Strumenti di packet filtering

1.1 IPTables/Netfilter

Linux fornisce un'infrastruttura di *packet filtering* a livello kernel

- Linux 2.2 `ipchains` solo stateless filtering
- Linux $\geq$2.4 `netfilter` (kernel mode) permette di scrivere regole stateful, organizzate in *catene* e *tabelle* da `iptables` (user mode)

Le funzionalità di filtraggio sono estendibili con moduli kernel. Potete vedere quali moduli sono installati esplorando le directory del kernel in uso. Nella macchina virtuale:

```
1 ls /lib/modules/*/kernel/net/netfilter
```

Per avere un'idea di cosa fa un certo modulo si usa `modinfo` (per informazioni piú dettagliate sarà necessario consultare la documentazione specifica, però)

```
1 modinfo xt_mac
```

Il kernel di Linux mantiene alcune tabelle di regole da esaminare nella manipolazione dei pacchetti di rete. Il numero e la natura delle tabelle dipende dalla configurazione del kernel. Ogni tabella contiene *catene* di regole: con `iptables` si possono alterare le regole e aggiungere/togliere catene.

filter Tabella di default. Catene di regole: INPUT, FORWARD, OUTPUT;

nat Tabella consultata quando si *crea* una nuova connessione. PREROUTING, OUTPUT, POSTROUTING

mangle Tabella usata in manipolazioni particolari. PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING

raw Tabella ad alta priorità. PREROUTING, OUTPUT

Per vedere il contenuto di filter:

```
1 iptables -t filter -L
```

Per seguire il destino di un pacchetto consideriamo la tabella filter, rappresentata schematicamente in Fig. 1.1:

1. Arriva un pacchetto (p.es. dalla scheda di rete)
2. Si decide dove deve essere consegnato (routing)
3. Se è locale, passa per la INPUT chain (ACCEPT/DROP)

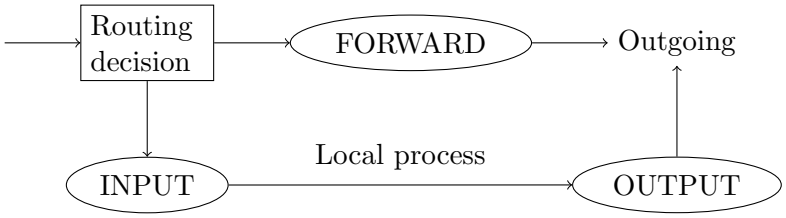

Figura 1.1: Tabella filter

4. Altrimenti DROP, ma se il forwarding è abilitato, si passa per la FORWARDING chain (ACCEPT/DROP)

5. Se un processo locale produce un pacchetto si passa per la OUTPUT chain (ACCEPT/DROP)

Il forwarding generalmente è disabilitato, a meno che non si tratti di un gateway con piú interfacce di rete.

Per scoprire se sulla macchina il forwarding è attivo

```
sysctl -a | grep forwarding
```

Nel caso lo si voglia attivare

```
sysctl -w net.ipv4.ip_forward=1
# equivalente a
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Questi parametri possono essere cambiati a run-time con `sysctl`, oppure resi permanenti (nel senso che vengono fissati al boot) mettendoli nel file `/etc/sysctl.conf`. A proposito: nel file `/etc/sysctl.conf` sono elencati altri parametri di configurazione del kernel che possono venire utili per difendere un nodo di una rete

```
# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1
```

Tornando a `iptables`, il grafo di flusso completo è schematizzato dalla Fig. 1.2.

La *routing decision* determina se il pacchetto ha destinazione locale o esterna: deve essere ripetuta piú volte perché il filtro potrebbe alterare i campi rilevanti del pacchetto.

1.1.1 Esercizi

- Leggere il manuale di `iptables` e `iptables-extensions`

- Interpretare l'effetto dei seguenti comandi

```
iptables −t filter −F # -t filter (default)
iptables −t filter −X
iptables −A INPUT −−source 10.49.165.18 −p tcp −−dport 22 −j ACCEPT
iptables −A INPUT −p tcp −−dport 22 −j DROP
```

- Come si fa a salvare e ripristinare un insieme di regole?

*S*oluzione: prima cancella tutte le regole esistenti e poi blocca tutti gli accessi alla porta 22, meno quelli provenienti da 10.49.165.18. L'ultimo comando serve ad assicurare che il default sia DROP.

1.2 Esperimenti con Mininet e iptables

Per i primi esperimenti è sufficiente la semplice topologia di default, con due host `h1` e `h2` attaccati al medesimo switch `s1`.

```
sudo mn
# mininet¿ xterm h1 h2
```

Verifichiamo che i due host comunichino quando non vi sono regole di filtraggio:

```
# su h1
ip addr # h1-eth0 è 10.0.0.1/8
iptables −L −t filter
nc −l −p 12345
# su h2
echo ciao | nc 10.0.0.1 12345
```

Aggiungiamo una regola che scarta i pacchetti con provenienza 10.0.0.2 sulla porta tcp 12345.

