Linux 2.4 Packet Filtering HOWTO

Rusty Russell, mailing list `netfilter@lists.samba.org` v1.0.1 Lunedì 1 Maggio 18:09:31 CST 2000

Questo documento descrive come usare iptables per filtrare i pacchetti con il kernel 2.4. Traduzione a cura di Masetti Marco *marcomas@libero.it* `mailto:marcomas@libero.it`

Indice

1 Introduzione

Benvenuto, caro lettore.

Si presuppone che i termini indirizzo IP, indirizzo di rete, maschera di rete, instradamento e DNS siano noti, altrimenti raccomando di leggere il Network Concepts HOWTO.

Questo HOWTO svaria da un'amichevole introduzione (che ti farà subito sentire al caldo e al sicuro, ma senza difese dal mondo reale) a crude rivelazioni (che lasceranno forse anche le anime più forti confuse, paranoiche e alla ricerca di armi pesanti).

La tua rete non è **sicura**. Il problema di permettere comunicazioni rapide e convenienti pur restringendone il loro uso per garantire buoni e non diabolici intenti è congruente ad un altro intrattabile problema, ossia permettere alle persone di parlare liberamente in un teatro affollato ma non consentire una chiamata Al fuoco!. Ciò non sarà risolto nello spazio di questo HOWTO.

Solo tu puoi decidere dove sarà il compromesso. Nel frattempo cercherò di istruirti sull'uso di alcuni tool disponibili e ad evitare alcune vulnerabilità, nella speranza che tu ne faccia buon uso, per scopi buoni e non diabolici. Un altro problema equivalente.

2 Dov'è il sito ufficiale? Esiste una Mailing List?

Ci sono tre siti ufficiali:

- Grazie a *Penguin Computing* `http://netfilter.filewatcher.org/` .
- Grazie a *The Samba Team e SGI* `http://netfilter.samba.org/` .
- Grazie a *Harald Welte* `http://netfilter.gnumonks.org/` .

Per la mailing list ufficiale di netfilter visita l'indirizzo

Netfilter List `http://www.netfilter.org/contact.html#list` .

3 Cos'è un filtro dei pacchetti ?

Un filtro dei pacchetti (packet filter) è un pezzo di software che guarda le *intestazioni* dei pacchetti, e ne decide il destino. Potrebbe decidere di scartare (**DROP**) il pacchetto (ossia eliminarlo come se non fosse mai stato ricevuto), accettarlo (**ACCEPT**) (ossia lasciare che il pacchetto prosegua), o di fare qualcos'altro di più complicato.

In Linux, il filtro dei pacchetti è contenuto nel kernel (come modulo o come parte integrante) e ci sono diverse cose delicate che si possono fare, ad ogni modo il principio generale è sempre quello di guardare l'intestazione di un pacchetto e di deciderne il destino.

3.1 Perché dovrei filtrare i pacchetti?

Controllo. Sicurezza. Vigilanza.

Controllo:

quando si utilizza una Linux box per connettere la propria rete interna ad un'altra (diciamo Internet) si ha l'opportunità di consentire un certo tipo di traffico, e di vietarne dell'altro. Per esempio, l'intestazione dei pacchetti contiene l'indirizzo di destinazione, quindi si può impedire che certi pacchetti arrivino ad una certa parte della rete esterna. Un altro esempio: uso Netscape per accedere agli archivi di Dilbert, nella pagina ci sono avvisi pubblicitari di doubleclick.net, e Netscape spreca allegramente tempo a scaricarli. Per risolvere il problema è sufficiente indicare al filtro dei pacchetti di non accettare alcun pacchetto diretto o proveniente dall'indirizzo di doubleclick.net (sebbene ci siano metodi migliori: vedi Junkbuster).

Sicurezza:

quando la tua Linux box è la sola cosa tra il caos di Internet e la propria bella ed ordinata rete, è bene sapere che si può limitare quel che viene a bussare alla propria porta. Per esempio, si potrebbe permettere a qualsiasi cosa di uscire dalla propria rete, ma certamente non si è ben disposti verso i cosiddetti Ping della Morte provenienti da chissà quale posto malvagio. Altro esempio: potresti voler impedire che estranei effettuino dei telnet alla tua Linux box, anche se tutti i tuoi account sono dotati di password. Potresti voler ancora (come molte altre persone) essere solo un osservatore di Internet, e non un server (volontario o no): semplicemente non si lasci a nessuno la possibilità di connettersi, a questo scopo basta impostare il filtro dei pacchetti in modo che rifiuti i pacchetti usati per richiedere una connessione.

Vigilanza:

qualche volta una macchina della rete locale configurata male, può decidere di sparare pacchetti al mondo esterno. E' buona cosa chiedere al filtro dei pacchetti che ti faccia notare se qualcosa di anormale sta accadendo; forse si potrebbe fare qualcosa per risolvere il problema o forse si potrebbe essere solo curiosi di natura.

3.2 Come filtrare i pacchetti sotto Linux?

I kernel Linux hanno il filtraggio dei pacchetti sin dalla versione 1.1. La prima generazione, basata su ipfw della BSD, è stata introdotta da Alan Cox verso la fine del 1994. Successivamente è stata migliorata da Jos Vos ed altri per Linux 2.0; il tool 'ipfwadm' controllava le regole di filtraggio del kernel. Nella metà del 1998, per Linux 2.2, ho rielaborato il kernel piuttosto pesantemente, con l'aiuto di Michael Neuling, e ho introdotto il tool 'ipchains'. Quindi finalmente nella metà del 1999 sono arrivati il tool di quarta generazione 'iptables' e un'altra riscrittura del kernel, ed è proprio su iptables che questo HOWTO si concentra.

E' necessario un kernel contenente l'infrastruttura netfilter: netfilter è un framework generico, presente nel kernel, dove possono essere inserite varie cose (ad esempio il modulo di iptables). Ciò richiede almeno un kernel 2.3.15 o più recente, e di rispondere 'Y' durante la configurazione del kernel alla voce CONFIG_NETFILTER.

Il tool `iptables` dialoga con il kernel e gli indica quali pacchetti filtrare. Se non sei un programmatore, o un curioso, allora è attraverso questo tool che controllerai il filtraggio dei pacchetti.

3.2.1 iptables

Il tool `iptables` inserisce e rimuove le regole dalla tabella di filtraggio del kernel. Questo significa che tutte le regole impostate andranno perse al reboot; leggi la sezione 3.2.2 (Rendere le regole permanenti) per capire come assicurarsi che le regole siano reimpostate al successivo avvio di Linux.

`iptables` è il sostituto di `ipfwadm` e di `ipchains`: leggi la sezione 8 (Usare ipchains e ipfwadm) per imparare come usare iptables senza troppi sforzi, se abituati ad usare uno di questi tool.

3.2.2 Rendere le regole permanenti

Le tue impostazioni correnti per il firewall sono memorizzare nel kernel, e quindi andranno perse al reboot. Scrivere iptables-save e iptables-restore è nella mia lista delle cose da fare. Quando saranno pronte, saranno eccezionali, promesso.

