# FIREWALL

## CORSI LINUX AVANZATI 2017

federico.izzo42@gmail.com

Federico Amedeo Izzo

# PER AVERE QUESTE SLIDES

Andate su:

slides.poul.org/2017/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 *kernelspace* 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.

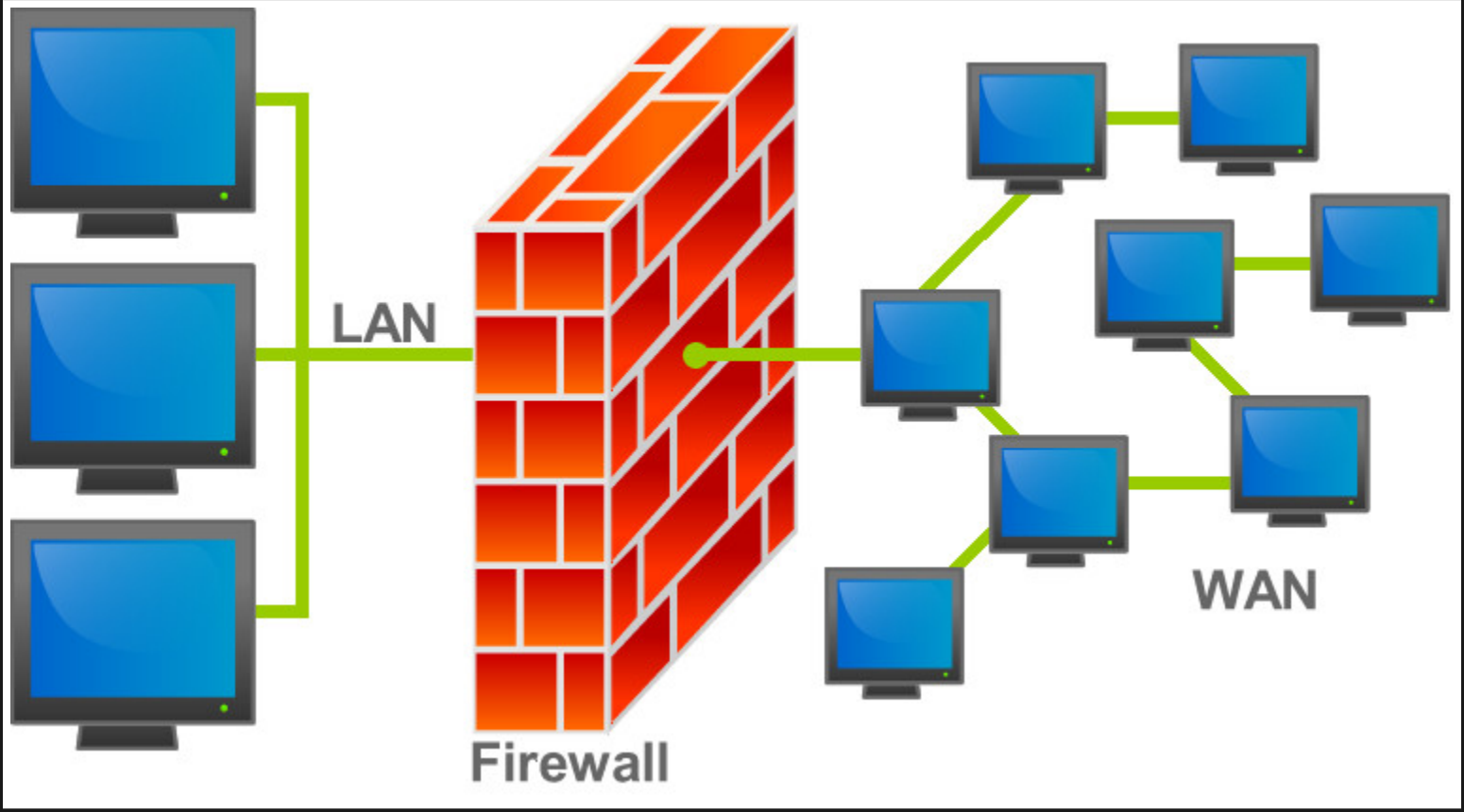
# IPTABLES

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.

