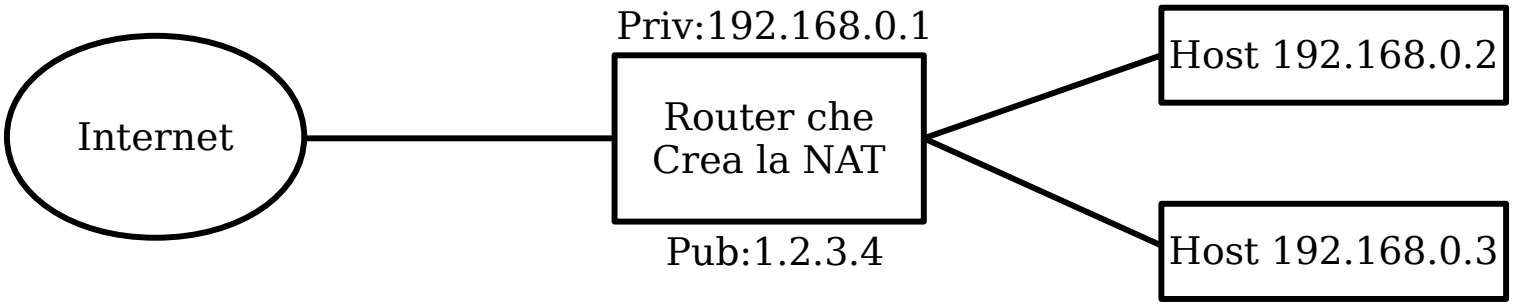
- Fate molta attenzione se state configurando una macchina remota (tipo via ssh) quando impostate le policy di default!
- Caso tipico:
  - "Oh cavolo, non ho impostato nessun firewall su questa macchina...
  - beh... per prima cosa policy di default DROP!"
  - "Perché non risponde più il server?"
  - Tramite cosa stavate comunicando? (-.-')

Chiudersi fuori di casa

Ricordatevi sempre di impostare le policy di default come ultimo passo, dopo esservi assicurati di aver messo una regola per accettare le connessioni nuove e stabilite per la vostra shell (ssh solitamente).

NAT - Network Address Translation

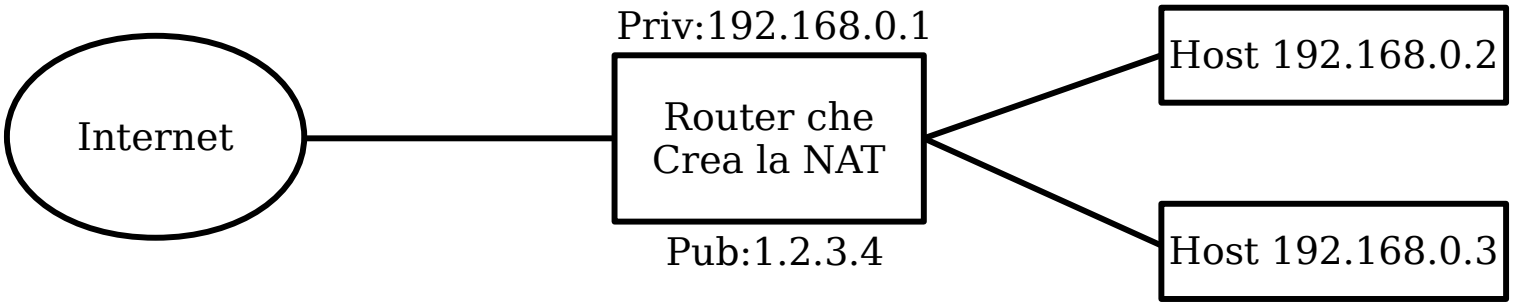
Come funziona?



- Quando un host, ad es. 192.168.0.2 vuole connettersi ad un server esterno, ad es. 5.6.7.8, invia i pacchetti al router di frontiera (gateway) che prende l'indirizzo sorgente e lo sostituisce con il proprio (1.2.3.4).
- Il router sostituisce la mia porta TCP sorgente con una sua porta libera e si annota questo scambio in una tabella di connessioni stabilite.
- Quando arrivano le risposte dal server (solo risposte relative ad una connessione già stabilita), il router capisce chi è il destinatario nella rete locale e sostiuisce l'indirizzo di destinazione (1.2.3.4) con l'indirizzo privato 192.168.0.2 e con la porta destinazione corretta.