Nel frattempo colloca i comandi che impostano le regole in uno script di inizializzazione. Assicurati che sia effettuato qualcosa di intelligente se uno dei comandi dovesse fallire (di solito 'exec /sbin/sulogin').

4 Chi diavolo sei, e perché stai giocando con il mio kernel?

Sono Rusty, il manutentore del firewall IP di Linux, nonché ennesimo coder che si è trovato nel posto giusto al momento giusto.

Ho scritto ipchains (vedi sopra 3.2 (Come filtrare i pacchetti sotto Linux?) per saperne di più sulle persone che hanno partecipato al progetto), e ho imparato abbastanza per realizzare nel modo corretto il filtraggio dei pacchetti. O almeno spero.

WatchGuard `http://www.watchguard.com` , un'eccellente compagnia che si occupa di firewall e che commercializza l'ottimo plug-in FireBox, si è offerta di pagarmi per fare praticamente nulla. Ho potuto impiegare tutto il tempo per scrivere questo materiale e mantenere il materiale precedente. Avevo predetto 6 mesi, ne ho impiegati invece 12, ma sento che alla fine tutto è stato fatto per bene. Molte riscritture, un hard-disk rotto, un portatile è stato rubato, un paio di filesystem corrotti e più tardi uno schermo distrutto, questo è quanto.

Mentre sono qui, voglio chiarire alcuni equivoci: io non sono un guru del kernel, lo riconosco, il mio lavoro sul kernel mi ha solo portato in contatto con alcuni di essi: David S. Miller, Alexey Kuznetsov, Andi Kleen, Alan Cox. Sono tutti molto occupati a curare gli aspetti più tecnici, e ciò mi hanno permesso di camminare dove si tocca, dove è sicuro.

5 La guida rapida di Rusty al filtraggio dei pacchetti

La maggior parte delle persone hanno una singola connessione PPP a Internet, e quello che vogliono è che nessuno possa penetrare nella loro rete, o nel firewall:

```
## Inserisci i moduli del connection-tracking (non necessari se inclusi nel kernel)
# insmod ip_conntrack
# insmod ip_conntrack_ftp

## Crea una catena che blocchi le nuove connessioni, eccetto quelle provenienti dall'interno.
# iptables -N block
# iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A block -m state --state NEW -i ! ppp0 -j ACCEPT
# iptables -A block -j DROP

## Dalle catene INPUT e FORWARD salta a questa catena
# iptables -A INPUT -j block
# iptables -A FORWARD -j block
```

6 Come i pacchetti attraversano i filtri

Il kernel ha tre liste di regole presenti nella tabella 'filter'; queste liste sono chiamate **catene del firewall** (firewall chains) o giusto **catene** (chains). I loro nomi sono: **INPUT**, **OUTPUT** e **FORWARD**.

Tutto ciò ora è molto differente rispetto al funzionamento dei kernel 2.0 e 2.2!

Per i fan dell'arte ASCII, le catene adesso sono organizzate così:

```
                          -----
Ingresso                 /     \      Uscita
       -->[Decisioni]-->|FORWARD|------>
          [di routing]   \_____/        ^
               |                        |
               v                       ----
              ---                     /    \
             /   \                   |OUTPUT|
            |INPUT|                   \____/
             \___/                      ^
               |                        |
                ----> Processi locali --
```

I tre cerchi rappresentano le tre catene citate sopra. Quando un pacchetto arriva ad un cerchio del diagramma, questa catena lo esamina per deciderne il destino. Se la catena dice di scartare (DROP) il pacchetto, questo sarà ucciso lì, se la catena dice di accettarlo (ACCEPT), il pacchetto allora potrà continuare a percorrere il diagramma.

Una catena è una lista di **regole**. Ogni regola dice 'se l'intestazione del pacchetto appare in questo modo, allora ecco cosa bisogna fare'. Se il pacchetto non soddisfa una certa regola, allora sarà consultata la regola successiva. Se non ci sono più regole da consultare allora il kernel, per decidere cosa fare, consulta la **tattica** (policy) della catena. In un sistema per la sicurezza, questa tattica in genere indica al kernel di scartare (DROP) il pacchetto.

1. Quando un pacchetto arriva (o meglio attraversa la scheda ethernet) il kernel prima di tutto osserva qual è la destinazione del pacchetto, questa fase è detta 'routing' (instradamento).

2. Se è destinato a questa box allora il pacchetto nel diagramma, prosegue in basso, verso la catena INPUT. Se la catena lo lascia passare i processi in attesa di questo pacchetto lo riceveranno.

3. Altrimenti, se non è stato attivato il forwarding nel kernel, o se il kernel non conosce come farlo proseguire, il pacchetto sarà scartato. Se invece il forwarding è attivato, e il pacchetto è destinato ad un'altra interfaccia di rete (se ne hai un'altra), allora il pacchetto nel nostro diagramma prosegue a destra verso la catena FORWARD. Se sarà accettato allora sarà inviato nella rete.

4. Infine anche un programma sulla box può inviare dei pacchetti nella rete. Questi pacchetti passano immediatamente attraverso la catena OUTPUT, e se questa li accetta allora proseguono verso l'interfaccia a cui sono destinati.

7 Usare iptables

Se hai bisogno di maggiori dettagli, iptables ha un completo manuale (`man iptables`). Tutti coloro che hanno familiarità con ipchains potrebbero aver bisogno di dare giusto un'occhiata a 10 (Differenze tra iptables e ipchains), anticipo che sono comunque molto simili.

Ci sono molte cose differenti che si possono fare con `iptables`. Si parte con tre catene preesistenti `INPUT`, `OUTPUT` e `FORWARD` che non si possono cancellare. Diamo un'occhiata alle operazioni utili per gestire intere catene:

1. Crea una nuova catena (-N).
2. Cancella una catena vuota (-X).
3. Cambia la politica di una delle catene preesistenti. (-P).
4. Elenca le regole presenti in una catena (-L).
5. Svuota una catena delle sue regole (-F).
6. Azzera i contatori dei pacchetti e dei byte di tutte le regole di una catena (-Z).

Ci sono poi diversi modi per manipolare le regole di una catena:

1. Appendi una nuova regola alla catena (-A).
2. Inserisci una nuova regola in una determinata posizione della catena (-I).
3. Sostituisci una regola presente in una certa posizione della catena (-R).
4. Cancella una regola presente in una certa posizione della catena (-D).
5. Cancella la prima regola di una catena (-D).

7.1 Cosa vedrai quando avvierai il computer

iptables potrebbe essere un modulo, chiamato ('iptable_filter.o'), in questo caso dovrebbe essere caricato automaticamente la prima volta che eseguirai il comando `iptables`, oppure potrebbe essere incluso nel kernel.

