

# Corsi Linux 2018

## Firewall

Federico Amedeo Izzo

[federico.izzo42@gmail.com](mailto:federico.izzo42@gmail.com)



**Queste slides le trovate su**

**[slides.poul.org/2018/corsi-linux-avanzati/Firewall](https://slides.poul.org/2018/corsi-linux-avanzati/Firewall)**

# Introduzione

# A cosa serve un firewall?

A controllare i pacchetti di rete che entrano, escono, transitano su un host.

# Ne abbiamo bisogno per

- Controllare tutte le connessioni di una macchina
- Limitare l'accesso ad una rete in un router di frontiera
- Sanitizzare i pacchetti in ingresso (integrity-check)
- Filtraggio *stateful* dei pacchetti
- (spoiler) NA[P]T

**OK, Dove lo compro?**

# Lo possiedi già

Tutte le distribuzioni GNU/Linux sono dotate dell'infrastruttura software per svolgere funzioni di firewall. Il software in questione è parte in *kernel space* per ragioni di performance e parte in *userspace* per comodità d'uso.

# Netfilter

Il kernel linux contiene **netfilter**, un framework per intercettare e manipolare pacchetti.

# Iptables

Per configurarlo useremo **iptables**, un programma CLI per comunicare al kernel le regole da applicare.

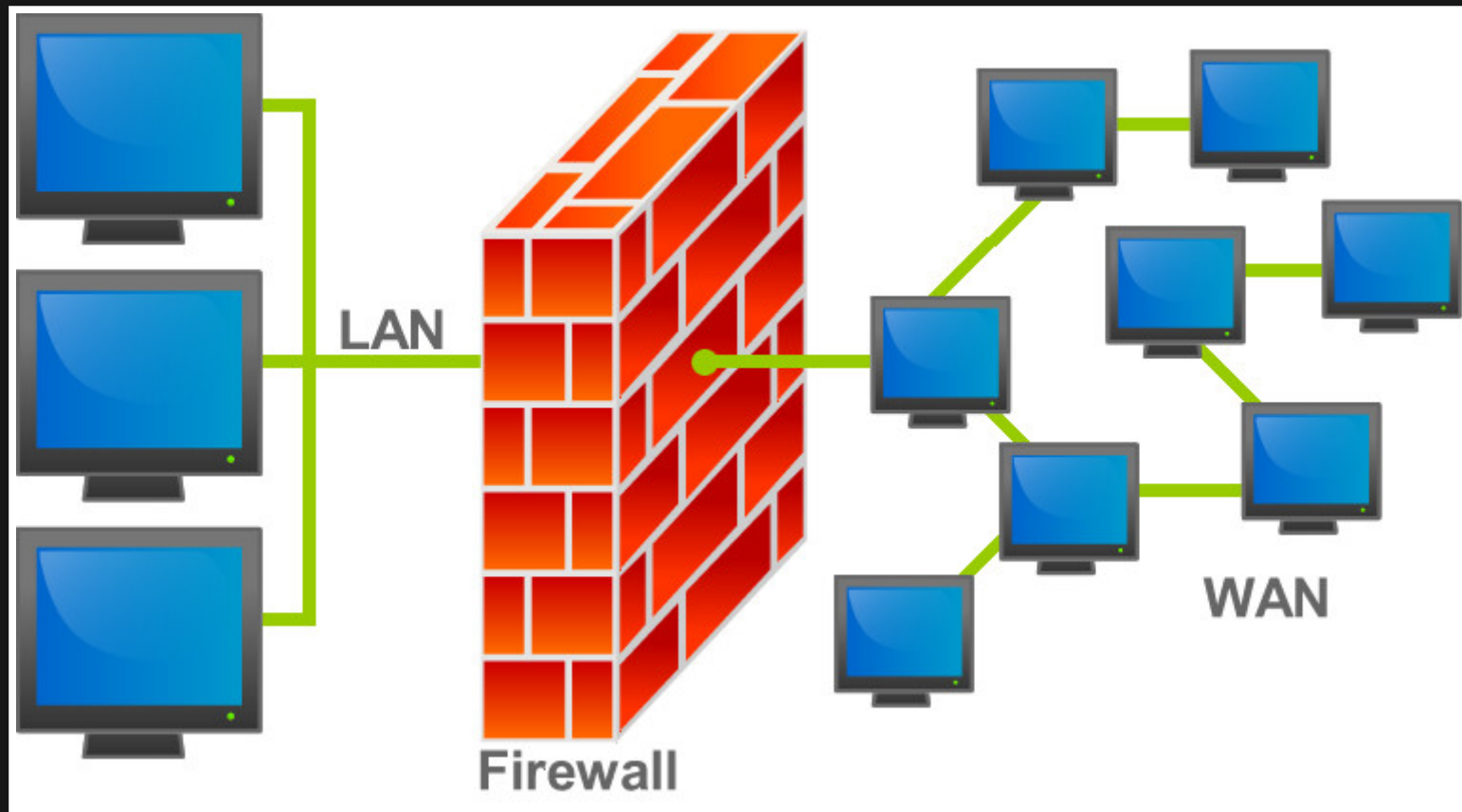
Esiste anche **ip6tables**, serve a specificare le regole per IPv6