NAT - Network Address Translation

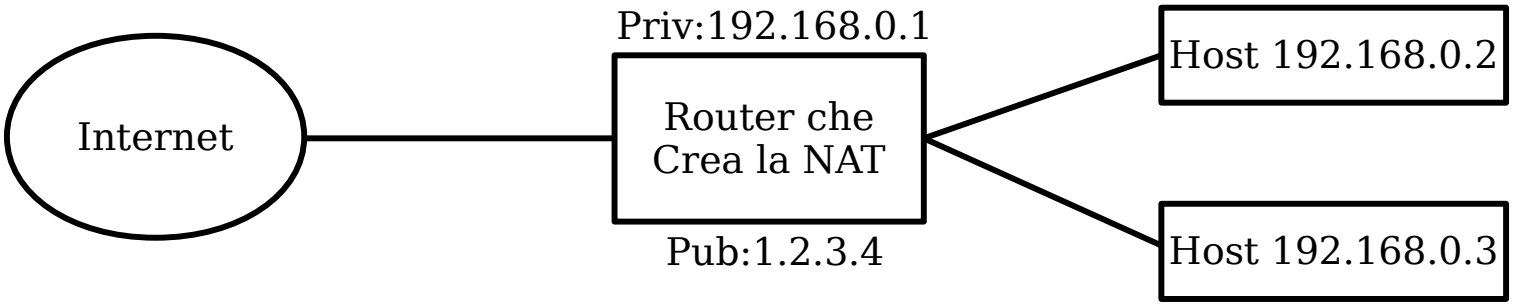
Come funziona?



- Quando un host, ad es. 192.168.0.2 vuole connettersi ad un server esterno, ad es. 5.6.7.8, invia i pacchetti al router di frontiera (gateway) che prende l'indirizzo sorgente e lo sostituisce con il proprio (1.2.3.4).
- Il router sostituisce la mia porta TCP sorgente con una sua porta libera e si annota questo scambio in una tabella di connessioni stabilite.
- Quando arrivano le risposte dal server (solo risposte relative ad una connessione già stabilita), il router capisce chi è il destinatario nella rete locale e sostiuisce l'indirizzo di destinazione (1.2.3.4) con l'indirizzo privato 192.168.0.2 e con la porta destinazione corretta.

NAT - Network Address Translation

Come funziona?



- Quando un host, ad es. 192.168.0.2 vuole connettersi ad un server esterno, ad es. 5.6.7.8, invia i pacchetti al router di frontiera (gateway) che prende l'indirizzo sorgente e lo sostituisce con il proprio (1.2.3.4).
- Il router sostituisce la mia porta TCP sorgente con una sua porta libera e si annota questo scambio in una tabella di connessioni stabilite.
- Quando arrivano le risposte dal server (solo risposte relative ad una connessione già stabilita), il router capisce chi è il destinatario nella rete locale e sostiuisce l'indirizzo di destinazione (1.2.3.4) con l'indirizzo privato 192.168.0.2 e con la porta destinazione corretta.

NAT: un abominio

La NAT sarà tanto bella e comoda ma è un workaround (temporaneo fino ad IPv6). Quando è possibile non andrebbe usata. Specialmente bisognerebbe **evitare di fare doppie NAT** o altre cose simili..

- NAT viola il modello gerarchico di IP
- I processi su Internet non sono obbligati ad utilizzare TCP e UDP (vedi SCTP - RFC 2960)
- Il numero di connessioni contemporanee diminuisce a causa del numero limitato di numeri di porte
- NAPT trasforma Internet da una rete ad assenza di connessione
- La NAT **da sola** non è un meccanismo firewalling. Le caratteristiche **stateful** del firewall che implementa la NAT proteggono, non la NAT stessa.