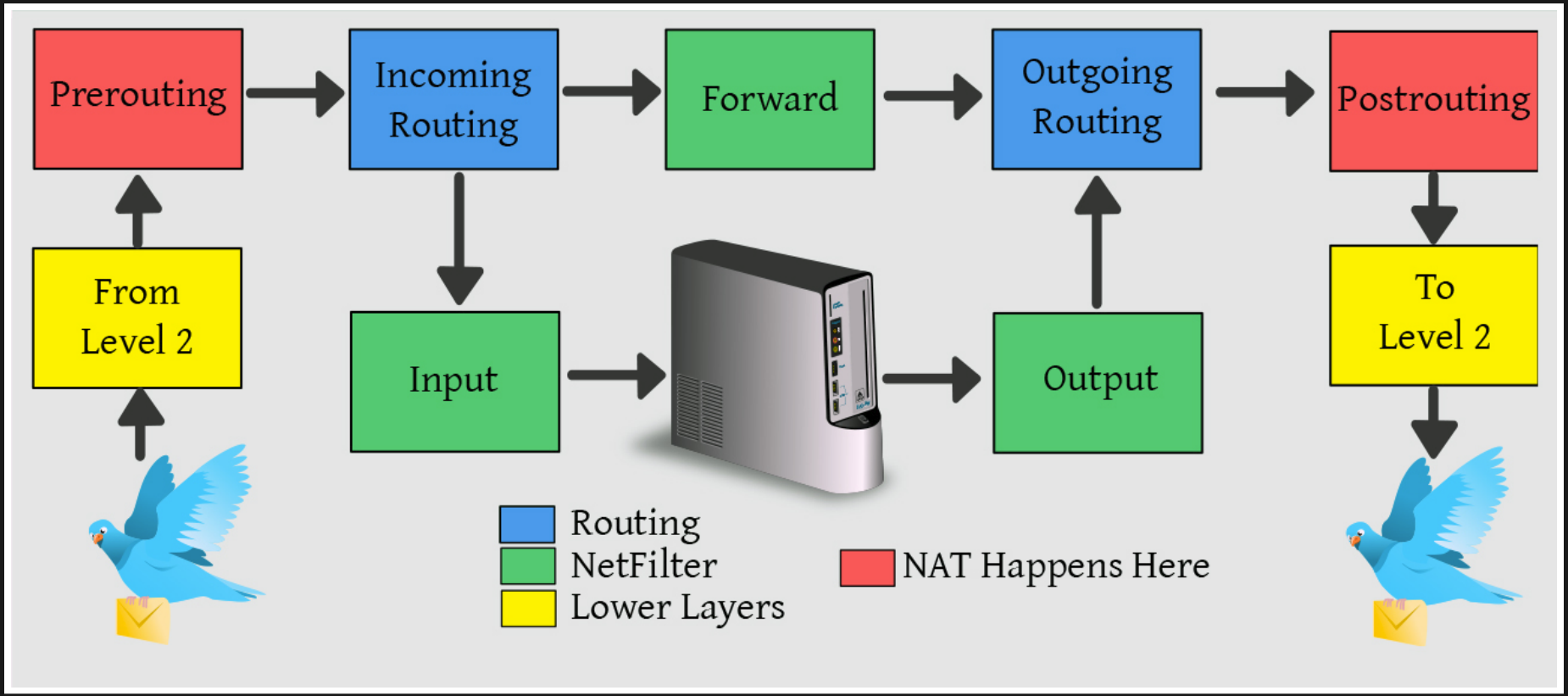


# STRUTTURA

# 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 ACCE
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 AC
iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW -j AC
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'é 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
```

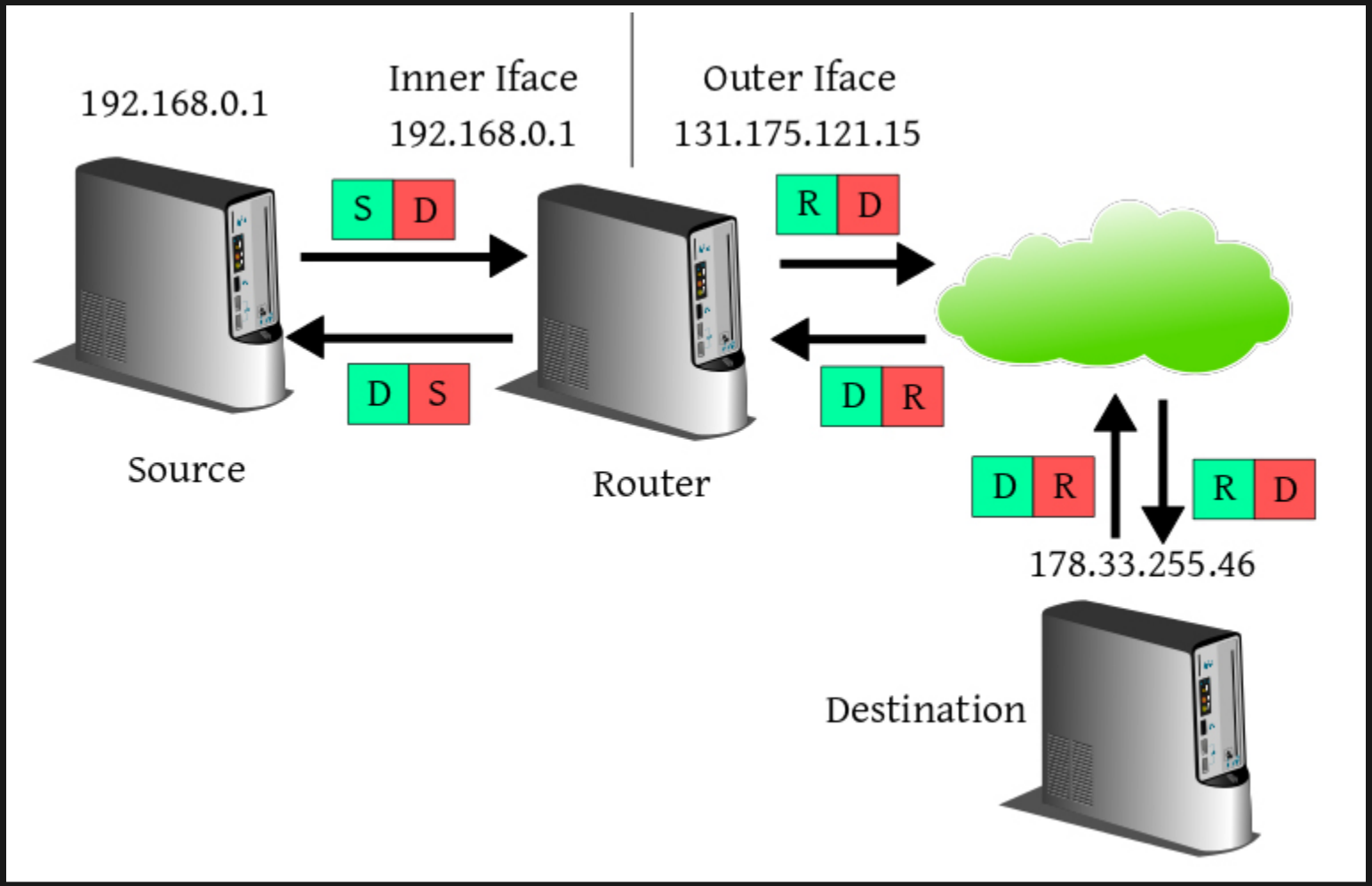