Prima che qualsiasi comando iptables sia eseguito (presta attenzione, alcune distribuzioni eseguono iptables nei loro script di inizializzazione), nelle catene predefinite ('INPUT', 'FORWARD' e 'OUTPUT') non sono presenti regole, e tutte hanno una tattica di ACCEPT. Puoi tuttavia modificare la tattica di default della catena FORWARD impostando l'opzione 'forward=0' nel modulo iptable_filter.

7.2 Operazioni su una singola regola

Questa è il pane e burro del filtraggio dei pacchetti: manipolare le regole. Nella maggior parte dei casi probabilmente utilizzerai i comandi appendi (-A) e cancella (-D), gli altri comandi (-I inserisci e -R sostituisci) sono delle semplici estensioni di questi concetti.

Ogni regola specifica un insieme di condizioni che il pacchetto deve soddisfare, e che cosa fare se le soddisfa (obiettivo). Per esempio, potresti voler rifiutare tutti i pacchetti ICMP provenienti dall'indirizzo IP 127.0.0.1. In questo caso le tue condizioni dovrebbero essere che il protocollo sia ICMP e che l'indirizzo sorgente sia 127.0.0.1, mentre l'obiettivo dovrebbe essere impostato a 'DROP' (scarta).

127.0.0.1 è l'interfaccia di 'loopback' che hai sicuramente anche senza possedere alcuna connessione reale ad una rete. Puoi usare il programma 'ping' per generare questo tipo di pacchetti (non fa altro che inviare un ICMP di tipo 8 (echo request), tutti gli host raggiunti dovrebbero rispondere con un pacchetto ICMP tipo 0 (echo reply)). Questo è molto utile per fare dei test.

```
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.2 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms
# iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
#
```

Come puoi notare il primo ping ha avuto successo (il '-c 1' serve ad indicare al ping di inviare un solo pacchetto).

Ora appendiamo (-A) alla catena 'INPUT', una regola che indichi che i pacchetti provenienti dall'indirizzo 127.0.0.1 ('-s 127.0.0.1') con protocollo ICMP ('-p icmp') devono essere scartati ('-j DROP').

Proviamo quindi la nostra regola usando un secondo ping. Dovrebbe esserci una pausa prima che il programma, in attesa di una risposta che non arriverà mai, si arrenda e abbandoni.

Possiamo cancellare la regola in uno, due modi. Innanzi tutto se sappiamo che la catena input ha solo una regola, possiamo usare una 'cancellazione numerata', ossia

```
# iptables -D INPUT 1
#
```

Questo comando rimuove la prima regola della catena INPUT.

Il secondo modo consiste nel ripetere la linea del comando -A, ma sostituendo -A con -D. Molto utile quando hai una catena complessa di regole e non vuoi contarle ad una ad una per scoprire che la regola di cui vuoi sbarazzarti è la 37esima. In questo caso usa:

```
# iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP
#
```

La sintassi di -D deve avere le stesse opzioni dei comandi -A (o -I o -R). Se ci sono regole multiple e identiche nella stessa catena, solo la prima sarà cancellata.

7.3 Specificare cosa filtrare

Abbiamo già visto l'uso dell'opzione '-p' per specificare il protocollo, e di '-s' per specificare l'indirizzo sorgente, esistono però anche altre opzioni per indicare altre caratteristiche dei pacchetti. Ciò che segue è un compendio piuttosto completo.

7.3.1 Specificare gli indirizzi IP sorgente e destinazione

Gli indirizzi IP sorgente ('-s', '–source' o '–src') e destinazione ('-d','–destination' o '–dst') possono essere specificati in 4 modi diversi. Il più comune è di usare il nome completo, ad esempio 'localhost' o 'www.linuxhq.com'. Il secondo modo è quello di specificare l'indirizzo IP ad esempio '127.0.0.1'.

La terza e la quarta modalità permettono di specificare un gruppo di indirizzi IP, ad esempio '199.95.207.0/24' oppure '199.95.207.0/255.255.255.0'. Entrambe specificano qualsiasi indirizzo IP a partire da 199.95.207.0 fino a 199.95.207.255. Di default viene usato '/32' o '/255.255.255.255' (tutti gli indirizzi IP). Si può anche usare '/0' per specificare nessun indirizzo IP, come in questo caso:

```
[ NOTA: '-s 0/0' qui è ridondante. ]
# iptables -A INPUT -s 0/0 -j DROP
#
```

E' comunque usato raramente visto che ha lo stesso effetto che si ottiene non specificando affatto l'opzione '-s'.

7.3.2 Specificare una negazione

Molte opzioni, incluse '-s' (o '–source') e '-d' ('–destination') possono avere gli argomenti preceduti dal carattere '!' (si pronuncia 'not'), usato per indicare gli indirizzi NON (not) uguali a quello indicato. Per esempio '-s ! localhost' indica qualsiasi pacchetto **non** proveniente da localhost.

7.3.3 Specificare il protocollo

Il protocollo può essere specificato usando l'opzione '-p' (o '–protocol'). Il protocollo può essere un numero (se conosci il valore numerico del protocollo per IP), oppure un nome, per i casi particolari 'TCP', 'UDP' o 'ICMP'. Maiuscolo o minuscolo non fa differenza, 'tcp' va bene come anche 'TCP'.

Il nome del protocollo può essere preceduto da un '!', per negarlo, ad esempio con '-p ! TCP' si specificano tutti i pacchetti che **non** sono TCP.

7.3.4 Specificare un'interfaccia

Le opzioni '-i' (o '–in-interface') e '-o' (o '–out-interface') servono a specificare il nome di un'interfaccia. Un'interfaccia è il dispositivo fisico dal quale un pacchetto entra ('-i') o esce ('-o'). Usa il comando `ifconfig` per ottenere un elenco delle interfacce 'attive' (funzionanti in quel momento).

I pacchetti che attraversano la catena `INPUT` non hanno un'interfaccia di output, perciò qualsiasi regola che usa '-o' in questa catena non troverà mai una corrispondenza. Allo stesso modo i pacchetti che attraversano la catena `OUTPUT` non hanno un'interfaccia di input, quindi nessuna regola che usi '-i' in questa catena troverà mai una corrispondenza.

Solo i pacchetti che attraversano la catena `FORWARD` hanno sia un'interfaccia di input che di output.

E' perfettamente legale specificare un'interfaccia che al momento non esiste; la regola non sarà mai soddisfatta fino a quando l'interfaccia non sarà attivata. Ciò è estremamente utile per i collegamenti dial-up PPP (di solito interfaccia `ppp0`) e simili.

Come caso speciale, un nome di interfaccia che termina con un '+' corrisponderà a tutte le interfacce (esistenti o meno) che cominciano con quella stringa. Per esempio, per specificare una regola che corrisponda a tutte le interfacce PPP si può utilizzare l'opzione `-i ppp+`.

Il nome dell'interfaccia può essere preceduto da un '!' per far sì che sia soddisfatta da tutte le interfacce che **NON** corrispondono all'interfaccia (o alle interfacce) specificata.