Approfondimenti su `http://tinyurl.com/nathorror`

NAT: un abominio

La NAT sarà tanto bella e comoda ma è un workaround (temporaneo fino ad IPv6). Quando è possibile non andrebbe usata. Specialmente bisognerebbe **evitare di fare doppie NAT** o altre cose simili..

- NAT viola il modello gerarchico di IP
- I processi su Internet non sono obbligati ad utilizzare TCP e UDP (vedi SCTP - RFC 2960)
- Il numero di connessioni contemporanee diminuisce a causa del numero limitato di numeri di porte
- NAPT trasforma Internet da una rete ad assenza di connessione
- La NAT **da sola** non è un meccanismo firewalling. Le caratteristiche **stateful** del firewall che implementa la NAT proteggono, non la NAT stessa.

Approfondimenti su `http://tinyurl.com/nathorror`

NAT: un abominio

La NAT sarà tanto bella e comoda ma è un workaround (temporaneo fino ad IPv6). Quando è possibile non andrebbe usata. Specialmente bisognerebbe **evitare di fare doppie NAT** o altre cose simili..

- NAT viola il modello gerarchico di IP
- I processi su Internet non sono obbligati ad utilizzare TCP e UDP (vedi SCTP - RFC 2960)
- Il numero di connessioni contemporanee diminuisce a causa del numero limitato di numeri di porte
- NAPT trasforma Internet da una rete ad assenza di connessione
- La NAT **da sola** non è un meccanismo firewalling. Le caratteristiche **stateful** del firewall che implementa la NAT proteggono, non la NAT stessa.

Approfondimenti su `http://tinyurl.com/nathorror`

NAT: un abominio

La NAT sarà tanto bella e comoda ma è un workaround (temporaneo fino ad IPv6). Quando è possibile non andrebbe usata. Specialmente bisognerebbe **evitare di fare doppie NAT** o altre cose simili..

- NAT viola il modello gerarchico di IP
- I processi su Internet non sono obbligati ad utilizzare TCP e UDP (vedi SCTP - RFC 2960)
- Il numero di connessioni contemporanee diminuisce a causa del numero limitato di numeri di porte
- NAPT trasforma Internet da una rete ad assenza di connessione
- La NAT **da sola** non è un meccanismo firewalling. Le caratteristiche **stateful** del firewall che implementa la NAT proteggono, non la NAT stessa.

Approfondimenti su `http://tinyurl.com/nathorror`

NAT: un abominio

La NAT sarà tanto bella e comoda ma è un workaround (temporaneo fino ad IPv6). Quando è possibile non andrebbe usata. Specialmente bisognerebbe **evitare di fare doppie NAT** o altre cose simili..

- NAT viola il modello gerarchico di IP
- I processi su Internet non sono obbligati ad utilizzare TCP e UDP (vedi SCTP - RFC 2960)
- Il numero di connessioni contemporanee diminuisce a causa del numero limitato di numeri di porte
- NAPT trasforma Internet da una rete ad assenza di connessione
- La NAT **da sola** non è un meccanismo firewalling. Le caratteristiche **stateful** del firewall che implementa la NAT proteggono, non la NAT stessa.

Approfondimenti su `http://tinyurl.com/nathorror`

NAT: un abominio

- Se avete una valanga di IP pubblici o siete già dentro ad una NAT, evitate di farne "yet another one", grazie.
- Purtroppo molte volte c'è poco da fare (abbiamo solo un indirizzo IP pubblico e tanti host da collegare) e siamo costretti ad usarla.
- Vediamo allora come fare...