**Ne voglio 5!**

# One firewall to rule them all

Il firewall deve essere l'unico punto di contatto tra la nostra rete e il mondo.

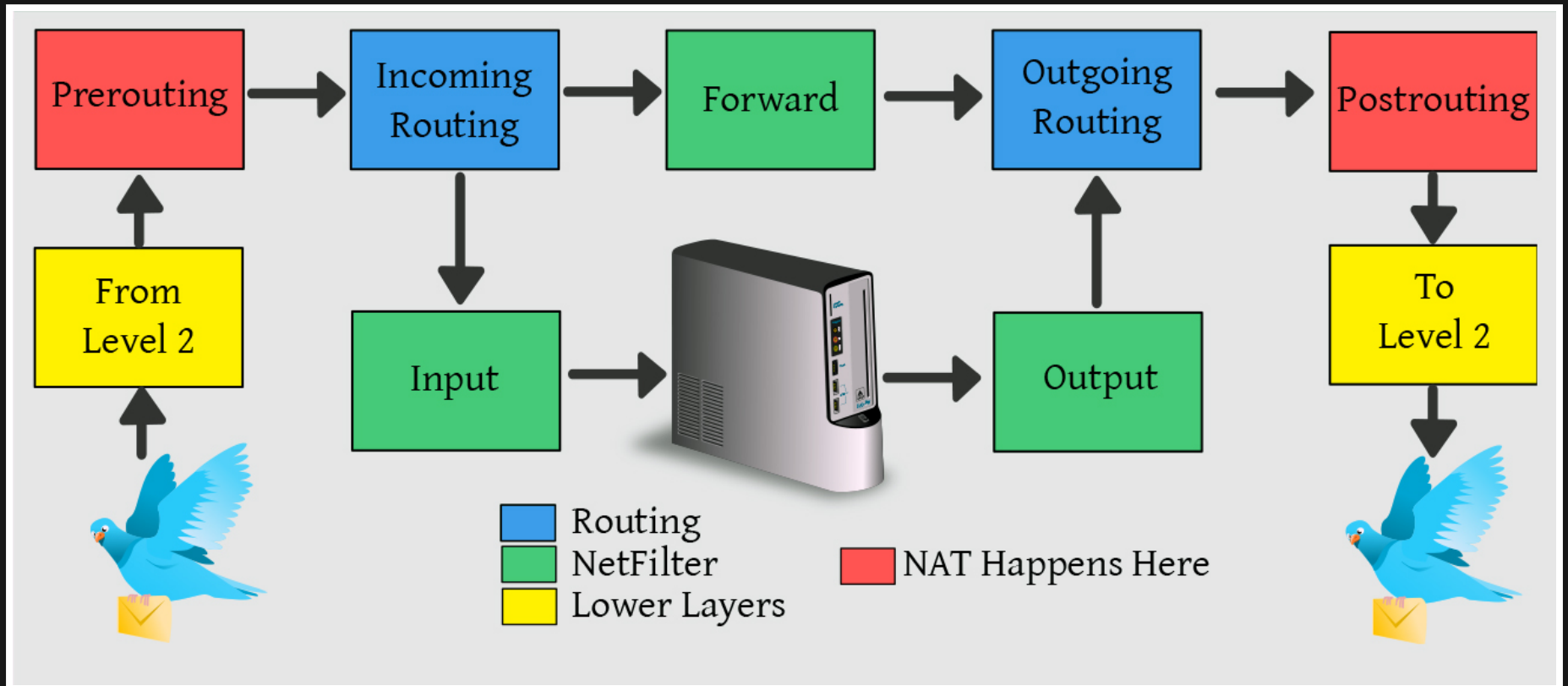


# Struttura

# Netfilter

- È composto da 5 *hooks* attraverso i quali i pacchetti in transito dovranno passare.
- Ogni *hook* impone delle regole a tutti i pacchetti che lo attraversano.

# Flusso dei pacchetti



Memorizzatelo, vi aiuterá dopo

# IP Forwarding

Permette al nostro server di comportarsi da router, lasciando passare i pacchetti non diretti alla nostra macchina.

Di default è disattivato nel kernel linux, per abilitarlo:

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

Per renderlo permanente modificare nel file `/etc/sysctl.conf`

```
net.ipv4.ip_forward = 1
```

# Netfilter tables

- Le regole sono divise in quattro **tabelle** in base al tipo di azione: filter, nat, mangle, raw
- In ogni tabella le regole sono organizzate in **chains**
- Le chain di base sono associate agli **hooks**

# Chains

- Le regole sono contenute in ordine di priorità
- Se non viene attivata nessuna regola viene applicata la **chain policy**

Per ispezionare la configurazione attuale:

```
# iptables -L [-t table]
```



# Configurazione

# Chain policy

Per impostare una **chain policy**

```
# iptables -P <chain> <policy>
```

Scegliendo tra:

- ACCEPT: il pacchetto viene lasciato passare
- QUEUE: il pacchetto viene inviato in userspace