7.3.5 Specificare i frammenti

Qualche volta un pacchetto è troppo grande per passare tutto in una volta attraverso il cavo. Quando accade, il pacchetto viene suddiviso in **frammenti**, e spedito sotto forma di pacchetti multipli. L'altro estremo della connessione si occuperà di riassemblare questi frammenti e di ricostruire l'intero pacchetto.

Il problema dei frammenti è che il frammento iniziale contiene i campi completi delle intestazioni (IP + TCP, UDP e ICMP) da esaminare, mentre i pacchetti successivi hanno solo un sottoinsieme delle intestazioni (IP senza i campi dei protocolli addizionali). Ed è proprio per questa ragione che non è possibile sbirciare le intestazioni dei protocolli in questi pacchetti.

Se stai effettuando il tracciamento delle connessioni (connection tracking) o il NAT, tutti i frammenti saranno ricomposti prima che raggiungano il codice che si occupa di filtrare i pacchetti, perciò non c'è motivo di temere i frammenti.

E' importante comunque comprendere come i frammenti vengono considerati dalle regole di filtraggio. Qualsiasi regola di filtraggio che richieda informazioni, che in realtà non sono presenti, *non* sarà soddisfatta. Quindi il primo frammento sarà trattato come un qualsiasi pacchetto, il secondo e i successivi no. In questo modo una regola `-p TCP -sport www` che specifica una porta d'origine 'www' non sarà mai soddisfatta da un frammento (escluso il primo). E nemmeno la regola opposta `-p TCP -sport !  www`.

Si può comunque indicare una regola specifica per il secondo e i successivi frammenti usando l'opzione '-f' (o '–fragment'). Inoltre è perfettamente legale specificare una regola con cui si indica, inserendo '!' prima di '-f', che questa *non* deve essere applicata al secondo e ai successivi frammenti.

Di solito è considerata buona norma lasciar passare il secondo e i successivi frammenti, poiché il filtraggio avrà effetto sul primo frammento, prevenendo la ricostruzione nell'host di destinazione. Sono noti comunque alcuni bug che causano il crash delle macchine semplicemente inviando dei frammenti. Sei avvisato.

Una nota per i capoccioni della rete: i pacchetti malformati (TCP, UDP e i pacchetti ICMP troppo corti affinché il codice di firewalling possa leggere le porte o il codice e tipo ICMP) quando sono esaminati sono scartati. Sono considerati insomma frammenti TCP che iniziano alla posizione 8.

Ad esempio, la regola seguente scarterà qualsiasi frammento diretto a 192.168.1.1:

```
# iptables -A OUTPUT -f -d 192.168.1.1 -j DROP
#
```

7.3.6 Estensioni di iptables: nuovi confronti

`iptables` è **estendibile**, ossia sia il kernel che il tool iptables possono essere estesi per fornire nuove caratteristiche.

Alcune di queste estensioni sono standard, altre sono più esotiche. Le estensioni possono essere realizzate da altre persone e distribuite separatamente per nicchie di utenti.

Le estensioni del kernel normalmente si trovano nella sottodirectory dei moduli del kernel, ad esempio /lib/modules/2.3.15/net. Sono poi caricate a richiesta, a meno che il kernel non sia stato compilato con la voce CONFIG_KMOD attivata, in questo caso non sarà necessario caricarli manualmente.

Le estensioni del programma iptables sono delle librerie condivise collocate nella directory /usr/local/lib/iptables/, sebbene una distribuzione potrebbe anche inserirle in /lib/iptables o /usr/lib/iptables.

Le estensioni sono di due tipi: nuovi obiettivi (target) e nuovi confronti. Parleremo dei nuovi obiettivi più avanti. Alcuni protocolli offrono nuovi test, correntemente questi sono TCP, UDP e ICMP come mostrati sotto.

Per questi ultimi puoi specificare i nuovi test sulla linea di comando dopo l'opzione '-p', che provvederà a caricare l'estensione. Nel caso di nuovi test specifici, per caricare l'estensione, si utilizzerà l'opzione '-m', dopo di ché saranno disponibili le opzioni delle estensioni.

Per ottenere aiuti sulle estensioni, usa l'opzione per caricarle ('-p', '-j' o '-m') seguite da '-h' o '–help', ad esempio:

```
# iptables -p tcp --help
#
```

Estensioni TCP Le estensioni riguardanti TCP sono caricate automaticamente se si specifica l'opzione '-p tcp'. Si possono così usare le seguenti opzioni (nessuna di queste viene soddisfatta dai frammenti).

–tcp-flags

Seguita eventualmente da un '!', e da due stringhe di flag, permette di filtrare in base ai flag specifici di TCP. La prima stringa di flag è la maschera: lista di flag che vuoi esaminare. La seconda stringa di flag invece serve ad indicare quale o quali dovrebbero risultare impostati.

Ad esempio:

```
# iptables -A INPUT --protocol tcp --tcp-flags ALL SYN,ACK -j DENY
```

Questo comando indica che devono essere esaminati tutti i flag ('ALL' è sinonimo di 'SYN,ACK,FIN,RST,URG,PSH'), ma solo i flag SYN e ACK dovrebbero risultare impostati. Esiste anche un argomento 'NONE' che indica nessun flag.

–syn

è un'abbreviazione di '–tcp-flags SYN,RST,ACK SYN' e può essere preceduta da '!'.

–source-port

Seguita eventualmente da un '!', e da una porta TCP singola o un intervallo di porte. Le porte possono essere indicate con il nome, come elencate in /etc/services, o con i numeri. Gli intervalli possono essere due porte separate da un '-', o una porta seguita da un '-' (per specificare tutte le porte uguali o maggiori di quella indicata) oppure una porta preceduta da un '-' (per specificare tutte le porte uguali o inferiori di quella indicata).

–sport

è sinonimo di '–source-port'.

–destination-port

e

–dport

sono equivalenti a ciò che abbiamo appena visto, solamente che servono a specificare la porta di destinazione, invece di quella sorgente.

–tcp-option

Seguita eventualmente da '!' e da un numero, un pacchetto la soddisfa se ha un'opzione TCP uguale a questo numero. Un pacchetto che non ha un'intestazione TCP completa viene automaticamente scartato al momento della verifica delle sue opzioni TCP.

Un chiarimento sui flag del TCP Qualche volta è utile permettere connessioni TCP in una direzione ma non nell'altra. Per esempio, potresti aver voglia di permettere connessioni verso un server WWW esterno, ma non da questo verso di te.

L'approccio più semplice sarebbe quello di bloccare i pacchetti TCP provenienti dal server. Sfortunatamente le connessioni TCP per funzionare correttamente richiedono che i pacchetti viaggino in entrambe le direzioni.

La soluzione consiste nel bloccare solo i pacchetti usati per richiedere una connessione. Questi pacchetti sono detti pacchetti **SYN** (ok, tecnicamente sono pacchetti con il flag SYN impostato, e i flag FIN ed ACK azzerati, ma per comodità li chiameremo pacchetti SYN). Se non accettiamo solo questi pacchetti allora possiamo impedire i tentativi di connessione.