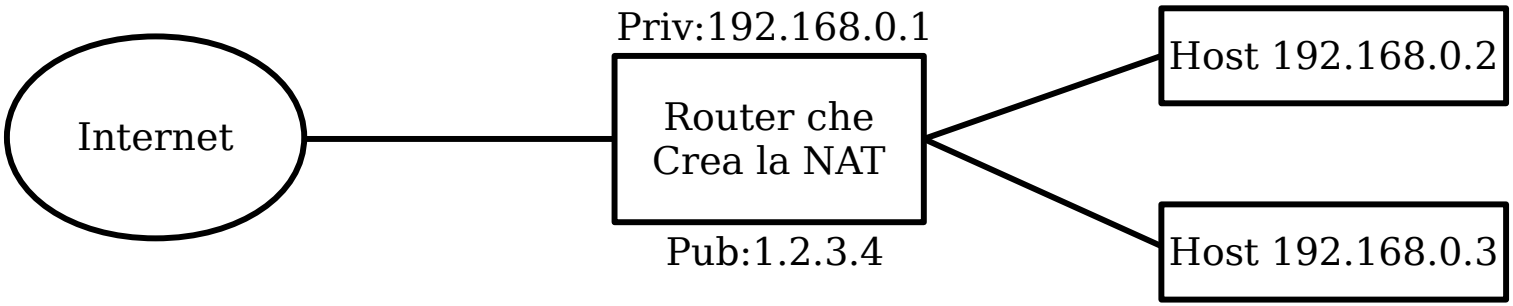
NAT: un abominio

- Se avete una valanga di IP pubblici o siete già dentro ad una NAT, evitate di farne "yet another one", grazie.
- Purtroppo molte volte c'è poco da fare (abbiamo solo un indirizzo IP pubblico e tanti host da collegare) e siamo costretti ad usarla.
- Vediamo allora come fare...

NAT: un abominio

- Se avete una valanga di IP pubblici o siete già dentro ad una NAT, evitate di farne "yet another one", grazie.
- Purtroppo molte volte c'è poco da fare (abbiamo solo un indirizzo IP pubblico e tanti host da collegare) e siamo costretti ad usarla.
- Vediamo allora come fare...

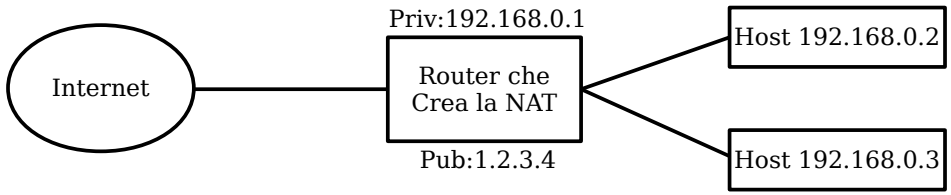
NAT




```
iptables -A FORWARD -s 192.168.0.0/24 -i eth1 -o eth0 -j ACCEPT

iptables -A FORWARD -i eth0 -o eth1 -m state --state RELATED,ESTABLISHED -j ACCEPT

iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o eth0 -j SNAT --to-source 1.2.3.4


echo "1" > /proc/sys/net/ipv4/ip_forward
```

NAT

- Se il router ha un indirizzo IP dinamico potete utilizzare **-j MASQUERADE** invece di SNAT, che identifica automaticamente l'indirizzo dell'interfaccia utilizzata per la route di default.
- Come facciamo se vogliamo mettere su un servizio (ad. es. un server web) sull'host 192.168.0.3?

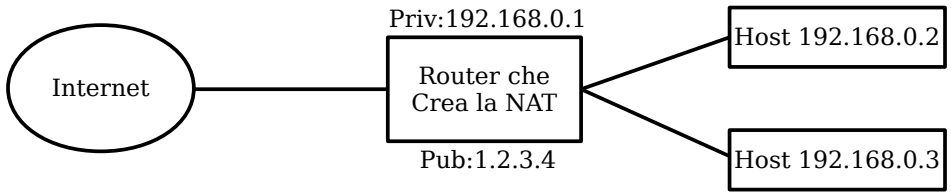



```
$ iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -j ACCEPT
$ iptables -t nat -A PREROUTING -d 1.2.3.4 -i eth0 -j DNAT --to-destination 19
```

NAT

- Se il router ha un indirizzo IP dinamico potete utilizzare **-j MASQUERADE** invece di SNAT, che identifica automaticamente l'indirizzo dell'interfaccia utilizzata per la route di default.

- Come facciamo se vogliamo mettere su un servizio (ad. es. un server web) sull'host 192.168.0.3?

