Linux IPCHAINS HOWTO

Rusty Russell

v1.0.8, Tue Jul 4 14:20:53 EST 2000

Questo documento intende descrivere come ottenere, installare e configurare il software di “enhanced IP firewalling chains” per Linux, e alcune idee su come si potrebbe usarlo. Traduzione a cura di Giovanni Bortolozzo *borto at pluto.linux.it* .

Indice

1 Introduzione

Questo è il Linux IPCHAINS-HOWTO; si veda la Sezione 1.4 (Dove?) per conoscere l'indirizzo del sito principale, che contiene l'ultima versione. Si dovrebbe leggere anche il Linux NET-3-HOWTO. L'IP-Masquerading HOWTO, il PPP-HOWTO, l'Ethernet-HOWTO e il Firewall HOWTO possono essere altre letture interessanti (come d'altronde lo possono essere le FAQ di alt.fan.bigfoot).

Se interessa il filtraggio dei pacchetti (packet filtering), si legga la Sezione 1.2 (Perché?), la Sezione 2.3 (Come?) e si dia un'occhiata ai titoli nella sezione 4 (IP Firewalling Chains).

Se si sta operando la conversione da `ipfwadm`, si legga la Sezione 1 (Introduzione), la sezione 2.3 (Come?) e le appendici nella sezione 8 (Differenze tra ipchains e ipfwadm) e nella sezione 9 (L'uso dello script 'ipfwadm-wrapper').

1.1 Cos'è?

Il Linux `ipchains` è una riscrittura del codice di firewalling IPv4 di Linux (il quale era stato rubato principalmente da BSD) e una riscrittura di `ipfwadm`, che a sua volta era, almeno penso, una riscrittura di `ipfw` di BSD. È richiesto per amministrare il filtraggio dei pacchetti IP nelle versioni del kernel di Linux 2.1.102 e superiori.

1.2 Perché?

Il vecchio codice di firewalling di Linux non gestisce i frammenti (fragment), ha contatori a 32 bit (almeno su Intel), non permette di specificare protocolli diversi da TCP, UDP e ICMP, non può rendere atomiche grosse modifiche, non può specificare regole inverse, ha qualche scemata e può risultare difficile da manipolare (rendendolo propenso a errori utente).

1.3 Come?

Attualmente il codice fa parte del kernel a partire dalla versione 2.1.102. Per la serie dei kernel 2.0, è necessario scaricare una patch dalla pagina web. Se il proprio kernel 2.0 è più recente della patch disponibile, una patch più vecchia dovrebbe andare bene lo stesso; questa parte del kernel 2.0 è piuttosto stabile (eg. la patch per il kernel 2.0.34 funziona bene sul kernel 2.0.35). Poiché la patch per il 2.0 è incompatibile con le patch ipportfw e ipautofw, non raccomando di applicarla a meno che non si abbia veramente bisogno di alcune delle funzionalità che offre ipchains.

1.4 Dove?

La pagina ufficiale è in tre posti:

Ringrazio la Penguin Computing `http://netfilter.filewatcher.org/ipchains`

Ringrazio il SAMBA Team `http://www.samba.org/netfilter/ipchains`

Ringrazio Jim Pick `http://netfilter.kernelnotes.org/ipchains`

Esiste una mailing list per le segnalazioni di bug, discussione, sviluppo e uso. È possibile associarsi alla mailing list inviando un messaggio contente “subscribe ipchains-list” a subscribe at east.balius.com. Per scrivere a tutti gli iscritti alla lista si usi ipchains-list at east.balius.com.

2 Fondamenti di packet filtering

2.1 Cos'è?

Tutto il traffico attraverso una rete è inviato sotto forma di **pacchetti**. Per esempio, lo scaricarsi questo documento (facciamo conto sia lungo 50k) fa sì che si ricevano 36 o più pacchetti da 1460 byte ognuno (tanto per mettere un po' di numeri).

L'inizio di ogni pacchetto specifica dove deve andare, da dove viene, il tipo di pacchetto e altri dettagli amministrativi. Questa parte iniziale del pacchetto è detta **header** (intestazione). Il resto del pacchetto, contenente i dati reali da trasmettere, solitamente è detto **body** (corpo).

Alcuni protocolli, come **TCP**, usato per il traffico web, mail e login remoti, usano il concetto di “connessione”: prima di inviare un qualsiasi pacchetto con i dati reali, sono scambiati diversi pacchetti di impostazione (con intestazioni speciali) che dicono «Voglio connettermi», «OK» e «Grazie». Poi sono scambiati i pacchetti normali.

Un filtro di pacchetti (packet filter) è un pezzo di software che guarda l'*intestazione* dei pacchetti che passano e decide il destino dell'intero pacchetto. Potrebbe decidere di **proibire** (deny) il pacchetto (ie. scartare il pacchetto come non fosse mai stato ricevuto), **accettare** (accept) il pacchetto (ie. lasciare che il pacchetto prosegua), o **rifiutare** (reject) il pacchetto (simile a proibire, ma comunica inoltre alla fonte del pacchetto quello che ha fatto).

Sotto Linux, il filtraggio dei pacchetti è contenuto all'interno del kernel e ci sono alcune cosette divertenti che si possono fare con i pacchetti, ma il principio generale è sempre quello di guardare le intestazioni e di decidere la sorte dei pacchetti.

2.2 Perché?

Controllo. Sicurezza. Vigilanza.

Controllo:

quando si usa una macchina Linux per connettere la propria rete interna a un'altra rete (Internet, ad esempio) si ha l'opportunità di permette un certo tipo di traffico e di proibirne dell'altro. Per esempio, l'intestazione di un pacchetto contiene l'indirizzo della destinazione del pacchetto, e quindi si può impedire ai pacchetti di andare verso 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 il mio tempo scaricandoli pedissequamente. Dicendo al filtro dei pacchetti di non permettere alcun pacchetto da o verso l'indirizzo posseduto da doubleclick.net si risolve quel problema (anche se penso ci siano modi migliori per risolverlo).

Sicurezza:

quando la propria macchina Linux è la sola cosa tra il caos di Internet e la propria bella e 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 estranei maliziosi. Un altro esempio: si può voler evitare che gli estranei possano fare telnet nella macchina, anche se tutti gli account hanno una password; oppure si vuole (come molti d'altronde) essere un semplice osservatore in Internet e non un server (volente o nolente): semplicemente non si permetta a nessuno di connettersi, facendo sì che il filtro dei pacchetti rifiuti tutti i pacchetti entranti usati per instaurare una connessione.

Vigilanza:

talvolta una macchina mal configurata in una rete locale decide di emettere pacchetti per il mondo esterno. È bene dire al filtro dei pacchetti di informarvi se succede qualcosa di anormale; forse si può fare qualcosa per risolverlo o forse si è solo curiosi per natura.

2.3 Come?

2.3.1 Un kernel con il filtraggio dei pacchetti

È necessario un kernel che possieda al suo interno il nuovo supporto per le "IP firewall chains" (catene firewall IP). È possibile sapere se il kernel attualmente in esecuzione ce l'ha cercando il file '/proc/net/ip_fwchains'. Se esiste, allora c'è.

Se così non è, è necessario compilare un kernel che supporti le catene firewall IP. Per prima cosa si scarichino i sorgenti del kernel che si vuole. Se si possiede un kernel versione 2.1.102 o superiore non si avrà bisogno di una patch (è già presente nel kernel). Altrimenti si applichi la patch scaricabile dalla pagina web suddetta e si effettui la configurazione come spiegato dettagliatamente nel seguito. Se non si sa come farlo, niente panico: si legga il Kernel-HOWTO.

Le opzioni di configurazione che sarà necessario impostare *per i kernel della serie 2.0* sono:

```
CONFIG_EXPERIMENTAL=y
CONFIG_FIREWALL=y
CONFIG_IP_FIREWALL=y
CONFIG_IP_FIREWALL_CHAINS=y
```

Per i *kernel delle serie 2.1 o 2.2*:

```
CONFIG_FIREWALL=y
CONFIG_IP_FIREWALL=y
```

Lo strumento `ipchains` dialoga con il kernel e lo istruisce su quali pacchetti filtrare. A meno che non si sia dei programmatori, o particolarmente curiosi, questo è il solo modo con cui si controllerà il filtraggio dei pacchetti.