Il flag '–syn' è usato a questo scopo ed è valido solo per regole che specificano TCP come protocollo. Ad esempio, per specificare un tentativo di connessione TCP da 192.168.1.1 usare:

```
-p TCP -s 192.168.1.1 --syn
```

Questo flag può essere negato facendolo precedere da un '!', che sta a significare qualsiasi pacchetto eccetto quello di inizio connessione.

Estensioni UDP Queste estensioni sono caricate automaticamente se si specifica '-p udp'. Si potranno così utilizzare le opzioni '–source-port', '–sport', '–destination-port' e '–dport' come spiegato prima riguardo al TCP.

Estensioni ICMP Queste estensioni sono caricate automaticamente se si specifica '-p icmp'. Fornisce solo una nuova opzione:

–icmp-type

seguita eventualmente da un '!', oltre che dal nome di un tipo di icmp (es. 'host-unreachable'), o da un tipo numerico (es. '3'), o da un tipo numerico più codice separati da un '/' (es. '3/3'). Si può ottenere un elenco dei tipi di icmp utilizzando '-p icmp –help'.

Altre estensioni Le altre estensioni presenti nel pacchetto netfilter sono le estensioni dimostrative, che (se installate) possono essere invocate con l'opzione '-m'.

mac

Questo modulo deve essere specificato esplicitamente con '-m mac' o '–match mac'. E' usato per confrontare i pacchetti in arrivo da una sorgente con indirizzo Ethernet (MAC), ed è utile per i pacchetti che attraversano le catene PREROUTING e INPUT. E' presente una sola opzione:

–mac-source

seguita eventualmente da un '!' (opzionale), e da un indirizzo ethernet in notazione hexbyte separata dal simbolo ':', es. '–mac-source 00:60:08:91:CC:B7'.

limit

Questo modulo deve essere specificato esplicitamente con '-m limit' o '–match limit'. E' usato per restringere il numero di confronti, come anche per sopprimere i messaggi di log. Effettuerà confronti solo un certo numero di volte al secondo (per default 3 confronti all'ora, con un burst (raffica) di 5). Richiede due argomenti facoltativi:

–limit

seguita da un numero, specifica la media massima di confronti permessi al secondo. Si può specificare l'unità esplicitamente usando '/second', '/minute', '/hour' o '/day', o parte di essi (quindi '5/second' è equivalente a '5/s').

–limit-burst

seguita da un numero, indica il massimo burst (raffica) prima che il limite visto sopra li respinga.

Queste estensioni possono essere usate spesso con l'obiettivo LOG per limitare il numero di registrazioni. Per comprendere come funzionano, diamo un'occhiata alla seguente regola, che registra i pacchetti usando i parametri di default di limit:

```
# iptables -A FORWARD -m limit -j LOG
```

La prima volta che la regola è consultata, il pacchetto sarà registrato, infatti il burst (raffica) per default è impostato a 5, i primi cinque pacchetti saranno perciò registrati. Dopo di ché, dovranno trascorrere venti minuti prima che un pacchetto sia nuovamente registrato da questa regola, senza riguardo al numero dei pacchetti arrivati. Inoltre ogni venti minuti trascorsi senza che siano giunti pacchetti, uno di questi sarà recuperato nel burst; se nessun pacchetto viene sottoposto alla regola per 100 minuti, il burst sarà completamente ricaricato; da dove cominciato.

Attualmente non è possibile creare una regola con un tempo di ricarica maggiore di 59 ore, perciò se imposti un flusso medio di 1 per ogni giorno, allora il burst (raffica) dovrà essere inferiore a 3.

Puoi usare questo modulo anche per evitare i vari attacchi denial of service (DoS) impostando un rate maggiore per incrementare la reattività.

Protezione dal Syn-flood:

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

Port scanner sospetti:

```
# iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit 1/s -j ACCEPT
```

Ping della morte (Ping of death):

```
# iptables -A FORWARD -p icmp --icmp-type echo-request -m limit 1/s -j ACCEPT
```

Questo modulo funziona come una porta isterica, come mostrato nel seguente grafico.

```
           rate (pacchetti/s)
                 ^        .---.
                 |       / DoS \
                 |      /       \
Limite DoS  -|.....:.........\.......................
 = (limit *  |    /:          \
limit-burst) |   / :           \         .-.
```

```
               |  /  :           \       /   \
               | /   :            \     /     \
Fine DoS      -|/....:.............:.../.......\..../.
 = limit       |     :             :`-'        `--'
-------------+-----+--------------+------------------> tempo (s)
  LOGICA =>  Confr.|  Non confr.  |    Confronta
```

In questo modo diciamo di confrontare un pacchetto al secondo con un burst (raffica) di cinque pacchetti, i pacchetti cominciano ad arrivare quattro volte al secondo, per tre secondi, poi ancora per altri tre secondi.

```
              <--Flusso 1-->          <---Flusso 2--->

Pacchetti ^                  Line  __--      YNNN
 Totali   |              Rate  __--      YNNN
          |           mum  __--      YNNN
       10 |       Maxi __--         Y
          |         __--           Y
          |     __--              Y
          | __--    YNNN
          |-    YNNN
        5 |    Y
          |   Y                           Legenda:  Y -> Regola soddisfatta
          |  Y                                      N -> Regola non soddisfatta
          | Y
          |Y
        0 +--------------------------------------------------->  Tempo (secondi)
           0   1   2   3   4   5   6   7   8   9  10  11  12
```

Come puoi vedere è consentito ai primi 5 pacchetti eccedere il pacchetto al secondo, ossia il limite fissato. Se c'è una pausa, è consentito un altro burst (raffica) che però non deve eccedere il rate massimo fissato con la regola (in questo caso 1 pacchetto al secondo dopo che il burst è stato usato).

owner

Questo modulo tenta di confrontare varie caratteristiche del creatore dei pacchetti generati localmente. E' valido solo nella catena OUTPUT, anche nel caso in cui alcuni pacchetti (come ad esempio risposte a ICMP ping) non abbiamo un proprietario, e che quindi non siano confrontabili.

–uid-owner userid

Confronta se il pacchetto è stato creato da un processo con lo user id (numerico) effettivo indicato.

–uid-owner groupid

Confronta se il pacchetto è stato creato da un processo con il group id (numerico) effettivo indicato.

–pid-owner processid

Confronta se il pacchetto è stato creato da un processo con l'id di processo indicato.

–sid-owner processid

Confronta se il pacchetto è stato creato da un processo della sessione di gruppo indicata.

unclean

Questo modulo sperimentale deve essere specificato esplicitamente con '-m unclean' o '–match unclean'. Effettua vari controlli casuali sulla correttezza di un pacchetto. Questo modulo non è stato verificato e non dovrebbe essere usato per la sicurezza (probabilmente può portare a dei peggioramenti in quanto potrebbe contenere dei bug). Non sono disponibili opzioni.