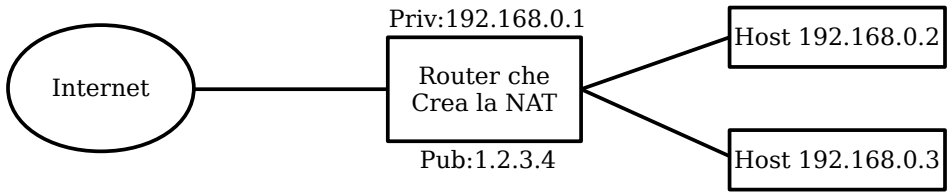



```
$ iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -j ACCEPT
$ iptables -t nat -A PREROUTING -d 1.2.3.4 -i eth0 -j DNAT --to-destination 19
```

NAT

- Se il router ha un indirizzo IP dinamico potete utilizzare **-j MASQUERADE** invece di SNAT, che identifica automaticamente l'indirizzo dell'interfaccia utilizzata per la route di default.

- Come facciamo se vogliamo mettere su un servizio (ad. es. un server web) sull'host 192.168.0.3?

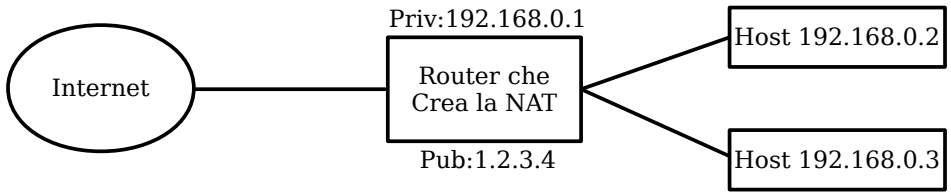



```
$ iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -j ACCEPT
$ iptables -t nat -A PREROUTING -d 1.2.3.4 -i eth0 -j DNAT --to-destination 19
```

NAT

- Se il router ha un indirizzo IP dinamico potete utilizzare **-j MASQUERADE** invece di SNAT, che identifica automaticamente l'indirizzo dell'interfaccia utilizzata per la route di default.

- Come facciamo se vogliamo mettere su un servizio (ad. es. un server web) sull'host 192.168.0.3?




```
$ iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 80 -j ACCEPT
$ iptables -t nat -A PREROUTING -d 1.2.3.4 -i eth0 -j DNAT --to-destination 19
```

IPv6

- Breve parentesi: Linux supporta nativamente Ipv6 fin dai primi standard.
  - Le assegnazioni di blocchi IPv4 da parte della IANA sono state esaurite nel 2011.
  - In tutti i grandi datacenter internazionali IPv6 è già attivo e utilizzato.
  - In netfilter ipv6 è già supportato.
- Per chiarezza il tool userspace per gestire le regole relative al traffico IPv6 è chiamato "ip6tables".
  - Funziona esattamente allo stesso modo di iptables normale.
  - La principale differenza è che non esiste la NAT
  - Una nota: Alcune funzionalità di ICMPv6 sono obbligatorie (Neighbor Discovery Protocol, type 135 e 136) e se le bloccate avrete problemi di connessione.

```
ip6tables -A INPUT -p ipv6-icmp -j ACCEPT
```

IPv6

- Breve parentesi: Linux supporta nativamente Ipv6 fin dai primi standard.
  - Le assegnazioni di blocchi IPv4 da parte della IANA sono state esaurite nel 2011.
  - In tutti i grandi datacenter internazionali IPv6 è già attivo e utilizzato.
  - In netfilter ipv6 è già supportato.
- Per chiarezza il tool userspace per gestire le regole relative al traffico IPv6 è chiamato "ip6tables".
  - Funziona esattamente allo stesso modo di iptables normale.
  - La principale differenza è che non esiste la NAT
  - Una nota: Alcune funzionalità di ICMPv6 sono obbligatorie (Neighbor Discovery Protocol, type 135 e 136) e se le bloccate avrete problemi di connessione.

```
ip6tables -A INPUT -p ipv6-icmp -j ACCEPT
```