2.3.2 ipchains

Il programma `ipchains` inserisce e cancella regole dalla sezione di filtraggio dei pacchetti del kernel. Ciò significa che qualsiasi cosa si imposti, sarà persa al riavvio; si veda la sezione 2.3.3 (Rendere permanenti le regole) per un metodo per assicurarsi che siano ripristinate all'avvio successivo di Linux.

`ipchains` rimpiazza `ipfwadm`, usato per il vecchio codice di Firewall IP. Nel sito ftp di ipchains è disponibile una serie di script molto utili:

http://netfilter.filewatcher.org/ipchains/ipchains-scripts-1.1.2.tar.gz `http://netfilter.filewatcher.org/ipchains/ipchains-scripts-1.1.2.tar.gz`

Questo pacchetto contiene uno script shell chiamato `ipfwadm-wrapper` che permette di effettuare il filtraggio dei pacchetti come lo si faceva prima. Probabilmente non si dovrebbe usare questo script a meno che non si voglia un metodo veloce per aggiornare un sistema che usa `ipfwadm` (è più lento, non controlla gli argomenti, ecc.). In tal caso, non è necessario proseguire la lettura di questo HOWTO.

Si veda l'appendice 8 (Differeze tra ipchains e ipfwadm) e l'appendice 9 (Usare lo script 'ipfwadm-wrapper') per maggiori dettagli.

2.3.3 Rendere permanenti le regole

La configurazione corrente del firewall è immagazzinata nel kernel e quindi sarà persa al riavvio. Raccomando di usare gli script 'ipchains-save' e 'ipchains-restore' per rendere permanenti le regole. Per far ciò, si impostino le proprie regole, poi (come root) si lanci:

```
# ipchains-save > /etc/ipchains.rules
#
```

Si crei uno script come il seguente:

```
#! /bin/sh
# Script per controllare il filtraggio dei pacchetti

# Se non ci sono regole, non fa niente.
[ -f /etc/ipchains.rules ] || exit 0

case "$1" in
    start)
```

```
            echo -n "Attivazione del filtraggio dei pacchetti:"
            /sbin/ipchains-restore < /etc/ipchains.rules || exit 1
            echo 1 > /proc/sys/net/ipv4/ip_forward
            echo "."
            ;;
        stop)
            echo -n "Disattivazione del filtraggio dei pacchetti:"
            echo 0 > /proc/sys/net/ipv4/ip_forward
            /sbin/ipchains -F
            /sbin/ipchains -X
            /sbin/ipchains -P input ACCEPT
            /sbin/ipchains -P output ACCEPT
            /sbin/ipchains -P forward ACCEPT
            echo "."
            ;;
        *)
            echo "Uso: /etc/init.d/packetfilter {start|stop}"
            exit 1
            ;;
    esac

    exit 0
```

Ci si assicuri che sia eseguito abbastanza presto nella procedura di avvio. Nel mio caso (Debian 2.1), ho creato un link simbolico chiamato 'S39packetfilter' nella directory '/etc/rcS.d' (così sarà eseguito prima di S40network).

3 Sono confuso! Instradamento, masquerading, portforwarding, ipautofw...

Questo HOWTO è sul filtraggio dei pacchetti. Ciò significa decidere quando un pacchetto debba o meno avere il permesso di passare. Comunque, essendo Linux il campo giochi degli hacker, si vorrà probabilmente fare qualcosina di più.

Un problema è che lo stesso strumento ("ipchains") è usato per controllare anche il masquerading e il proxy trasparente, sebbene queste siano concettualmente diverse dal filtraggio dei pacchetti (l'implementazione corrente di Linux innaturalmente le fonde insieme, dando l'impressione che siano strettamente collegate).

Il masquerading e il proxy sono trattati da altri HOWTO, mentre le caratteristiche auto forwading e port forwarding sono controllate da altri strumenti, ma poiché la gente continua a domandarmene, includerò un insieme di scenari comuni e indicherò quando ognuno possa essere applicato. I meriti di sicurezza di ciascuna configurazione non saranno qui discussi.