Condizione state Il criterio di selezione più utile è dato dall'estensione 'state', che interpreta le analisi del connection-tracking (tracciamento delle connessioni) del modulo 'ip_conntrack'. Questo modulo è assolutamente raccomandato.

Specificando '-m state' si può utilizzare l'opzione addizionale '–state', seguita da una lista di stati, separati da virgole, da confrontare (il flag '!' indica di **non** confrontare questi stati). Questi stati sono:

NEW

Un pacchetto che crea una nuova connessione.

ESTABLISHED

Un pacchetto che appartiene ad una connessione esistente (ossia una connessione che ha avuto dei pacchetti in risposta).

RELATED

Un pacchetto che è relativo, ma che non fa parte, di una connessione esistente, come ad esempio un errore ICMP, o (con il modulo FTP caricato) un pacchetto che tenta una connessione ftp data.

INVALID

Un pacchetto che non può essere identificato per alcune ragioni, tra cui ad esempio esaurimento della memoria ed errori ICMP che non corrispondono a nessuna connessione conosciuta. Generalmente questi pacchetti dovrebbero essere scartati.

7.4 Specificare gli obiettivi

Ora conosciamo quali controlli possiamo effettuare sui pacchetti, abbiamo quindi bisogno di qualcosa che dica cosa fare dei pacchetti che soddisfano i nostri test. Questo è chiamato **obiettivo** (target) di una regola.

Ci sono due semplici obiettivi già disponibili: DROP e ACCEPT. Li abbiamo già incontrati. Se un pacchetto soddisfa una regola e l'obiettivo è uno di questi due, nessun'altra regola è consultata: il destino del pacchetto è stato deciso.

Ci sono altri due tipi di obiettivi oltre a quelli già disponibili: le estensioni e le catene create dall'utente.

7.4.1 Catene create dall'utente

Una caratteristica potente che `iptables` ha ereditato da `ipchains` è la possibilità per l'utente di creare nuove catene, che si aggiungono alle tre catene già disponibili (INPUT, FORWARD e OUTPUT). Per distinguerle, le catene create dall'utente per convenzione sono in minuscolo (descriveremo come creare una nuova catena successivamente in 7.5 (Operazioni su intere catene)).

Quando un pacchetto soddisfa una regola che ha per obiettivo una catena creata dall'utente, il pacchetto comincia ad attraversare le regole di quest'ultima. Se la catena termina e il destino del pacchetto non è stato deciso allora si passa nuovamente alla catena corrente però alla regola successiva.

E ancora tempo di arte ASCII. Considera due (sciocche) catene: `INPUT` (catena già disponibile) e `test` (catena creata dall'utente).

```
 'INPUT'                         'test'
----------------------------    ----------------------------
| Regola1: -p ICMP -j DROP |    | Regola1: -s 192.168.1.1  |
|--------------------------|    |--------------------------|
| Regola2: -p TCP -j test  |    | Regola2: -d 192.168.1.1  |
```

```
|--------------------------|    -----------------------------
| Regola3: -p UDP -j DROP  |
----------------------------
```

Considera un pacchetto TCP proveniente da 192.168.1.1 e diretto a 1.2.3.4. Questo entra nella catena `INPUT`, viene sottoposto alla Regola1 - non soddisfatta. Regola2 invece è soddisfatta, e il suo obiettivo è `test`, perciò la regola successiva a cui sottoporre il pacchetto è la prima di `test`. La Regola1 di `test` è soddisfatta, ma non specifica un obiettivo, quindi viene esaminata la regola successiva, Regola2. Questa non è soddisfatta, e inoltre abbiamo raggiunto la fine della catena. Torniamo alla catena `INPUT`, dove abbiamo già esaminato Regola2, perciò passiamo a Regola3, che non è soddisfatta.

Quindi il percorso del pacchetto è:

```
                        v     __________________________
 `INPUT'                |  /    `test'                 v
------------------------|--/    ------------------------|----
| Regola1               | /|    | Regola1               |   |
|-----------------------|/-|    |-----------------------|---|
| Regola2               /  |    | Regola2               |   |
|--------------------------|    ------------------------v----
| Regola3               /--+____________________________/
------------------------|---
                        v
```

Le catene definite dall'utente possono saltare in altre catene sempre definite dall'utente (ma non creare dei cicli: i pacchetti appartenenti ad un ciclo saranno scartati).

7.4.2 Estensioni a iptables: nuovi obiettivi

Un altro tipo di obiettivo può essere un'estensione. Un obiettivo estensione, per provvedere nuove opzioni per la linea di comando, deve consistere in un modulo per il kernel, e in un'estensione opzionale per `iptables`. Ci sono molte estensioni incluse nella distribuzione di netfilter:

LOG

Questo modulo permette la registrazioni da parte del kernel dei pacchetti specificati. Ha le seguenti opzioni:

–log-level

seguito da un numero o nome di livello. Nomi validi sono (minuscoli o maiuscoli) 'debug', 'info', 'notice', 'warning', 'err', 'crit', 'alert' e 'emerg', che corrispondono in ordine ai numeri dal 7 allo 0. Leggi le pagine del manuale di syslog.conf per una spiegazione di questi livelli.

–log-prefix

seguito da una stringa di max. 30 caratteri, questo messaggio è collocato all'inizio del messaggio registrato, per permettere di poterlo identificare immediatamente.

Questo modulo è molto utile dopo un obiettivo limit, in quanto evita un numero esagerato di registrazioni.

REJECT

Questo modulo ha lo stesso effetto di 'DROP', però in più viene inviato in risposta un messaggio di errore ICMP di tipo 'port unreachable'. Nota che il messaggio di errore non è inviato se (vedi RFC 1122):

- Il pacchetto filtrato era in primo luogo un messaggio di errore ICMP, oppure un ICMP di tipo sconosciuto.
- Il pacchetto filtrato non era un frammento di testa.
- Abbiamo inviato recentemente alla destinazione troppi messaggi di errore.

REJECT ha inoltre un '–reject-with' opzionale che altera il pacchetto inviato in risposta: leggi le pagine del manuale.

7.4.3 Obiettivi speciali già disponibili

Ci sono due obiettivi speciali già disponibili: : `RETURN` e `QUEUE`.

`RETURN` ha lo stesso effetto di quando si arriva alla fine di una catena: se l'ultima regola appartiene a una catena predefinita, allora viene eseguita la tattica della catena. Altrimenti se appartiene ad una catena definita dall'utente, si prosegue con la catena precedente, con la regola successiva a quella che aveva causato il salto nella catena dell'utente.

`QUEUE` è un obiettivo speciale, che accoda i pacchetti per elaborazioni userspace. Perché possa essere utile sono necessari ulteriori componenti:

- un gestore delle code, che si occupi dell'attuale meccanismo di passaggio tra il kernel e lo userspace, e
- un'applicazione userspace che riceva, possibilmente manipoli ed emetta verdetti sui pacchetti.