IPv6

- Breve parentesi: Linux supporta nativamente Ipv6 fin dai primi standard.
  - Le assegnazioni di blocchi IPv4 da parte della IANA sono state esaurite nel 2011.
  - In tutti i grandi datacenter internazionali IPv6 è già attivo e utilizzato.
  - In netfilter ipv6 è già supportato.
- Per chiarezza il tool userspace per gestire le regole relative al traffico IPv6 è chiamato "ip6tables".
  - Funziona esattamente allo stesso modo di iptables normale.
  - La principale differenza è che non esiste la NAT
  - Una nota: Alcune funzionalità di ICMPv6 sono obbligatorie (Neighbor Discovery Protocol, type 135 e 136) e se le bloccate avrete problemi di connessione.

```
ip6tables -A INPUT -p ipv6-icmp -j ACCEPT
```

IPv6

- Breve parentesi: Linux supporta nativamente Ipv6 fin dai primi standard.
  - Le assegnazioni di blocchi IPv4 da parte della IANA sono state esaurite nel 2011.
  - In tutti i grandi datacenter internazionali IPv6 è già attivo e utilizzato.
  - In netfilter ipv6 è già supportato.
- Per chiarezza il tool userspace per gestire le regole relative al traffico IPv6 è chiamato "ip6tables".
  - Funziona esattamente allo stesso modo di iptables normale.
  - La principale differenza è che non esiste la NAT
  - Una nota: Alcune funzionalità di ICMPv6 sono obbligatorie (Neighbor Discovery Protocol, type 135 e 136) e se le bloccate avrete problemi di connessione.

```
ip6tables -A INPUT -p ipv6-icmp -j ACCEPT
```

IPv6

- Breve parentesi: Linux supporta nativamente Ipv6 fin dai primi standard.
  - Le assegnazioni di blocchi IPv4 da parte della IANA sono state esaurite nel 2011.
  - In tutti i grandi datacenter internazionali IPv6 è già attivo e utilizzato.
  - In netfilter ipv6 è già supportato.
- Per chiarezza il tool userspace per gestire le regole relative al traffico IPv6 è chiamato "ip6tables".
  - Funziona esattamente allo stesso modo di iptables normale.
  - La principale differenza è che non esiste la NAT
  - Una nota: Alcune funzionalità di ICMPv6 sono obbligatorie (Neighbor Discovery Protocol, type 135 e 136) e se le bloccate avrete problemi di connessione.

```
ip6tables -A INPUT -p ipv6-icmp -j ACCEPT
```

IPv6

- Breve parentesi: Linux supporta nativamente Ipv6 fin dai primi standard.
  - Le assegnazioni di blocchi IPv4 da parte della IANA sono state esaurite nel 2011.
  - In tutti i grandi datacenter internazionali IPv6 è già attivo e utilizzato.
  - In netfilter ipv6 è già supportato.
- Per chiarezza il tool userspace per gestire le regole relative al traffico IPv6 è chiamato "ip6tables".
  - Funziona esattamente allo stesso modo di iptables normale.
  - La principale differenza è che non esiste la NAT
  - Una nota: Alcune funzionalità di ICMPv6 sono obbligatorie (Neighbor Discovery Protocol, type 135 e 136) e se le bloccate avrete problemi di connessione.

```
ip6tables -A INPUT -p ipv6-icmp -j ACCEPT
```