3.1 Le tre linee guida di Rusty al masquerading

Si assume che la propria interfaccia **esterna** si chiami 'ppp0'. Si usi ifconfig per scoprire qual è e si aggiusti il tutto a proprio gusto.

```
# ipchains -P forward DENY
# ipchains -A forward -i ppp0 -j MASQ
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

3.2 Promozione gratuita: WatchGuard

I firewall si possono anche comprare. Uno eccellente è il FireBox della WatchGuard. È eccellente perché mi piace, è sicuro, è basato su Linux e anche perché quelli della WatchGuard hanno finanziato il mantenimento di ipchains e del nuovo codice di firewall (per il 2.4). In breve, la WatchGuard mi sta pagando da mangiare mentre lavoro per voi. Quindi vi invito a considerare i loro prodotti.

http://www.watchguard.com `http://www.watchguard.com`

3.3 Configurazioni comuni di firewall

Supponiamo di amministrare il dominio littlecorp.com. Si ha una rete interna e una sola connessione in dialup (PPP) verso Internet (firewall.littlecorp.com che è 1.2.3.4). Si usa Ethernet nella propria rete locale e la propria macchina personale si chiama “myhost”.

Questa sezione illustrerà diversi arrangiamenti comuni. Li si legga attentamente, perché sono subdolamente diversi uno dall’altro.

3.3.1 Rete privata: proxy tradizionale

In questo scenario, i pacchetti provenienti dalla rete privata non traverseranno mai Internet e viceversa. Gli indirizzi IP della rete privata dovranno essere assegnati secondo le Address Allocation for Private Internets RFC1918 (ie. 10.*.*.*, 172.16.*.* o 192.168.*.*).

Il solo modo nel quale ci si potrà mai connettere a Internet è di connettersi al firewall, che è la sola macchina in entrambe le reti che può uscire dalla rete locale. Per far questo si può eseguire un programma (nel firewall) detto proxy (ci sono proxy per FTP, accesso al web, telnet, RealAudio, Usenet News e altri servizi). Si veda il Firewall HOWTO.

Qualsiasi servizio che si vuole sia accessibile da Internet deve essere nel firewall (si veda anche 3.3.5 (Servizi interni limitati) nel seguito).

Esempio: Permettere l’accesso web dalla rete privata a Internet

1. Alla rete privata sono assegnati gli indirizzi 192.168.1.*, e in particolare a myhost è assegnato il 192.168.1.100 e all’interfaccia Ethernet del firewall il 192.168.1.1.
2. Nel firewall è installato e configurato un proxy web (eg. “squid”), ed è in funzione sulla porta 8080.
3. Il Netscape sulla rete privata è configurato per usare la porta 8080 del firewall come proxy.
4. Nel rete privata non serve sia configurato il DNS.
5. Il DNS deve essere configurato sul firewall.
6. Nella rete privata non dev’essere configurato nessun instradamento predefinito (gateway).

Netscape su myhost legge http://slashdot.org.

1. Netscape si connette alla porta 8080 del firewall usando la porta 1050 su myhost. Chiede la pagina web di “http://slashdot.org”.
2. Il proxy cerca il nome “slashdot.org” e ottiene 207.218.152.131. Apre una connessione con quell’indirizzo IP (usando la porta 1025 sull’interfaccia esterna del firewall) e chiede al server web (porta 80) la pagina web.

3. Come il proxy riceve la pagina web dalla sua connessione con il server web, copia i dati nella connessione del Netscape.
4. Netscape mostra la pagina.

Dal punto di vista di slashdot.org, la connessione è fatta dalla porta 1025 di 1.2.3.4 (l'interfaccia PPP del firewall) verso la porta 80 di 207.218.152.131 (slashdot.org). Dal punto di vista di myhost, è fatta dalla porta 1050 di 192.168.1.100 (myhost) verso la porta 8080 di 192.168.1.1 (l'interfaccia Ethernet del firewall).

3.3.2 Rete privata: proxy trasparente

In questo scenario i pacchetti dalla rete privata non passano mai su Internet e viceversa. Gli indirizzi IP della rete privata dovranno essere assegnati secondo le Address Allocation for Private Internets RFC1918 (ie. 10.*.*.*, 172.16.*.* o 192.168.*.*).

Il solo modo nel quale ci si potrà mai connettere a Internet è di connettersi al firewall, che è la sola macchina in entrambe le reti che può uscire dalla rete locale. Per far questo si può eseguire un programma (nel firewall) detto proxy trasparente; il kernel invia i pacchetti al proxy trasparente invece di farli uscire (ie. imbastardisce l'instradamento).

Il proxy fatto in modo trasparente significa che ai client non serve sapere che c'è un proxy coinvolto.

Qualsiasi servizio che si vuole sia accessibile da Internet deve essere nel firewall (si veda anche 3.3.5 (Servizi interni limitati) nel seguito).

Esempio: Permettere l'accesso web dalla rete privata a Internet

1. Alla rete privata sono assegnati gli indirizzi 192.168.1.*, e in particolare a myhost è assegnato il 192.168.1.100 e all'interfaccia Ethernet del firewall il 192.168.1.1.
2. Nell firewall è installato un proxy web trasparente (credo esistano delle patch per squid per permettergli di funzionare in questo modo, oppure si usi transproxy) ed è in funzione sulla porta 8080.
3. Usando ipchains al kernel è detto di redirigere al proxy le connessioni dirette alla porta 80.
4. Netscape nella rete privata è configurato per connettersi direttamente.
5. Nella rete privata dev'essere configurato il DNS (ie. si deve eseguire anche un server DNS oltre al proxy sul firewall).
6. Nella rete privata dev'essere configurato l'instradamento predefinito (gateway), in modo da inviare i pacchetti al firewall.

Netscape su myhost legge http://slashdot.org.

1. Netscape cerca il nome slashdot.org e ottiene 207.218.152.131. Allora apre una connessione verso quell'indirizzo IP usando la porta locale 1050 e chiede la pagina web al server web (porta 80).
2. Come i pacchetti da myhost (porta 1050) verso slashdot.org (porta 80) passano attraverso il firewall, sono rediretti al proxy trasparente in attesa sulla porta 8080. Il proxy trasparente apre una connessione (usando la porta locale 1025) verso la porta 80 di 207.218.152.131 (che è dove stavano andando i pacchetti originali).
3. Come il proxy riceve la pagina web dalla sua connessione con il server web, copia i dati nella connessione del Netscape.
4. Netscape mostra la pagina.

Dal punto di vista di slashdot.org, la connessione è fatta dalla porta 1025 di 1.2.3.4 (l'interfaccia PPP del firewall) verso la porta 80 di 207.218.152.131 (slashdot.org). Dal punto di vista di myhost, è fatta dalla porta 1050 di 192.168.1.100 (myhost) verso la porta 80 di 207.218.152.131, ma in realtà sta dialogando con il proxy trasparente.

3.3.3 Rete privata: masquerading

In questo scenario i pacchetti dalla rete privata non passano mai su Internet e viceversa. Gli indirizzi IP della rete privata dovranno essere assegnati secondo le Address Allocation for Private Internets RFC1918 (ie. 10.*.*.*, 172.16.*.* o 192.168.*.*).

Invece di usare un proxy, usiamo una speciale caratteristica del kernel detta "masquerading" (mascheramento, travestimento). Il masquerading riscrive i pacchetti non appena passano per il firewall, così sembra sempre che provengano dal firewall stesso. Poi riscrive le risposte in modo che sembri provengano dalle fonti originali.

Il masquerading ha moduli separati per gestire protocolli "banali", come FTP, RealAudio, Quake, ecc. Per protocolli veramenente difficili da gestire, la funzione di "auto forwarding" (inoltro automatico) ne può gestire alcuni automaticamente impostando il port forwarding per l'insieme delle porte a loro relative: si veda "ipportfw" (kernel 2.0) o "ipmasqadm" (kernel 2.1).

Qualsiasi servizio che si vuole sia accessibile da Internet deve essere nel firewall (si veda anche 3.3.5 (Servizi interni limitati) nel seguito).

Esempio: Permettere l'accesso web dalla rete privata a Internet

1. Alla rete privata sono assegnati gli indirizzi 192.168.1.*, e in particolare a myhost è assegnato il 192.168.1.100 e all'interfaccia Ethernet del firewall il 192.168.1.1.

2. Il firewall è impostato per "mascherare" qualsiasi pacchetto proveniente dalla rete privata e destinato alla porta 80 di un host Internet.

3. Netscape è configurato per connettersi direttamente.

4. Nella rete privata dev'essere configurato correttamente il DNS.

5. Il firewall dovrà essere l'instradamento predefinito (gateway) per la rete privata.

Netscape su myhost legge http://slashdot.org.

1. Netscape cerca il nome "slashdot.org", e ottiene 207.218.152.131. Allora apre una connessione verso quell'indirizzo IP usando la porta locale 1050 e chiede la pagina web al server web (porta 80).

2. Come i pacchetti da myhost (porta 1050) verso slashdot.org (porta 80) passano attraverso il firewall, sono riscritti per apparire provenienti dall'interfaccia PPP del firewall (porta 65000). Il firewall ha un indirizzo Internet valido (1.2.3.4) e quindi i pacchetti di risposta provenienti da slashdot.org vengono correttamente instradati indietro.

3. Come i pacchetti da slashdot.org (port 80) verso firewall.littlecorp.com (porta 65000) arrivano, sono riscritti per andare alla porta 1050 di myhost. Questa è la vera magia del masquerading: si ricorda quando riscrive pacchetti uscenti e così può riscriverli non appena arrivano le risposte.

4. Netscape mostra la pagina.

Dal punto di vista di slashdot.org, la connessione avviene dalla porta 65000 di 1.2.3.4 (l'interfaccia PPP del firewall) verso la porta 80 di 207.218.152.131 (slashdot.org). Dal punto di vista di myhost, la connessione è fatta dalla porta 1050 di 192.168.1.100 (myhost), verso la porta 80 di 207.218.152.131 (slashdot.org).

3.3.4 Rete pubblica

In questo scenario la propria rete personale è parte di Internet: i pacchetti possono fluire senza modifiche tra le due reti. Gli indirizzi IP della rete interna devono essere assegnati richiedendo un blocco di indirizzi IP, in modo che il resto della rete sappia come raggiungerla. Ciò implica una connessione permanente.

In tal caso, il filtraggio dei pacchetti è usato per decidere quali pacchetti possono essere inoltrati tra la propria rete e il resto di Internet, eg. per restringere il resto di Internet al solo accesso ai propri server web interni.

Esempio: Permettere l'accesso web dalla rete privata verso Internet.

1. Alla propria rete interna è assegnato un blocco di indirizzi IP che si è richiesto (diciamo 1.2.3.*).
2. Il firewall è impostato per permettere tutto il traffico.
3. Netscape è configurato per connettersi direttamente.
4. Nella propria rete dev'essere configurato il DNS.
5. Il firewall dovrà essere l'instradamento predefinito (gateway) per la rete privata.

Netscape su myhost legge http://slashdot.org.

1. Netscape cerca il nome slashdot.org e ottiene 207.218.152.131. Apre poi una connessione verso quel indirizzo IP usando la porta locale 1050 e chiede la pagina al server web (porta 80).
2. I pacchetti passano attraverso il proprio firewall, proprio come passano attraverso diversi altri router tra myhost e slashdot.org.
3. Netscape mostra la pagina.

C'è solo una connessione: dalla porta 1050 di 1.2.3.100 (myhost) verso la porta 80 di 207.218.152.131 (slashdot.org).

3.3.5 Servizi interni limitati

Ci sono un po' di trucchi che si possono usare per permettere a Internet di accedere ai propri servizi interni, piuttosto che far girare i servizi sul firewall. Funzioneranno solo con un approccio basato su proxy o masquerading per le connessioni esterne.

L'approccio più semplice è di usare un "dirottatore" (redirector) che altro non è se non un proxy dei poveri, che attende la connessione su una data porta e poi apre una connessione a una porta fissa di un host interno copiando i dati tra le due connessioni. Un esempio è il programma "redir". Dal punto di vista di Internet, la connessione è fatta verso firewall. Dal punto di vista del server interno la connessione è fatta dall'interfaccia interna del firewall al server.

Un altro approccio (che richiede un kernel 2.0 con la patch per ipportfw, oppure un kernel 2.1 o più recente) è di usare il port forwarding del kernel. Quest'ultimo fa lo stesso lavoro di "redir" ma in modo diverso: il kernel riscrive i pacchetti mentre passano, cambiando il loro indirizzo di destinazione e la porta per indirizzarli verso l'host interno e la relativa porta. Dal punto di vista di Internet, la connessione è fatta verso il firewall. Dal punto di vista del server interno è fatta una connessione diretta tra l'host Internet e il server.

3.4 Ulteriori informazioni sul masquerading

David Ranch ha scritto un eccellente nuovo HOWTO sul masquerading, che ha parecchi argomenti in comune con questo HOWTO. Attualmente lo si può trovare a

http://www.linuxdoc.org/HOWTO/IP-Masquerade-HOWTO.html

L'home page ufficiale del Masquerading è a

http://ipmasq.cjb.net `http://ipmasq.cjb.net`

4 IP Firewall Chains

Questa sezione descrive tutto quello che bisogna veramente sapere per costruire un filtro di pacchetti che incontri le proprie esigenze.

4.1 Come passano i pacchetti attraverso i filtri

Il kernel parte con tre elenchi di regole; questi elenchi sono detti **firewall chains** (catene firewall) o semplicemente **chains** (catene). Le tre catene predefinite sono chiamate rispettivamente **input** (ingresso), **output** (uscita) e **forward** (inoltro). Quando arriva un pacchetto (diciamo, attraverso la scheda Ethernet) il kernel usa la catena `input` per decidere il suo destino. Se sopravvive a quel passo, allora il kernel decide dove mandare sucessivamente il pacchetto (ciò è detto **routing** (instradamento)). Se è destinato a un'altra macchina, consulta la catena `forward`. Alla fine, appena prima che il pacchetto esca, il kernel consulta la catena `output`.

Una catena è una lista di **regole**. Ogni regola dice «se l'intestazione del pacchetto è fatta così, allora questo è quello che si deve fare con il pacchetto». Se il pacchetto non soddisfa (verifica) una regola, allora viene consultata la successiva regola nella catena. Alla fine, se non ci sono altre regole da consultare, il kernel guarda la **policy** (tattica) della catena per decidere cosa fare. In un sistema conscio delle problematiche di sicurezza, questa tattica dice al kernel se rifiutare o proibire il pacchetto.

Per gli amanti dell'arte ASCII, questa figura mostra il percorso completo di un pacchetto entrante in una macchina.