Il gestore standard della coda, per IPV4 e iptables, è il modulo ip_queue, distribuito con il kernel ancora come sperimentale.

Il seguente è un esempio veloce di come si possa usare iptables per accodare i pacchetti per elaborazioni userspace:

```
# modprobe iptable_filter
# modprobe ip_queue
# iptables -A OUTPUT -p icmp -j QUEUE
```

Con questa regola i pacchetti ICMP generati localmente (creati diciamo con ping) sono passati al modulo ip_queue che tenta di consegnarli all'applicazione userspace. Se non c'è in attesa un'applicazione userspace i pacchetti saranno scartati.

Per scrivere un'applicazione userspace, usa le API della libipq, distribuita con iptables. Sorgenti di esempio si possono trovare nella suite di tool di test (es. redirect.c) nella CVS.

Lo stato della ip_queue può essere verificato attraverso:

```
/proc/net/ip_queue
```

La lunghezza massima della coda (ossia il numero di pacchetti consegnati allo userspace senza emissione di verdetto) può essere controllata attraverso:

```
/proc/sys/net/ipv4/ip_queue_maxlen
```

Il valore di default della lunghezza massima della coda è fissata a 1024. Quando questo limite è raggiunto, i nuovi pacchetti saranno scartati finché la lunghezza della coda non tornerà nuovamente sotto il limite. Protocolli buoni come TCP interpretano i pacchetti scartati come congestione, e speranzosi rinunciano quando la coda è piena. Comunque può fare alcuni esperimenti per determinare una lunghezza massima ideale della coda per una certa situazione se il valore di default è troppo piccolo.

7.5 Operazioni su intere catene.

Una caratteristica veramente utile di `iptables` è l'abilità di raggruppare regole correlate in catene. Puoi chiamare una catena come vuoi, ma raccomando di usare lettere minuscole per evitare confusioni con le catene predefinite e con gli obiettivi. I nomi delle catene possono arrivare ad una lunghezza di 31 lettere.

7.5.1 Creare una nuova catena

Creiamo una nuova catena. Visto che sono una persona di grande immaginazione, chiamiamola `test`. Usiamo l'opzione '-N' oppure '–new-chain':

```
# iptables -N test
#
```

Questa è semplice. Ora possiamo aggiungervi le regole come spiegato precedentemente.

7.5.2 Cancellare una catena

Cancellare una catena è ugualmente semplice, basta usare l'opzione '-X' o '–delete-chain'. Perché '-X'? Beh, le altre lettere buone erano già state assegnate.

```
# iptables -X test
#
```

Ci sono un paio di restrizioni che riguardano la cancellazione delle catene: devono essere vuote (vedi 7.5.3 (Svuotare una catena) sotto) e non devono essere l'obiettivo di nessuna regola. Non puoi inoltre cancellare nessuna delle tre catene predefinite.

Se non si specifica una catena, allora *tutte* le catene definite dall'utente saranno cancellate, se possibile.

7.5.3 Svuotare una catena

C'è un modo semplice per svuotare una catena di tutte le sue regole, ossia usare il comando '-F' (o '–flush').

```
# iptables -F forward
#
```

Se non si specifica una catena, allora *tutte* le catene saranno svuotate.

7.5.4 Consultare una catena

Puoi ottenere una lista delle regole di una catena usando il comando '-L' (o '–list').

Il valore 'refcnt' mostrato per ogni catena definita dall'utente rappresenta il numero delle regole che hanno questa catena come loro obiettivo. Deve essere a zero (catena vuota) prima di poter cancellare una catena.

Se il nome della catena è omesso, saranno elencate tutte le catene, anche quelle vuote.

Ci sono tre opzioni che possono accompagnare '-L'. L'opzione '-n' (numeric) è davvero utile in quanto previene che `iptables` cerchi di capire a quale nome corrisponde l'indirizzo IP, che (se stai utilizzando il DNS come fanno molte persone) causa lunghe attese se il DNS non è impostato correttamente, o se le richieste DNS sono respinte dal filtraggio. Inoltre visualizza le porte TCP e UDP con i rispettivi numeri invece che con i nomi.

L'opzione '-v' mostra tutti i dettagli delle regole, come i contatori dei pacchetti e dei byte, le comparazioni TOS, e le interfacce. Se non la si specifica queste informazioni sono omesse.

Nota che i contatori dei pacchetti e dei byte sono visualizzati usando il suffisso 'K', 'M' o 'G' che significano rispettivamente 1000, 1000000 e 1000000000. Usando l'opzione '-x' (expand numbers) verranno visualizzati i numeri completi, senza preoccuparsi di quanto siano grandi.

7.5.5 Resettare (azzerare) i contatori

E' utile resettare i contatori. Ciò può essere ottenuto con l'opzione '-Z' (o '–zero').

Il problema con questo approccio è che talvolta si ha bisogno di conoscere i valori dei contatori immediatamente prima che vengano azzerati. Nell'esempio suddetto, alcuni pacchetti potrebbero passare tra i comandi '-L' e '-Z'. Per questa ragione, per resettarli mentre li si sta leggendo, è possibile usare '-L' e '-Z' *assieme*.

7.5.6 Impostare la tattica

Abbiamo visto cosa succede quando un pacchetto raggiunge la fine della catena predefinita precedentemente, quando abbiamo parlato di come i pacchetti attraversano le catene. In questo caso, la **tattica** della catena determina il destino del pacchetto. Solo le catene predefinite (`INPUT`, `OUTPUT` e `FORWARD`) hanno delle tattiche, in quanto se un pacchetto raggiunge la fine di una catena definita dall'utente, proseguirà riprendendo il percorso della catena precedente.

La tattica può essere `ACCEPT` o `DROP`.

8 Usare ipchains e ipfwadm

Ci sono dei moduli nella distribuzione netfilter denominati ipchains.o e ipfwadm.o. Inseriscine uno nel kernel (NOTA: sono incompatibili con iptables.o, ip_conntrack.o e ip_nat.o!), poi usa tranquillamente ipchains o ipfwadm come fai di solito.

Questi moduli saranno supportati ancora per un po' di tempo. Ritengo ragionevole la formula 2 * [avviso della sostituzione - release iniziale stabile], oltre la data della disponibilità di una release stabile del successore.

Questo significa che per ipfwadm il supporto cesserà nel (CORREGGI: indica data reale):

```
2 * [Ottobre 1997 (2.1.102 release) - Marzo 1995 (ipfwadm 1.0)]
        + Gennaio 1999 (2.2.0 release)
    = Novembre 2003.
```

Questo significa che per ipchains il supporto cesserà nel (CORREGGI: indica data reale):

```
2 * [Agosto 1999 (2.3.15 release) - Ottobre 1997 (2.2.0 release)]
        + Gennaio 2000 (2.3.0 release?)
    = Settembre 2003.
```

Comunque fino al 2004 non dovrai preoccuparti.