Rendere le regole persistenti

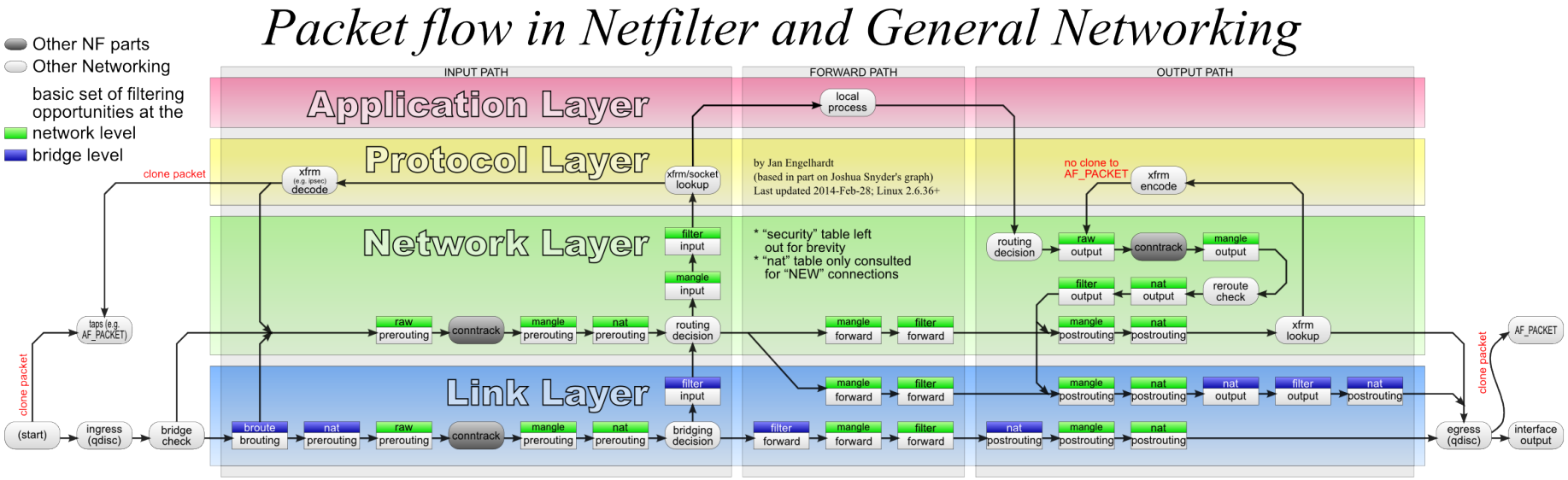
- Problema: Abbiamo tutte le nostre belle regole di iptables, però come facciamo a salvarle tra un riavvio e l'altro (al riavvio il sistema operativo cancella tutte le regole e ripristina tutte le chain vuote)
- iptables-save e iptables-restore fanno al caso nostro.

```
iptables-save > /etc/firewall_rules
iptables-restore < /etc/firewall_rules
```

- Possiamo aggiungere in /etc/rc.local o altri script di startup a seconda della distro.

Altri spunti

Abbiamo solo grattato la superficie di netfilter! ;)



Fonti

- Libro classico di Networking: Andrew S. Tanenbaum *"Reti di calcolatori"* IV edizione
- Miei appunti di Reti e Internet (non coprono tutti gli argomenti ma molto dettagliati): `http://otacon22.it/upload/reti.pdf`
- Materiale su iptables:
  - Un po' datato ma ben scritto: `http://www.netfilter.org/documentation/HOWTO/it/packet-filtering-HOWTO.html`
  - Nuovo ma un po' confusionario: `http://www.iptables.info/en/iptables-contents.html`
- O'Reilly - "Linux Networking cookbook" `http://shop.oreilly.com/product/9780596102487.do`:
  - È pieno di esempi di configurazione di vari programmi network-related, ma è un "cookbook". Non c'è una spiegazione dettagliata

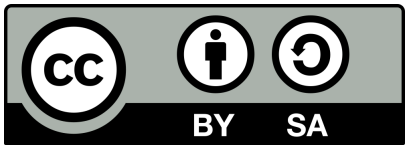
Fonti

- "Designing and Implementing Linux Firewalls and QoS using netfilter, iproute2, NAT and l7-filter":
  - Utile per un po' di argomenti ma leggermente datato
- Slides su firewalling su poul.org del 2012 `http://www.poul.org/wp-content/uploads/2012/05/presentazione_netfilter.pdf`
- "Linux Advanced Routing & Traffic Control" `http://www.lartc.org/` : Un po' di dettagli avanzati sul routing e Quality of Service (non chiarissimo ma è l'unico manuale che lo spiega).
- Le manpages sono spesso noiose da leggere ma sono ben fatte.

Mandatemi una mail se non trovate quello che cercate ;)

Ultima slide

Grazie per l'attenzione!



`http://www.poul.org`