```
        ----------------------------------------------------------------
        |            ACCEPT/                              lo interface |
        v           REDIRECT                  _______                  |
--> C --> S --> ______ --> D --> ~~~~~~~~ -->|forward|----> _______ -->
    h     a    |input |    e    {Routing }   |Chain  |     |output |ACCEPT
    e     n    |Chain |    m    {Decision}   |_______| --->|Chain  |
    c     i    |______|    a     ~~~~~~~~        |     | ->|_______|
    k     t       |        s        |            |     | |     |
    s     y       |        q        |            v     | |     |
    u     |       v        e        v           DENY/  | |     v
    m     |     DENY/      r   Local Process   REJECT  | |   DENY/
    |     v    REJECT      a        |                  | |  REJECT
    |   DENY               d        --------------------- |
    v                      e -----------------------------
   DENY
```

Di seguito una descrizione di ogni stadio:

Checksum:

Questo è il test per verificare che il pacchetto non sia in qualche modo corrotto. Se lo è, è rifiutato.

Sanity:

In realtà c'è una verifica di integrità del pacchetto prima di ogni catena firewall, ma quello della catena input è il più importante. Alcuni pacchetti malformati possono confondere il codice per il controllo delle regole, e quindi sono qui rifiutati (se ciò avviene è stampato un messaggio nel syslog).

input chain:

È la prima catena firewall contro la quale viene testato il pacchetto. Se il verdetto della catena non è `DENY` o `REJECT`, il pacchetto prosegue.

Demasquerade:

Se il pacchetto è una risposta (reply) a un precedente pacchetto "mascherato", è "demascherato" e passa direttamente alla catena `output`. Se non si usa L'IP Masquerading si cancelli questo percorso dal diagramma.

Routing decision:

Il codice di instradamento esamina il campo di destinazione per decidere se questo pacchetto deve andare a un processo locale (si veda Local process qui sotto) o inoltrato a una macchina remota (si veda forward chain più avanti)

Local process:

Un processo in esecuzione sulla macchina può ricevere pacchetti dopo il passo di Routing Decision, e può inviare pacchetti (che passano per il passo Routing Decision, e poi attraversano la catena output).

lo interface:

Se i pacchetti provenienti da un processo locale sono destinati a un processo locale, attraverseranno la catena output con il campo interfaccia impostato a 'lo', e poi ritornano attraverso la catena input sempre con interfaccia impostato a 'lo'. L'interfaccia lo è solitamente chiamata interfaccia loopback.

local:

Se il pacchetto non era stato creato da un processo locale, allora è esaminata la catena forward, altrimenti il pacchetto va verso la catena output.

forward chain:

Questa catena è attraversata da qualsiasi pacchetto che provi a passare attraverso questa macchina per andare verso un'altra.

output chain:

Questa catena è attraversata da qualsiasi pacchetto un attimo prima di essere spedito fuori.

4.1.1 Usare ipchains

Per prima cosa si controlli di avere la versione di ipchains a cui fa riferimento questo documento:

```
$ ipchains --version
ipchains 1.3.9, 17-Mar-1999
```

Si noti che io raccomando la 1.3.4 (che non ha le opzioni lunghe, come '–sport'), o la 1.3.8 o superiori; queste sono molto stabili.

ipchains ha una pagina man piuttosto dettagliata (`man ipchains`), e se servono ulteriori dettagli o particolari, si può dare un'occhiata all'interfaccia di programmazione (`man 4 ipfw`), oppure al file `net/ipv4/ip_fw.c` nei sorgenti dei kernel 2.1.x, che sono (ovviamente) le fonti più autorevoli.

Nel pacchetto sorgente c'è pure una eccellente scheda di riferimento rapido di Scott Bronson, in PostScript(TM) sia in formato A4 che US Letter.

Ci sono parecchie cose diverse che si possono fare con `ipchains`. Per prima cosa le operazioni per gestire intere catene. Si parte con tre catene predefinite `input`, `output` e `forward` che non possono essere cancellate.

1. Creare una nuova catena (-N).
2. Cancellare una catena vuota (-X).
3. Cambiare la tattica per una catena predefinita (-P).
4. Elencare le regole in una catena (-L).
5. Svuotare una catena delle sue regole (-F).
6. Azzerare i contatori di pacchetti e byte per tutte le regole in una catena (-Z).

Ci sono diversi modi per manipolare le regole in una catena:

1. Aggiungere una nuova regola a una catena (-A).
2. Inserire in una posizione specifica una nuova regola in una catena (-I).
3. Rimpiazzare una regola in una qualche posizione in una catena (-R).
4. Cancellare da una posizione specifica una regola da una catena (-D).
5. Cancellare da una catena la prima regola corrispondente (-D).

Ci sono alcune operazioni per il masquerading, che, in mancanza di un posto migliore dove metterle, sono in `ipchains`:

1. Elenca le connessioni attualmente mascherate (-M -L).
2. Imposta i valori di timeout del masquerading (-M -S) (si veda anche 6.10 (Non posso impostare i timeout del masquerading!)).

La funzione finale (e forse la più utile) permette di controllare cosa succederebbe a un dato pacchetto se volesse traversare una data catena.

4.1.2 Cosa si vedrà quando il proprio computer viene avviato

Prima che qualsiasi comando ipchains sia eseguito (attenzione: alcune distribuzioni lanciano ipchains nei loro script di inizializzazione), non ci sarà alcuna regola in alcuna delle catene predefinite ('input', 'forward' e 'output'), e ognuna delle catene avrà la tattica ACCEPT. Questa è la massima apertura che si può avere.

4.1.3 Operazioni su una sola regola

Questa è la ragione di sussistenza di ipchains: la manipolazione delle regole. Molto probabilmente si useranno i comandi di aggiunta (-A) e cancellazione (-D). Gli altri (-I per l'inserimento e -R per il rimpiazzo) sono semplici estensioni di questi concetti.

Ogni regola specifica un insieme di condizioni che il pacchetto deve soddisfare, e cosa fare se il pacchetto le soddisfa (un "obiettivo"). Per esempio, si possono voler proibire tutti i pacchetti ICMP provenienti dall'indirizzo 127.0.0.1. Quindi in questo caso le nostre condizioni sono che il protocollo deve essere ICMP e che l'indirizzo di provenienza deve essere 127.0.0.1. Il nostro obiettivo è 'DENY'.

127.0.0.1 è l'interfaccia 'loopback', presente anche se non si possiede una vera connessione di rete. Si può usare il programma 'ping' per generare tali pacchetti (semplicemente invia un pacchetto ICMP di tipo 8 (echo request) al quale tutti gli host cooperativi dovrebbero rispondere obbligatoriamente con un pacchetto ICMP di tipo 0 (echo reply)). Ciò lo rende molto utile per i 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
# ipchains -A input -s 127.0.0.1 -p icmp -j DENY
# 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
#
```

Si può vedere che il primo ping ha successo (il '-c 1' dice a ping di inviare un solo pacchetto).