- DROP: il pacchetto viene scartato, senza dire niente a nessuno

La policy di default è ACCEPT.

# Chain policy sane

- INPUT: DROP, decideremo noi chi potrà entrare
- FORWARD: DROP, per lo stesso motivo
- OUTPUT: ACCEPT, o DROP a seconda del livello di sicurezza desiderato
- PRE/POSTROUTING: ACCEPT, il filtraggio non va fatto qui

# Modificare le tabelle

```
# iptables [-t table] <action> <rule>
```

## Le azioni possibili sono:

- A <chain> Aggiunge la regola alla catena
- D <chain> Cancella la regola specificata (anche tramite numero)
- I <chain> <num> Inserisce la regola alla posizione specificata
- R <chain> <num> Rimpiazza la regola alla posizione specificata
- L Mostra tutte le regole della catena
- F Elimina tutte le regole (ma non la chain policy)

# Regole

Le regole sono composte da due parti:

- match
- target

Entrambi dichiarati tramite coppie parametro-valore.

Sintassi per specificare il target:

```
-j, --jump <target>
```

Specifica che operazione effettuare sul pacchetto.

# Target

Determinano il destino dei pacchetti che attivano una regola.

- ACCEPT/DROP: come le omonime **chain policy**
- REJECT: come DROP ma il mittente riceve una notifica
- LOG: il pacchetto viene tracciato nel log del kernel (syslog)
- MIRROR: inverte indirizzi sorgente e destinazione e lo rimanda indietro
- RATEEST: conta il pacchetto nel misuratore di traffico

# Match

# Match - I - Interface

Per prima cosa guardiamo da dove entrano.