iptables-apply ci permette di eseguirli atomicamente.

# 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

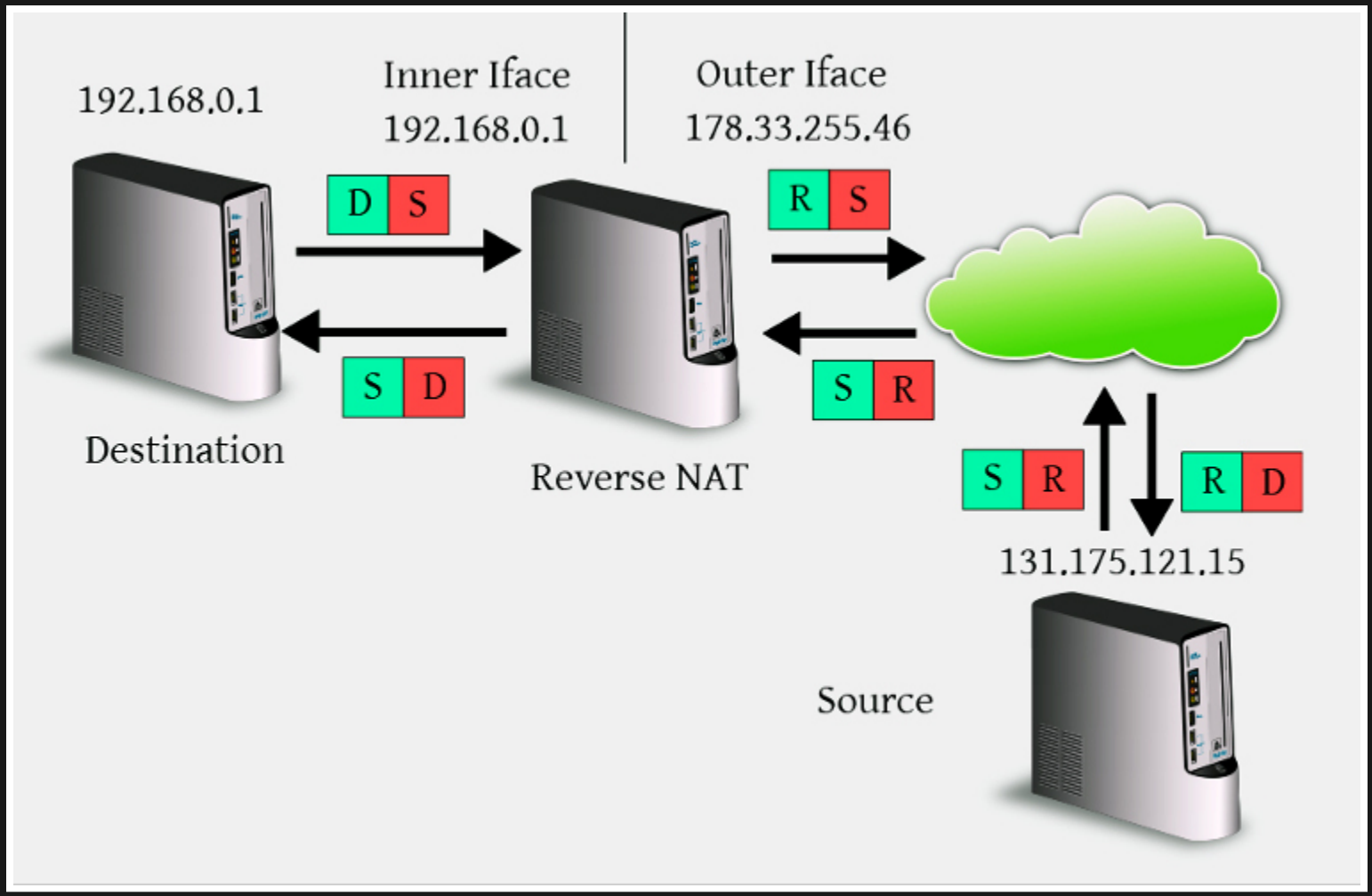
# SNAT - STRUTTURA



# 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.

# DOCKER

# DOCKER E IPTABLES

Docker modifica la configurazione di iptables per far comunicare i container

Il risultato non é esattamente un bello spettacolo