Poi si è aggiunta alla catena 'input' una regola che specifica che per i pacchetti provenienti da 127.0.0.1 ('-s 127.0.0.1') con protocollo ICMP ('-p ICMP') si dovrà saltare a DENY ('-j DENY').

Poi si è verificata la nostra regola, usando un secondo ping. Ci sarà una pausa prima che il programma si stanchi di aspettare una risposta che non arriverà mai.

Si può cancellare la regola in due modi. Per prima cosa, poiché sappiamo che è la sola regola nella catena input, possiamo usare la cancellazione in base alla posizione, come in

```
# ipchains -D input 1
#
```

per cancellare la regola numero 1 nella catena input.

Il secondo modo è di ricopiare il comando -A precedente, rimpiazzando -A con -D. Ciò è utile quando si ha una complessa catena di regole e non si vuole star lì a contarle per scoprire che quella che si vuole cancellare è la 37-esima. In tal caso useremo:

```
# ipchains -D input -s 127.0.0.1 -p icmp -j DENY
#
```

La sintassi di -D deve avere esattamente le stesse opzioni del comando -A (o -I oppure -R). Se c'è una multitudine di regole identiche nella stessa catena, è cancellata solo la prima.

4.1.4 Specificare il filtraggio

Si è visto l'uso di '-p' per specificare il protocollo e di '-s' per specificare l'indirizzo di provenienza, ma ci sono altre opzioni che si possono usare per specificare le caratteristiche del pacchetto. Quel che segue è un compendio esaustivo.

Specificare gli indirizzi di provenienza e destinazione Gli indirizzi IP di provenienza (-s) e destinazione (-d) possono essere specificati in quattro modi. Il modo più comune è di usare il nome completo, come 'localhost' o 'www.linuxhq.com'. Il secondo modo è di specificare l'indirizzo IP, come ad esempio '127.0.0.1'.

Il terzo e il quarto modo permettono di specificare un gruppo di indirizzi IP, come ad esempio '199.95.207.0/24' o '199.95.207.0/255.255.255.0'. Entrambi specificano un qualsiasi indirizzo IP tra 192.95.207.0 e 192.95.207.255 estremi inclusi; le cifre dopo '/' dicono quali parti dell'indirizzo IP sono significative. I valori predefiniti sono '/32' o '/255.255.255.255' (corrispondenti a tutti gli indirizzi IP). Per specificare nessun indirizzo IP può essere usato '/0', come segue:

```
# ipchains -A input -s 0/0 -j DENY
#
```

Questa cosa è raramente usata, in quanto il suo effetto è lo stesso che non specificare affatto l'opzione '-s'.

Specificare una negazione Molte opzioni, tra le quali '-s' e '-d', possono avere gli argomenti preceduti da '!' (pronunciato 'not') per effettuare la corrispondenza con indirizzi NON uguali a quelli dati. Per esempio, '-s ! localhost' corrisponde a qualsiasi pacchetto non proveniente da localhost.

Non si dimentichimo gli spazi attorno al '!': sono veramente necessari.

Specificare un protocollo Il protocollo può essere specificato con l'opzione '-p'. Il protocollo può essere un numero (se si conoscono i valori numerici dei protocolli IP) o un nome per i casi speciali di 'TCP', 'UDP' o 'ICMP'. Non è importante come è scritto: vanno bene sia 'tcp' che 'TCP'.

I nomi dei protocolli possono essere preceduti da un '!' per negarli, con ad esempio '-p ! TCP'.

Specificare porte UDP e TCP Nei casi particolari nei quali è specificato TCP o UDP come protocollo, ci può essere un argomento aggiuntivo che indica la porta TCP e UDP, oppure un intervallo (inclusivo) di porte (si veda anche 4.1.4 (Gestire i frammenti) nel seguito). Un intervallo è rappresentato usando un carattere ':', come ad esempio '6000:6010', che copre 11 numeri di porte da 6000 a 6010 estremi compresi. Se è omesso il limite inferiore, il valore predefinito è 0. Se è omesso quello superiore, il suo valore predefinito è 65535. Quindi per specificare le connessioni TCP provenienti da porte inferiori alla 1024, la sintassi potrebbe essere '-p TCP -s 0.0.0.0/0 :1023'. I numeri di porta possono essere specificati anche attraverso il nome, eg. 'www'.

Si noti che la specificazione può essere preceduta da un '!', che la nega. Quindi per specificare qualsiasi pacchetto TCP TRANNE i pacchetti WWW, si potrebbe specificare

```
-p TCP -d 0.0.0.0/0 ! www
```

È importante realizzare che la specifica

```
-p TCP -d ! 192.168.1.1 www
```

è molto diversa da

```
-p TCP -d 192.168.1.1 ! www
```

La prima specifica qualsiasi pacchetto TCP verso la porta WWW su qualsiasi macchina tranne la 192.168.1.1. La seconda specifica qualsiasi connessione TCP verso qualsiasi porta di 192.168.1.1 tranne la porta WWW.

Per finire, questo caso indica di escludere sia la porta WWW e che 192.168.1.1:

```
-p TCP -d ! 192.168.1.1 ! www
```

Specificare tipo e codice ICMP Anche ICMP permette argomenti opzionali, ma poiché ICMP non ha le porte (ICMP ha un **tipo** e un **codice**) hanno un significato diverso.

Possono essere specificati come nomi ICMP (si usi `ipchains -h icmp` per la lista di nomi) dopo l'opzione '-s', oppure come tipo e codice ICMP numerici, dove il tipo segue l'opzione '-s' e il codice segue l'opzione '-d'.

I nomi ICMP sono piuttosto lunghi: basta usare solamente abbastanza lettere da rendere il nome distinguibile da ogni altro.

Di seguito una piccola tabella dei più comuni pacchetti ICMP:

```
Numero  Nome                     Richiesto da

0       echo-reply               ping
3       destination-unreachable  qualsiasi traffico TCP/UDP.
5       redirect                 per l'instradamento, se non sta girando
                                 il demone di instradamento
8       echo-request             ping
11      time-exceeded            traceroute
```

Si noti che i nomi ICMP al momento non possono essere preceduti da '!'.

NON NON NON si blocchino tutti i messaggi ICMP di tipo 3! (si veda 5.2.1 (Pacchetti ICMP) più avanti).

Specificare un'interfaccia L'opzione '-i' specifica il nome di un'**interfaccia**. Un'interfaccia è il dispositivo fisico dal quale entra o esce il pacchetto. Si può usare il comando `ifconfig` per avere un elenco delle interfacce che al momento sono 'su' (ie. che al momento funzionano).

L'interfaccia per i pacchetti in arrivo (ie. i pacchetti che stanno attraversando la catena `input`) è considerata essere l'interfaccia dalla quale entrano. Logicamente, l'interfaccia per i pacchetti in partenza (i pacchetti che stanno attraversando la catena `output`) è l'interfaccia dalla quale usciranno. Anche l'interfaccia per i pacchetti che passano per la catena `forward` è l'interfaccia dalla quale usciranno; a me sembra una decisione piuttosto arbitraria.

È perfettamente legale specificare un'interfaccia che al momento non esiste; la regola non sarà mai soddisfatta finché l'interfaccia non viene attivata. Ciò è estremamente utile per le connessioni PPP in dial-up (solitamente l'interfaccia `ppp0`) e simili.

Come caso speciale, un nome di intefaccia che termina con un '+' corrisponderà a tutte le interfacce (che esistano o meno) che iniziano con quella stringa. Per esempio, per specificare una regola che corrisponda a tutte le interfacce PPP, può essere usata 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.

Specificare solo pacchetti TCP SYN Talvolta è utile permettere connessioni TCP in una direzione, ma non nell'altra. Per esempio, si può voler permettere connessioni verso un server WWW esterno ma non le connessioni da quel server.

L'approccio naive sarebbe quello di bloccare i pacchetti TCP provenienti da server. Sfortunantamente, le connessioni TCP richiedono per funzionare che i pacchetti possono andare in entrambe le direzioni.