```
# su h1
iptables −t filter −F
iptables −t filter −A INPUT −p tcp −−source 10.0.0.2 −−dport 12345 −j DROP
iptables −t filter −L
nc −l −p 12345
# su h2
ip addr # h2-eth0 è 10.0.0.2/8
echo ciao | nc 10.0.0.1 12345
ip addr del 10.0.0.2/8 dev h2−eth0
ip addr add 10.0.0.22/8 dev h2−eth0
echo ciao | nc 10.0.0.1 12345
```

1.2.1 Esercizi

- Che succede se invece di `DROP` si usa `REJECT`? Perché?
- Per errore avete dato un indirizzo con netmask a 0 bit (ip addr add 10.0.0.22 dev h2−eth0. Che succede? Perché?

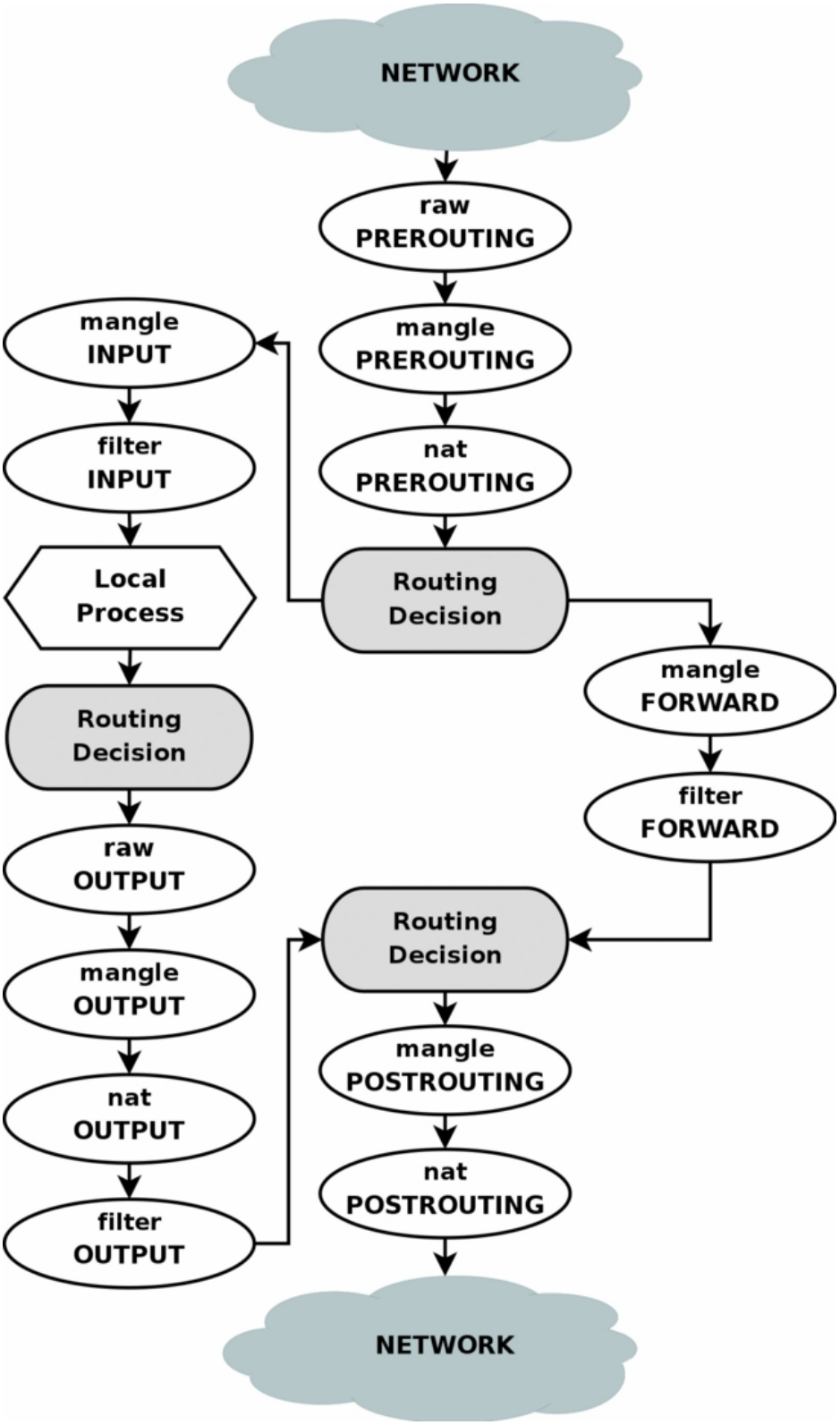

Figura 1.2: Flusso di un pacchetto fra le tabelle di `iptables`