```
Chain INPUT (policy ACCEPT 330K packets, 22M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
 804K  308M DOCKER-ISOLATION  all  --  any    any     anywhere             anywhere
 3802   41M DOCKER     all  --  any    br-7a09bbf27b3c  anywhere             anywhere
 3802   41M ACCEPT     all  --  any    br-7a09bbf27b3c  anywhere             anywhere
 6196  424K ACCEPT     all  --  br-7a09bbf27b3c !br-7a09bbf27b3c  anywhere             anywhere
    0     0 ACCEPT     all  --  br-7a09bbf27b3c br-7a09bbf27b3c  anywhere             anywhere
    0     0 DOCKER     all  --  any    br-96d7da020dc0  anywhere             anywhere
    0     0 ACCEPT     all  --  any    br-96d7da020dc0  anywhere             anywhere
    0     0 ACCEPT     all  --  br-96d7da020dc0 !br-96d7da020dc0  anywhere             anywhere
    0     0 ACCEPT     all  --  br-96d7da020dc0 br-96d7da020dc0  anywhere             anywhere
 374K 3369M DOCKER     all  --  any    docker0  anywhere             anywhere
 344K 3367M ACCEPT     all  --  any    docker0  anywhere             anywhere             ctstate
 954K  144M ACCEPT     all  --  docker0 !docker0  anywhere             anywhere
    0     0 ACCEPT     all  --  docker0 docker0  anywhere             anywhere
 814K  454M DOCKER     all  --  any    br-7b9e2eed461b  anywhere             anywhere
 676K  439M ACCEPT     all  --  any    br-7b9e2eed461b  anywhere             anywhere
 866K  316M ACCEPT     all  --  br-7b9e2eed461b !br-7b9e2eed461b  anywhere             anywhere
16282 1399K ACCEPT     all  --  br-7b9e2eed461b br-7b9e2eed461b  anywhere             anywhere
54783   95M DOCKER     all  --  any    br-7a05e45cb2f4  anywhere             anywhere
54783   95M ACCEPT     all  --  any    br-7a05e45cb2f4  anywhere             anywhere
69958   18M ACCEPT     all  --  br-7a05e45cb2f4 !br-7a05e45cb2f4  anywhere             anywhere
    0     0 ACCEPT     all  --  br-7a05e45cb2f4 br-7a05e45cb2f4  anywhere             anywhere
    0     0 DOCKER     all  --  any    br-c9d31d6604e7  anywhere             anywhere
    0     0 ACCEPT     all  --  any    br-c9d31d6604e7  anywhere             anywhere
```

# PORTE

Se esponiamo una porta con