La soluzione è di bloccare solo i pacchetti usati per richiedere una connessione. Questi pacchetti sono chiamati pacchetti **SYN** (ok, tecnicamente sono pacchetti con il flag SYN impostato, e i campi FIN e ACK non attivati, ma li chiameremo lo stesso pacchetti SYN). Non permettendo solamente questi pacchetti, si possono fermare i tentativi di connessione.

Per questo viene usata l'opzione '-y': è valida solamente per regole che specificano TCP come loro protocollo. Per esempio, per specificare un tentativo di connessione TCP da 192.168.1.1:

```
-p TCP -s 192.168.1.1 -y
```

Ancora una volta, questa opzione può essere negata precedendola con un '!' che significa tutti pacchetti tranni quelli per iniziare la connessione.

Gestire i frammenti Qualche volta un pacchetto è troppo grande per passarci tutto intero nel cavo. Quando questo succede, il pacchetto è diviso in **frammenti** (fragments), e inviato come pacchetti multipli. L'altro capo della connessione riassembla i pacchetti per ricostruire il pacchetto intero.

Il problema con i frammenti è che alcune delle specificazioni suddette (in particolare porta d'origine, porta di destinazione, tipo ICMP, codice ICMP o flag SYN TCP) richiedono che il kernel sbirci nella parte iniziale del pacchetto, che è contenuta solo nel primo frammento.

Se le propria macchina è la sola connessione a una rete esterna, allora si può dire al kernel di riassemblare tutti i frammenti che gli passano attraverso, compilando il kernel con l'opzione `IP: always defragment` posta a 'Y'. Ciò evita il problema in maniera pulita.

Diversamente, è importante capire come sono trattati i frammenti dalle regole di filtraggio. Qualsiasi regola di filtraggio che chiede informazioni che non si hanno *non* verrà soddisfatta. Ciò significa che il primo frammento è trattato come qualsiasi altro pacchetto. Non lo saranno invece il secondo e i successivi frammenti. Quindi una regola `-p TCP -s 192.168.1.1 www` (specificante una porta sorgente 'www') non sarà mai soddisfatta da un frammento (tranne al primo). E neppure la regola opposta `-p TCP -s 192.168.1.1 ! www`.

Comunque, si può specificare una regola specifica per il secondo e i successivi frammenti, usando l'opzione '-f'. Ovviamente in questa regola per i frammenti è illegale specificare una porta TCP o UDP, un tipo o un codice ICMP oppure l'opzione per il SYN TCP.

È pure legale specificare che una regola *non* si applica al secondo e ai successivi frammenti, facendo precedere '-f' da '!'.

Solitamente è considerata una cosa sicura lasciar passare il secondo e i successi frammenti, poiché il filtraggio avrà effetto sul primo frammento e quindi impedirà la ricostruzione nell'host di destinazione. Comunque, sono noti alcuni bug che permettono il crash di macchine semplicemente inviandogli frammenti.

Una nota per i capoccioni della rete: i pacchetti malformati (pacchetti TCP, UDP o ICMP troppo corti affinché il codice di firewalling possa leggere le porte o il codice e tipo ICMP) sono trattati come frammenti. Solo i frammenti TCP che iniziano alla posizione 8 sono esplicitamente scartati dal codice di firewall (dovrebbe apparire un messaggio nel syslog se succede questo).

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

```
# ipchains -A output -f -d 192.168.1.1 -j DENY
#
```

4.1.5 Effetti collaterali del filtraggio

OK, quindi ora si conoscono tutti i modi con i quali si può creare una regola che corrisponda a un pacchetto. Se un pacchetto soddisfa una regola, succedono le seguenti cose:

1. Il contatore di byte per quella regola è incrementato della dimensione del pacchetto (intestazione e tutto il resto).
2. È incrementato il contatore di pacchetti per quella regola.
3. Se la regola lo richiede, il pacchetto è registrato.
4. Se la regola lo richiede, è cambiato il campo Type Of Service del pacchetto.
5. Se la regola lo richiede, il pacchetto è marcato (non nei kernel della serie 2.0).
6. È esaminata la regola obiettivo per decidere cosa fare dopo al pacchetto.

Per varietà, le esaminerò in ordine di importanza.

Specificare un obiettivo Un **obiettivo** dice al kernel cosa farne di un pacchetto che soddisfa una regola. ipchains usa '-j' (penso 'jump-to' – salta a) per la specifica dell'obiettivo. Il nome dell'obiettivo deve essere più corto di 8 caratteri, ed è importante come è scritto: RETURN e return sono completamente diversi.

Il caso più semplice è quando non è specificato alcun obiettivo. Questo tipo di regola (spesso detta regola di 'accounting' – contabilità) è utile con contare semplicemente un certo tipo di pacchetto. Che questa regola sia o meno soddisfatta, semplicemente il kernel esamina la regola successiva nella catena. Per esempio, per contare il numero di pacchetti da 192.168.1.1, si può fare così:

```
# ipchains -A input -s 192.168.1.1
#
```

(Usando 'ipchains -L -v' si possono vedere i contatori di byte e pacchetti associati con ciascuna regola).

Esistono sei obiettivi speciali. I primi tre, `ACCEPT`, `REJECT` e `DENY` sono piuttosto semplici. `ACCEPT` permette che il pacchetto passi. `DENY` scarta il pacchetto come non fosse mai stato ricevuto. `REJECT` scarta il pacchetto, ma (se non è un pacchetto ICMP) genera una risposta ICMP per la sorgente per dirle che la destinazione non è raggiungibile.

Quella dopo, `MASQ` dice al kernel di mascherare il pacchetto. Affinché ciò funzioni, il proprio kernel dev'essere compilato con il supporto per l'IP Masquerading. Per i dettagli si veda il Masquerading-HOWTO e l'appendice 8 (Differenze tra ipchains e ipfwadm). Questo obiettivo è valido solo per pacchetti che attraversano la catena `forward`.

L'altro obiettivo principale è `REDIRECT` che dice al kernel di inviare un pacchetto a una porta locale invece che alla sua destinazione. Può essere specificato solo per regole che specificano TCP o UDP come loro protocolli. Opzionalmente, può essere specificata una porta (nome o numero) dopo di '-j REDIRECT' che farà sì che quel pacchetto sia dirottato a quella particolare porta anche se era indirizzato a un'altra porta. Questo obiettivo è valido solo per pacchetti che attraversano la catena `input`.

L'obiettivo finale speciale è `RETURN` che è identico all'uscita immediata dalla catena (si veda 4.1.5 (Impostare la tattica) nel seguito).

Qualsiasi altro obiettivo indica una catena definita dall'utente (come descritto in 4.1.5 (Operazione con intere catene) nel seguito). Il pacchetto inizierà ad attraversare le regole in quella catena. Se quella catena non decide il destino del pacchetto, allora, una volta che è terminata la traversata della catena, si riprende dalla regola successiva nella catena corrente.

È ora di un altro po' arte ASCII. Si considerino due catene (sciocche): `input` (la catena predefinita) e `Test` (una catena definita dall'utente).

```
 ‘input’                           ‘Test’
------------------------------  ------------------------------
| Regola1: -p ICMP -j REJECT |  | Regola1: -s 192.168.1.1    |
|----------------------------|  |----------------------------|
| Regola2: -p TCP -j Test    |  | Regola2: -d 192.168.1.1    |
|----------------------------|  ------------------------------
| Regola3: -p UDP -j DENY    |
------------------------------
```

Si consideri un pacchetto TCP proveniente da 192.168.1.1 e destinato a 1.2.3.4. Entra nella catena `input` e viene controllato rispetto a Regola1: non la soddisfa. Soddisfa invece Regola2 il cui obiettivo è `Test`. Quindi la successiva regola a essere esaminata è la prima di `Test`. Regola1 in `Test` è soddisfatta ma non specifica un obiettivo, quindi Regola2 è la prossima a essere esaminata. Questa non è soddisfatta e si è così raggiunta la fine delle catena. Si quindi ritorna alla catena `input`, dove si è appena esaminata Regola2 e quindi ora si esamina Regola3, che non viene soddisfatta.