Match sulle interfacce di rete

```
-i <iface> / -o <iface>
```

Applicabile solo su alcune chain:

- -i solo per INPUT FORWARD PREROUTING
- -o solo per OUTPUT FORWARD POSTROUTING



# MATCH - I - Interface

Attenti al **loopback!**

È un'interfaccia virtuale che molti programmi usano per comunicare con servizi presenti sulla stessa macchina.

```
iptables -A INPUT -i lo -j ACCEPT  
iptables -A OUTPUT -o lo -j ACCEPT
```

Dobbiamo accettare tutti i pacchetti in entrata e uscita da quest'interfaccia.

# MATCH - II - Address

Chi ti manda?

Match sugli indirizzi sorgente e destinazione

```
-s, --source address[/mask][, ...]  
-d, --destination address[/mask][, ...]
```

La netmask può essere specificata in due modi:

- Forma esplicita /a.b.c.d (176.31.102.216/255.255.255.192)
- Notazione VLSM /n (176.31.102.216/26)
- Implicita solo per /32 (172.31.102.216)
- Forma non contigua (255.255.255.249)

# MATCH - III - Proto

Protocollo di livello 4

```
-p [tcp|udp|udplite|icmp|icmpv6|esp|ah|sctp|mh|all]
```

**Meglio non filtrare icmp e icmpv6!**

- Destination Unreachable è importante
- ping, traceroute sono molto utili

# MATCH - IV - Ports

## Porte

```
--sports port[,port|,port:port] / --dports port[,port|,port:port]
```

È obbligatorio specificare il protocollo L4 (udp, tcp).

Utile per filtrare selettivamente i servizi:

- web (porta 80)
- ssh (porta 22)

# MATCH - V - Stateful

Conntrack memorizzerà per noi lo stato dei pacchetti  
Filtriamo per stato della connessione

```
-m conntrack --ctstate <state-list>
```

Lo stato del pacchetto può essere:

- NEW: se inizia una nuova connessione [SYN]
- ESTABLISHED: se appartiene ad un flusso di conntrack
- RELATED se inizia una nuova connessione associata (errori ICMP)
- INVALID se non è associato a nessuna connessione esistente
- UNTRACKED se il pacchetto viene escluso (-j CT -notrack)

# MATCH - VI - Extensions

Sono moduli aggiuntivi di cui possiamo fare uso:

```
-m limit --limit rate[/second|/minute|/hour|/day]
```

Può essere utile per limitare il logging.

```
-m recent [---name listname|---set|---rcheck|---update|---remove]
```

Per mantenere una lista di indirizzi, utile come banhammer.

- `-set`: aggiunge il source address alla lista
- `[!] -rcheck`: match solo se il source address è nella lista
- `[!] -update`: come check ma aggiorna il timestamp
- `[!] -remove`: se l'indirizzo è presente lo elimina

`man iptables-extensions`

# Consigli

# REJECT o DROP?

Meglio non inviare nessun REJECT:

- spreca banda inutilmente
- ci rende più vulnerabili ai DoS
- facilita il port scanning



# Blacklisting o whitelisting?

Qui il *blacklisting* non ha senso.

il *whitelisting* ci da più controllo.

# Configurazione standard

- INPUT e FORWARD impostati su DROP
- regole per fare whitelisting sul traffico legittimo
- OUTPUT su ACCEPT, quel traffico lo stiamo generando noi

# ICMP

È un protocollo di servizio utile per fare diagnostica o inviare informazioni di controllo sulla rete.

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

Meglio accettarli tutti, o altrimenti accettare comunque:

```
# iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT
```

i messaggi echo request.

```
# iptables -A INPUT -p icmp --icmp-type 3 -j ACCEPT
```

i messaggi destination unreachable.

```
# iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
```

e quelli echo reply.

# Esempio

```
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
```

Le policy di default devono essere impostate sempre per ultime, specialmente se si lavora via **ssh** su un server remoto.

**Ho riavviato e...**

# Salvataggio/Ripristino della configurazione

Ad ogni riavvio il firewall viene resettato.