```
docker run --name nginx -d -p 9090:80 nginx
```

Viene creata una regola di questo tipo

```
Chain DOCKER (5 references)
target     prot opt source               destination
ACCEPT     tcp  --  0.0.0.0/0            172.18.0.2           tcp dpt:9090
```

Ora la porta 9090 é disponibile a tutto l'internet perché l'indirizzo sorgente é *anywhere*

Per esporre una porta solo in locale é possibile usare:

```
docker run --name nginx -d -p 127.0.0.1:9090:80 nginx
```

# SE ABBIAMO GIÁ CONFIGURATO IPTABLES

Le porte aperte con Docker ci passeranno attraverso.

**Docker, smettila di pasticciare con il mio iptables**

Possiamo dire a Docker di non modificare le regole di iptables aggiungendo un'opzione all'avvio del demone.

Per farlo creiamo il file /etc/systemd/system/docker.service.d/noiptables.conf

```
[Service]
ExecStart=
ExecStart=/usr/bin/docker daemon -H fd:// --iptables=false
```

# DOCKER RULES

Riavviando il server spariranno le regole create da Docker

Per farlo funzionare di nuovo aggiungiamo queste regole:

```
# iptables -A FORWARD -i docker0 -o eth0 -j ACCEPT
# iptables -A FORWARD -i eth0 -o docker0 -j ACCEPT
```

# FINE

Queste slides sono distribuite sotto licenza:

Creative Commons Attribution-ShareAlike 4.0 International.

E sono state create con il framework reveal.js.