Quindi il percorso del pacchetto è:

```
                          v    ___________________________
 ‘input’                  |   /  ‘Test’                  v
--------------------------|--/  -------------------------|----
| Regola1                 | /|  | Regola1                |   |
|-------------------------|/-|  |------------------------|---|
| Regola2                 /  |  | Regola2                |   |
|----------------------------|  -------------------------v----
| Regola3                 /--+___________________________/
--------------------------|---
                          v
```

Si veda la sezione 5.1 (Come organizzare le proprie regole di firewall) per alcuni metodi per usare efficacemente le catene definite dall'utente.

Registrazione dei pacchetti Questo è un effetto collaterale che può avere una regola che viene soddisfatta: usando l'opzione '-l' si può far sì che il pacchetto che la soddisfa sia registrato. Solitamente questa cosa non la si vuole per i pacchetti di routine, ma è una caratteristica utile se si vogliono cercare eventi eccezionali.

Il kernel registra questa informazione come segue:

```
Packet log: input DENY eth0 PROTO=17 192.168.2.1:53 192.168.1.1:1025
  L=34 S=0x00 I=18 F=0x0000 T=254
```

Questo messaggio di registrazione è pensato per essere conciso e contiene informazioni tecniche utili solo ai guru del networking, ma può essere utile anche al resto di noi. Si suddivide così:

1. 'input' è la catena che contiene la regola soddisfatta dal pacchetto, che ha causato il messaggio di log.
2. 'DENY' è quanto la regola dice di fare del pacchetto. Se è '-' allora la regola non ha effetto sul pacchetto (una regola di accounting).
3. 'eth0' è il nome dell'interfaccia. Poiché la catena era la input, indica che il pacchetto è entrato da 'eth0'.
4. 'PROTO=17' indica che il pacchetto era con protocollo 17. Un elenco dei protocolli è dato in '/etc/protocols'. I più comuni sono 1 (ICMP), 6 (TCP) e 17 (UDP).
5. '192.168.2.1' indica che l'indirizzo IP di provenienza era 192.168.2.1.
6. ':53' indica che porta di provenienza era la porta 53. Cercando in '/etc/services' si veda che questa è la porta 'domain' (ie. probabilmente è una risposta DNS). Per UDP e TCP questo numero è la porta di provenienza. Per ICMP, è il tipo ICMP. Per gli altri, sarà 65535.
7. '192.168.1.1' è l'indirizzo IP di destinazione.
8. ':1025' indica che la porta di destinazione era la 1025. Per UDP e TCP questo numero è la porta di destinazione. Per ICMP è il codice ICMP. Per gli altri, sarà 65535.
9. 'L=34' indica che il pacchetto era lungo 34 byte.
10. 'S=0x00' indica il campo Type of Service (lo si divida per 4 per ottenere il Type of Service usato da ipchains).
11. 'I=18' è l'ID IP.
12. 'F=0x0000' è l'offset a 16 bit del frammento più i flag. Un valore che cominci con '0x4' o '0x5' indica che il bit 'Don't Fragment' è impostato. '0x2' or '0x3' indicano che è impostato il bit 'More Fragments'; sono da aspettarsi altri frammenti dopo di questo. Il resto del numero è l'offset, diviso per 8, di questo frammento.
13. 'T=254' è il 'Time To Live' (tempo di vita) del pacchetto. È diminuito di uno a ogni hop, e solitamente parte a 15 o 255.
14. '(#5)' ci può essere un numero finale tra parentesi nei kernel più recenti (forse dopo il 2.2.9). È il numero della regola che ha causato la registrazione del pacchetto.

Nei sistema Linux standard, questo output del kernel è catturato da klogd (il demone di registrazione del kernel) che lo passa poi al syslogd (il demone di registrazione di sistema). Il file '/etc/syslog.conf' controlla il comportamento di syslogd, specificando una destinazione per ogni 'facility' (nel nostro caso la facility è kernel) e 'livello' (per ipchains, il livello usato è "info").

Per esempio, il mio /etc/syslog.conf (Debian) contiene due righe corrispondenti a 'kern.info':

```
kern.*                          -/var/log/kern.log
*.=info;*.=notice;*.=warn;\
        auth,authpriv.none;\
        cron,daemon.none;\
        mail,news.none          -/var/log/messages
```

Indica che i messaggi sono duplicati in '/var/log/kern.log' e in '/var/log/messages'. Per maggiori dettagli, si veda 'man syslog.conf'.

Manipolare il 'Type Of Service' Nell'intestazione IP ci sono quattro bit raramente usati, detti bit **Type of Service** (TOS – Tipo Di Servizio). Influenzano il modo in cui sono trattati i pacchetti; i quattro bit sono "Minimum Delay" (Ritardo Minimo), "Maximum Throughput" (Massima Velocità di Trasmissione), "Maximum Reliability" (Massima Affidabilità) e "Minimum Cost" (Minimo Costo). Solo a uno di questo bit è permesso di essere impostato. Rob van Nieuwkerk, l'autore del codice di "maltrattamento" TOS, ne parla in questi termini:

> Per me è importante specialmente il "Minimum Delay". L'ho attivato per i pacchetti "interattivi" nel mio router (Linux) a monte. Io sono dietro una connessione modem a 33k6. Linux prioritizza i pacchetti in 3 code. In questo modo ottengo accettabili prestazioni interattive mentre faccio dei meri download (potrebbe andare ancora meglio se non ci fosse una coda così grande nel driver della seriale, ma ora la latenza è mantenuta sotto gli 1.5 secondi).

Nota: ovviamente, non si ha controllo sui pacchetti in arrivo; si può controllare la priorità solamente dei pacchetti che lasciano la propria macchina. Per negoziare le priorità con l'altro capo della connessione, deve essere usato un protocollo tipo RSVP (non so niente in proposito, quindi non chiedete a me).

L'uso più comune è di impostare le connessioni di controllo di telnet e ftp a "Minimum Delay" e quelle dati FTP a "Maximum Throughput". Ciò può essere fatto come segue:

```
ipchains -A output -p tcp -d 0.0.0.0/0 telnet -t 0x01 0x10
ipchains -A output -p tcp -d 0.0.0.0/0 ftp -t 0x01 0x10
ipchains -A output -p tcp -s 0.0.0.0/0 ftp-data -t 0x01 0x08
```

L'opzione '-t' accetta altri due parametri addizionali, entrambi in esadecimale. Questi permettono di far giochetti complessi con i bit TOS: con la prima maschera è fatta l'AND con i TOS correnti del pacchetto, e poi del risultato viene fatta l'XOR con la seconda. Se è troppo confuso, allora si usi semplicemente la tabella seguente:

```
Nome del TOS            Valore          Uso Tipico

Minimum Delay           0x01 0x10       ftp, telnet
Maximum Throughput      0x01 0x08       ftp-data
Maximum Reliability     0x01 0x04       snmp
Minimum Cost            0x01 0x02       nntp
```

Andi Kleen puntualizza quanto segue:

> Forse potrebbe essere utile aggiungere riferimenti al paramentro txqueuelen di ifconfig alla discussione dei bit TOS. La lunghezza predefinita della coda del dispositivo, regolata per le schede ethernet, per i modem è troppo lunga e fa sì che lo schedulatore a 3 bande (le cui code sono basate sui TOS) funzioni in maniera subottima. È una buona idea impostarla a un valore tra 4 e 10 per le connessioni via modem o ISDN a canale b singolo. Questo è un problema dei kernel 2.0 e 2.1, ma mentre nei 2.1 esiste un'opzione di ifconfig (nei nettools recenti), nei 2.0 è necessaria una patch ai sorgenti dei device driver.

Quindi, per vedere i massimi benefici dalla manipolazione dei TOS nella connessioni PPP via modem, si usi 'ifconfig $1 txqueuelen' nel proprio script /etc/ppp/ip-up. Il numero da usare dipende dalla velocità del modem e dalla dimesione del buffer nel modem; Andi ci mostra ancora la direzione da seguire:

> Il miglior valore per una data configurazione si determina sperimentalmente. Se la coda è troppo corta in un router allora saranno scartati i pacchetti. Naturalmente si traggono benefici anche senza la riscrittura dei TOS, solo che la riscrittura dei TOS aiuta a dare beneficio ai programmi non cooperativi (tutti i programmi standard di Linux sono cooperativi).

Marcare un pacchetto Ciò permette una complessa e potente iterazione con la nuova implementazione di 'Quality of Service' di Alexey Kuznetsov e con il forwading basato sulla marcatura degli ultimi kernel della serie 2.1. Darò maggiori informazioni non appena ne verrò in possesso. Questa opzione è ignorata nei kernel della serie 2.0.