È possibile salvare la configurazione attuale con

```
# iptables-save -c > /etc/iptables.rules
```

E caricare lo stato del firewall da un file di salvataggio

```
# iptables-restore < /etc/iptables.rules
```

# Salvataggio automatico

Possiamo fare tutto questo automagicamente.

Su Ubuntu/Debian c'è un pacchetto apposta

```
# sudo apt install iptables-persistent
```

Mentre su Arch Linux esiste un servizio di systemd

```
# touch /etc/iptables/iptables.rules  
# systemctl enable iptables
```

# Ma io sono per l'open data!

Per disabilitare tutte le funzioni del firewall:

Impostiamo le policy di default su ACCEPT.

```
# iptables -P INPUT ACCEPT  
# iptables -P OUTPUT ACCEPT  
# iptables -P FORWARD ACCEPT
```

Eliminiamo tutte le regole di tutte le catene

```
# iptables -F
```

Eliminiamo tutte le catene da noi definite

```
# iptables -X
```

iptables-apply ci permette di eseguirli atomicamente.

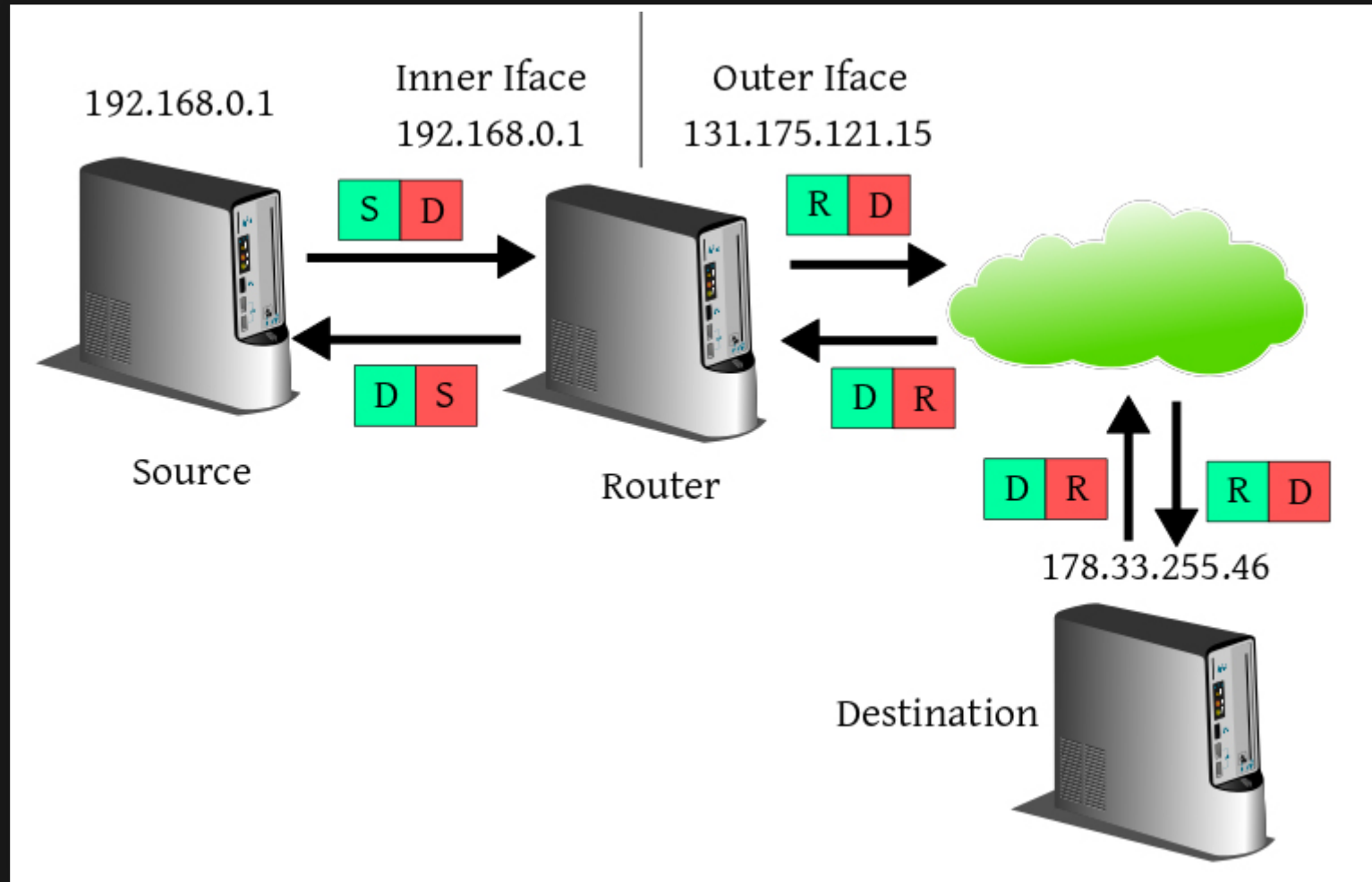


# Network Address Translation

# SNAT

- Serve quando si hanno a disposizione pochi indirizzi IP
- Piú host accedono ad una rete tramite lo stesso indirizzo.
- Spesso nelle LAN tutti i dispositivi accedono ad internet con un solo IP pubblico.
- Il posto migliore dove effettuarla è il router