9 Mescolare NAT e filtraggio dei pacchetti

E' abbastanza comune il desiderio di effettuare sia il Network Address Translation (vedi il NAT HOWTO) che il filtraggio dei pacchetti. La buona notizia è che assieme funzionano davvero bene.

Si può realizzare il filtraggio dei pacchetti ignorando completamente qualsiasi NAT presente, in quanto le sorgenti e le destinazioni controllate dal filtro dei pacchetti sono le sorgenti e le destinazioni 'reali'.

Ad esempio, se stai effettuando del DNAT per inviare tutte le connessioni dirette a 10.1.1.1 porta 8080 verso 1.2.3.4 porta 80, il filtro dei pacchetti vedrà solo dei pacchetti diretti all'indirizzo 10.1.1.1 porta 8080 (la destinazione reale), e non verso 1.2.3.4 porta 80. Allo stesso modo si può ignorare anche il mascheramento: i pacchetti sembreranno provenire dall'indirizzo IP reale (diciamo 10.1.1.1), e le risposte ritorneranno indietro lì.

Puoi usare l'estensione 'state' senza obbligare il filtraggio dei pacchetti a del lavoro extra, il NAT richiede comunque il tracciamento delle connessioni. Per migliorare il semplice esempio sul mascheramento presente nel NAT HOWTO vietiamo tutte le richieste di nuova connessione che possono arrivare all'interfaccia ppp0, in questo modo:

```
# Mascheriamo ppp0
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

# I pacchetti di tipo NEW e INVALID in ingresso (INPUT) o in transito (FORWARD)
# non sono consentiti, ossia scartati.
iptables -A INPUT -i ppp0 -m state --state NEW,INVALID -j DROP
iptables -A FORWARD -i ppp0 0 -m state --state NEW,INVALID -j DROP

# Abilitiamo l'IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
```

10 Differenze tra iptables e ipchains

- Prima di tutto i nomi delle catene predefinite sono passate da minuscolo in MAIUSCOLO, dato che ora le catene INPUT e OUTPUT si occupano solo di pacchetti destinati localmente e generati localmente. Sono usati per vedere rispettivamente tutti i pacchetti entranti e tutti i pacchetti uscenti.
- L'opzione '-i' ora serve ad indicare l'interfaccia di ingresso, e funziona solo con le catene INPUT e FORWARD. Le regole nelle catene FORWARD e OUTPUT che prima usavano l'opzione '-i' ora devono essere cambiate e usare '-o'.
- TCP e UDP ora devono essere accompagnate anche dalle opzioni –source-port o –sport (oppure –destination-port/–dport), in questo modo le estensioni TCP e UDP sono rispettivamente caricate.
- L'opzione TCP -y è stata sostituita con –syn, e deve seguire '-p tcp'.
- Finalmente l'obiettivo DENY è stato sostituito con DROP.
- Azzerare singole catene mentre le si visualizzano ora funziona.
- Azzerare le catene predefinite comporta anche l'azzeramento dei contatori della tattica.
- Visualizzare le catene fornisce una istantanea completa dei contatori.
- REJECT e LOG sono ora obiettivi estensione, ossia sono moduli separati dal kernel.
- I nomi delle catene possono essere lunghi 31 caratteri.
- MASQ ora è MASQUERADE e usa una sintassi differente. REDIRECT, pur avendo mantenuto lo stesso nome, ha subito un cambiamento nella sintassi. Leggi il NAT-HOWTO per maggiori informazioni su come configurarli.

- L'opzione -o non è più utilizzata per dirigere i pacchetti nello userspace (vedi -i sopra). I pacchetti sono ora inviati allo userspace attraverso l'obiettivo QUEUE.
- Probabilmente una montagna di altre cose che non ricordo.

11 Consigli su come realizzare il filtraggio dei pacchetti

Nell'area della sicurezza dei computer è considerato saggio bloccare qualsiasi cosa e poi aprire i buchi strettamente necessari. Si dice in genere 'tutto ciò che non è esplicitamente permesso è proibito'. Raccomando questo approccio se il tuo massimo interesse è la sicurezza.

Non avviare nessun servizio a meno che tu non ne abbia bisogno, anche se pensi di averne bloccato l'accesso.

Se stai creando un firewall dedicato, comincia dal non avviare nulla, poi aggiungi i servizi e lascia passare i pacchetti che sono necessari.

Raccomando security in depth (sicurezza in profondità): combina tcp-wrapper (per connessioni al filtro dei pacchetti stesso), proxy (per connessioni che transitano attraverso il filtro dei pacchetti), verifica degli instradamenti e filtraggio dei pacchetti. La verifica degli instradamenti consiste nello scartare i pacchetti che arrivano da un'interfaccia inaspettata. Ad esempio se la tua rete locale ha indirizzi del tipo 10.1.1.0/24, e un pacchetto con questo indirizzo sorgente arriva alla tua interfaccia esterna, allora sarà scartato. Questo può essere abilitato per una interfaccia (ppp0) così:

```
# echo 1 > /proc/sys/net/ipv4/conf/ppp0/rp_filter
#
```

O per tutte le interfacce esistenti e future in questo modo:

```
# for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
#     echo 1 > $f
# done
#
```

Debian lo fa per default dove possibile. Se hai instradamento asimmetrico (ossia aspetti pacchetti provenienti da diverse direzioni), vorrai disabilitare il filtraggio su queste interfacce.

Registrare i messaggi è utile quando si imposta un firewall e qualcosa non funziona, ma con un firewall pronto all'uso, combinalo sempre con 'limit' per impedire che qualcuno cerchi di inondarti di messaggi.

Raccomando il connection tracking (tracciamento delle connessioni) per i sistemi sicuri: introduce un po' di lavoro in più, in quanto tutte le connessioni saranno tracciate, ma è davvero utile per controllare gli accessi alla tua rete. Potrebbe essere necessario caricare il modulo 'ip_conntrack.o' se non è compilato direttamente nel kernel o se il kernel non carica i moduli automaticamente. Se vuoi tenere traccia accurata di protocolli complessi, allora devi caricare il modulo appropriato di aiuto (es. 'ip_conntrack_ftp.o').

```
# iptables -N no-conns-from-ppp0
# iptables -A no-conns-from-ppp0 -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A no-conns-from-ppp0 -m state --state NEW -i ! ppp0 -j ACCEPT
# iptables -A no-conns-from-ppp0 -i ppp0 -m limit -j LOG --log-prefix "Bad packet from ppp0:"
# iptables -A no-conns-from-ppp0 -i ! ppp0 -m limit -j LOG --log-prefix "Bad packet not from ppp0:"
# iptables -A no-conns-from-ppp0 -j DROP

# iptables -A INPUT -j no-conns-from-ppp0
# iptables -A FORWARD -j no-conns-from-ppp0
```

Come realizzare un buon firewall va oltre lo scopo di questo HOWTO, consiglio comunque di essere sempre ‘minimalisti’. Leggi il Security HOWTO per maggiori informazioni su come fare il test e provare la tua box.