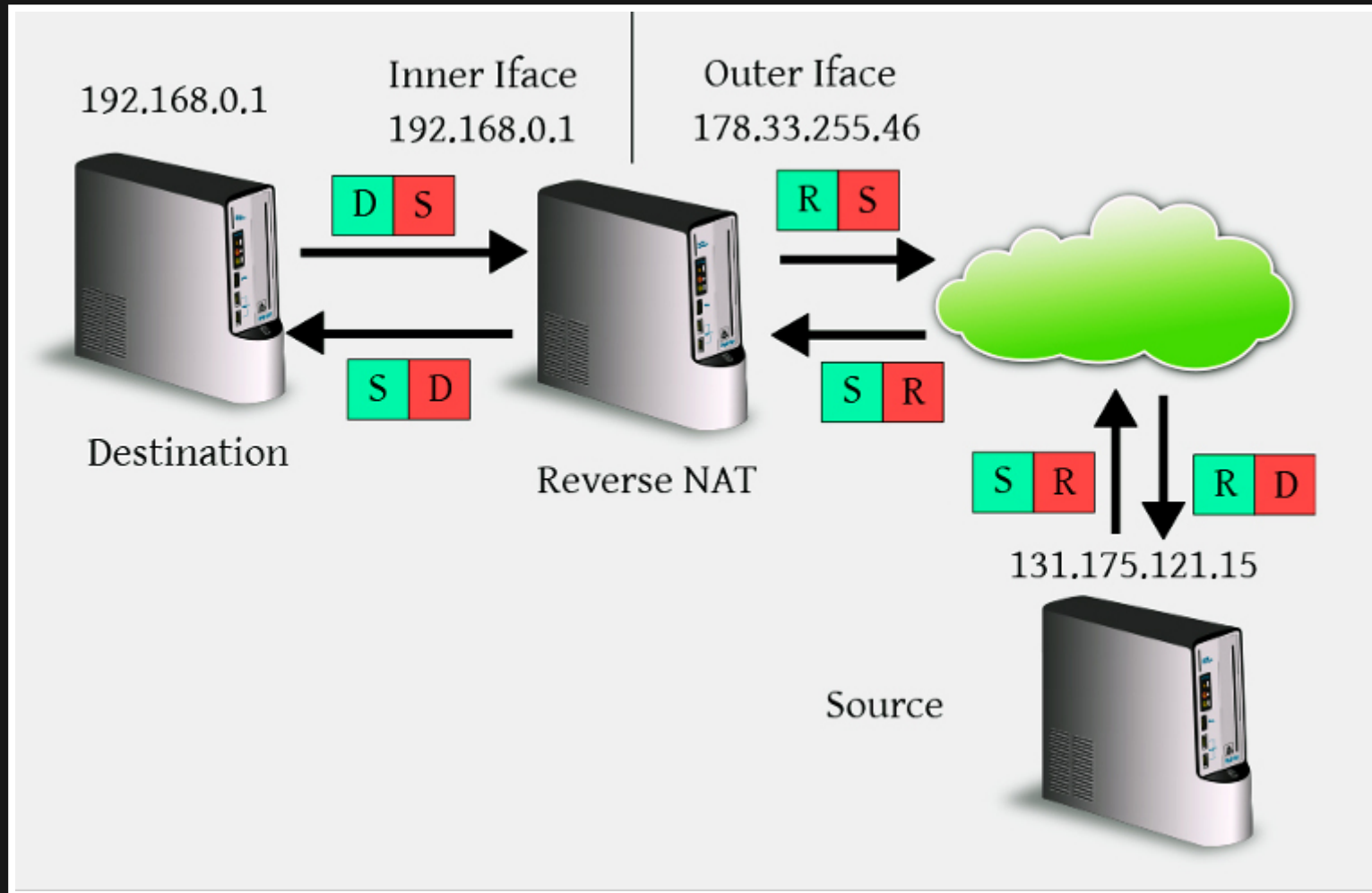
# SNAT - Struttura



# DNAT

- È il duale del Source NAT
- Utile per suddividere il carico di rete di un server
- Modifica dinamicamente l'indirizzo di destinazione
- Permette di sostituire un server senza compromettere il servizio

# DNAT - Struttura



# SNAT - Regole

SNAT viene effettuato nel POSTROUTING hook, prima che un pacchetto venga spedito. La traduzione corrispondente per il pacchetto in ritorno viene gestita automaticamente.

```
-t NAT -A POSTROUTING -j SNAT --to <address>
```

Questa regola permette di mascherare tutti i pacchetti che fanno match. Può essere utile l'opzione -o per specificare quale connessione mascherare.

```
-j MASQUERADE
```

È un target speciale che assegna come nuovo indirizzo sorgente quello dell'interfaccia di uscita. Utile nel caso di IP assegnati dinamicamente, ad esempio per connessioni DSL.

# DNAT - Regole

DNAT viene effettuato simmetricamente in PREROUTING, prima di qualunque altra elaborazione sul pacchetto. Anche qui viene garantita la bidirezionalità delle modifiche.

```
-t nat -A PREROUTING -j DNAT --to <address>
```

Questa regola indica l'indirizzo al quale il pacchetto deve essere inoltrato. Ovviamente qui non possiamo effettuare nessuna selezione automatica della destinazione.

# IPv6



# Nozioni base

- 128 bit di indirizzo invece di 32
- 8 gruppi da 4 caratteri hex  
2001:470:1f1c:005f:0000:0000:0000:0002
- Gli zeri possono essere omessi con **un solo** ::  
2001:470:1f1c:005f::2
- ICMPv6 sostituisce ARP nella *risoluzione livello-2*
- **Il NAT non serve piú** (e non va usato!)

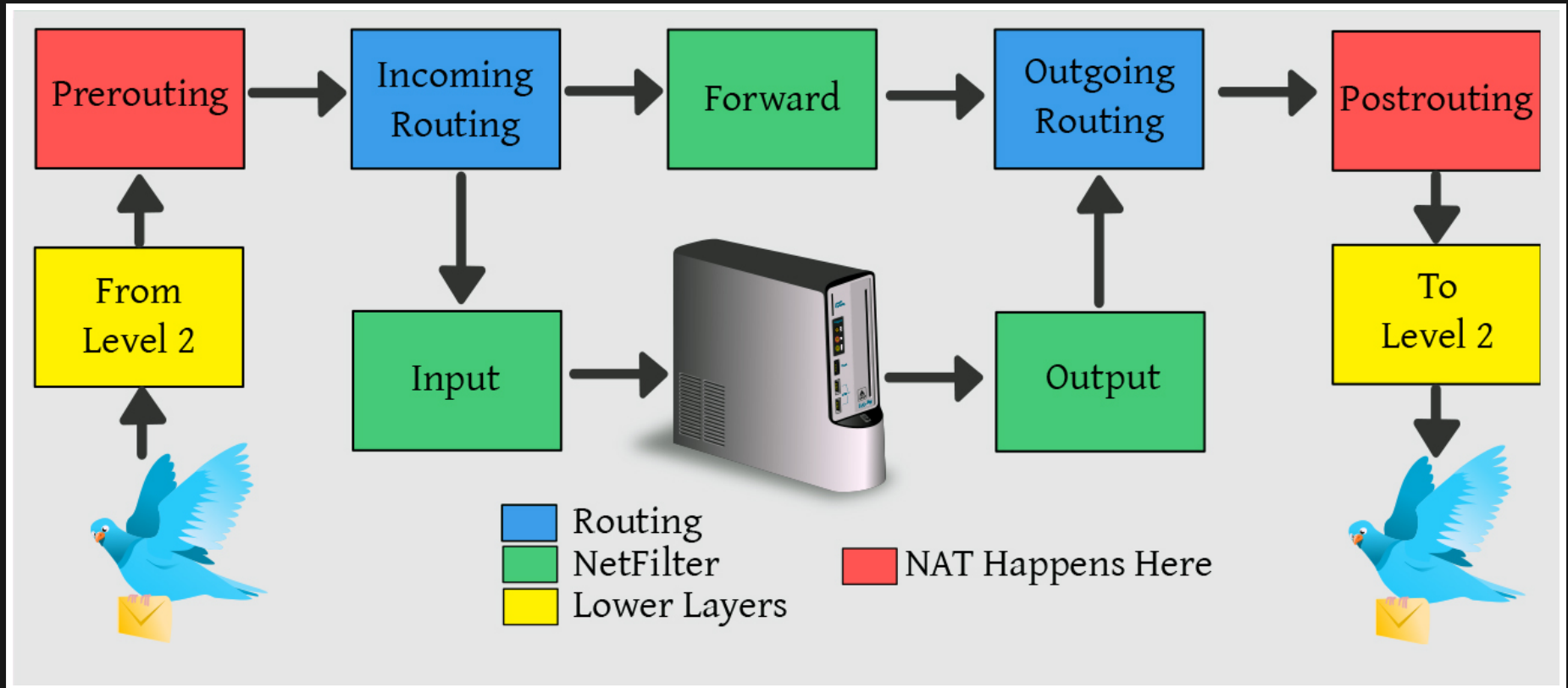
## Viene gestito da un firewall separato

Se la nostra connessione lo supporta  
dobbiamo configurare il firewall per IPv6,

Per evitare accessi indesiderati, port scan...

Il firewall si imposta con `ip6tables`

# Vi ricordate questo?



Non essendoci piú il NAT, le connessioni da internet alla LAN vanno filtrate in **FORWARD** e non piú in **INPUT**

# IPv6 best practices

- Mai filtrare **ICMPv6**
- Non serve **NAT**
- I pacchetti da internet vanno filtrati in **FORWARD**

# Esempio ip6tables

```
# Filter all packets that have RH0 headers (security measure)
ip6tables -A INPUT -m rt --rt-type 0 -j DROP
ip6tables -A FORWARD -m rt --rt-type 0 -j DROP
ip6tables -A OUTPUT -m rt --rt-type 0 -j DROP

# Filter packets to the router
ip6tables -A INPUT -i lo -j ACCEPT
ip6tables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
ip6tables -A INPUT -p icmpv6 -j ACCEPT
ip6tables -A INPUT -m conntrack --ctstate NEW -i br-lan -j ACCEPT
# Open SSH port for the router
ip6tables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW -j ACCEPT

# Filter packets to the LAN
ip6tables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
ip6tables -A FORWARD -p icmpv6 -j ACCEPT
```

Gli indirizzi privati **link local** iniziano con fe80::

# FINE

Queste slides sono distribuite sotto licenza:

**Creative Commons Attribution-ShareAlike 4.0 International.**

E sono state create con il framework **reveal.js**.