Operazioni su un'intera catena Una caratteristica molto utile di ipchains è la possibilità di raggruppare regole collegate dentro catene. Si possono chiamare le catene come si vuole a meno che il nome non sia in conflitto con le catene (`input`, `output` e `forward`) o gli obiettivi (`MASQ`, `REDIRECT`, `ACCEPT`, `DENY`, `REJECT` o `RETURN`) predefiniti. Suggerisco di evitare in toto l'uso di etichette in maiuscolo, in quanto le potrei usare per estensioni future. Il nome della catena può essere lungo fino a 8 caratteri.

Creare una nuova catena Suvvia creiamo una nuova catena! Poiché sono un tipo con un sacco di immaginazione, la chiamerò `test`.

```
# ipchains -N test
#
```

Tutto qua. Ora le si possono mettere dentro le regole come spiegato in precedenza.

Cancellare una catena Anche cancellare una catena è semplice.

```
# ipchains -X test
#
```

Perché '-X'? Beh, tutte le altre lettere buone erano già occupate.

Ci sono un paio di restrizioni sulla cancellazione di una catena: deve essere vuota (si veda 4.1.5 (Svuotare una catena) nel seguito) e non deve essre l'obiettivo di nessuna regola. Non è possibile cancellare nessuna delle tre catene predefinite.

Svuotare una catena C'è un modo semplice per svuotare una catena di tutte le regole, usando il comando '-F' (flush).

```
        # ipchains -F forward
        #
```

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

Elencare le regole in una catena Si possono elencare tutte le regole in una catena usando il comando '-L' (list).

```
# ipchains -L input
Chain input (refcnt = 1): (policy ACCEPT)
target     prot opt    source                destination           ports
ACCEPT     icmp -----  anywhere              anywhere              any
# ipchains -L test
Chain test (refcnt = 0):
target     prot opt    source                destination           ports
DENY       icmp -----  localnet/24           anywhere              any
#
```

Il valore di 'refcmt' mostrato per `test` è il numero di regole che hanno `test` come loro obiettivo. Deve essere zero (e la catena essere vuota) prima che si possa cancellarla.

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

Ci sono tre opzioni che possono accompagnare '-L'. L'opzione '-n' (numeric) è molto utile in quanto previene `ipchains` dal tentativo di ricercare gli indirizzi IP, che (se si usa un DNS come fanno molti) causerà parecchio ritardo se il proprio DNS non è configurato correttamente, o si sono filtrate tutte le richieste DNS. Inoltre fa sì che le porte siano mostrate come numeri piuttosto che con i loro nomi.

L'opzione '-v' mostra tutti i dettagli delle regole, come i contatori di pacchetti e byte, le maschere TOS, l'interfaccia e la marcatura dei pacchetti. Diversamente questi valori sono omessi. Per esempio:

```
# ipchains -v -L input
Chain input (refcnt = 1): (policy ACCEPT)
 pkts bytes target     prot opt   tosa tosx  ifname    mark       source                destination
   10   840 ACCEPT     icmp ----- 0xFF 0x00  lo                   anywhere              anywhere
```

Si noti che i contatori di pacchetti e byte sono mostrati usando i suffissi 'K', 'M' o 'G' rispettivamente per 1000, 1000000 e 1000000000. Usando l'opzione '-x' (espandi i numeri) verranno mostrati i numeri interi, senza preoccuparsi di quanto grandi siano.

Azzerare i contatori È utile poter azzerare i contatori. Ciò può essere fatto con l'opzione '-Z' (zero counters). Per esempio:

```
# ipchains -v -L input
Chain input (refcnt = 1): (policy ACCEPT)
 pkts bytes target     prot opt   tosa tosx  ifname    mark       source                destination
   10   840 ACCEPT     icmp ----- 0xFF 0x00  lo                   anywhere              anywhere
# ipchains -Z input
# ipchains -v -L input
Chain input (refcnt = 1): (policy ACCEPT)
 pkts bytes target     prot opt   tosa tosx  ifname    mark       source                destination
    0     0 ACCEPT     icmp ----- 0xFF 0x00  lo                   anywhere              anywhere
#
```

Il problema con questo approccio è che talvolta serve sapere il valore dei contatori un attimo prima di azzerarli. Nell'esempio precedente, tra i comandi '-L' e '-Z' potrebbero essere passati degli altri pacchetti. Per questa ragione, si possono usare '-L' e '-Z' *assieme*, per azzerare i contatori mentri li si legge. Sfortunatamente, se si fa così, non si può operare su una sola catena: si devono mostrare e azzerare tutte le catene in una volta.

```
# ipchains -L -v -Z
Chain input (policy ACCEPT):
 pkts bytes target     prot opt   tosa tosx  ifname    mark       source                destination
   10   840 ACCEPT     icmp ----- 0xFF 0x00  lo                   anywhere              anywhere

Chain forward (refcnt = 1): (policy ACCEPT)
Chain output (refcnt = 1): (policy ACCEPT)
Chain test (refcnt = 0):
    0     0 DENY       icmp ----- 0xFF 0x00  ppp0                 localnet/24           anywhere
# ipchains -L -v
Chain input (policy ACCEPT):
 pkts bytes target     prot opt   tosa tosx  ifname    mark       source                destination
```

```
   10   840 ACCEPT     icmp ----- 0xFF 0x00  lo                     anywhere             anywhere

Chain forward (refcnt = 1): (policy ACCEPT)
Chain output (refcnt = 1): (policy ACCEPT)
Chain test (refcnt = 0):
    0     0 DENY       icmp ----- 0xFF 0x00  ppp0                   localnet/24          anywhere
#
```

Impostare la tattica Si è visto cosa succede quando un pacchetto raggiunge la fine di una catena predefinita quando si è discusso come un pacchetto cammina attraverso una catena in 4.1.5 (Specificare un obiettivo). In questo caso, la **tattica** (policy) di una catena determina il destino del pacchetto. Solo le catene predefinite (`input`, `output` e `forward`) hanno delle tattiche, poiché se un pacchetto cade fuori dalla fine di una catena definita dall'utente, la traversata riprende nelle catena precedente.

La tattica può essere una qualsiasi dei primi quattro obiettivi speciali: `ACCEPT`, `DENY`, `REJECT` o `MASQ`. `MASQ` è valida solamente per la catena 'forward'.

Inoltre è importante notare che un obiettivo `RETURN` in una regola in una delle catene predefinite è utile per stabilire esplicitamente la tattica di una catena quando un pacchetto soddisfa una regola.

4.1.6 Operazioni sul masquerading

Ci sono diversi parametri per l'IP Masquerading con i quali si può giocare. Sono inglobati in `ipchains` perché non valeva la pena scrivere uno strumento separato (anche se questa cosa cambierà).

Il comando per il masquerading IP è '-M', e può essere combinato con '-L' per mostrare l'elenco delle connessioni attualmente mascherate, o con '-S' per impostare i parametri del masquerading.

Il comando '-L' può essere accompagnato da '-n' (mostra i numeri invece dei nomi degli host e delle porte) o '-v' (mostra i delta nelle sequenze di numeri per le connessioni mascherate, nel caso importi qualcosa).

Il comando '-S' dovrebbe essere seguito da tre valori di timeout in secondi: per le sessioni TCP, per le sessioni TCP dopo un pacchetto FYN e per i pacchetti UDP. Se non si vuole cambiare uno di questi tre valori, semplicemente si specifichi '0' come valore.

I valori predefiniti sono elencati in '/usr/src/linux/include/net/ip_masq.h', e rispettivamente sono 15 minuti, 2 minuti e 5 minuti.

Il valore più comune da cambiare è il primo, per FTP (si veda 5.2.3 (Incubi da FTP) più avanti).

Si noti il problema nell'impostazione dei timeout descritto in 6.10 (Non riesco a impostare i timeout del masquerading!).

4.1.7 Controllare un pacchetto

Talvolta si vuole vedere cosa succede quando un certo pacchetto entra nella propria macchina, ad esempio per fare il debug delle catene firewall. `ipchains` ha il comando '-C' per permetterlo, che usa le stesse routine che usa il kernel per la diagnosi dei pacchetti reali.

Si specifica su quale catena verificare il pacchetto facendo seguire l'argomento di '-C' con il suo nome. Mentre il kernel inizia la trasversata sempre dalla catena `input`, `output` oppure `forward`, per gli scopi di test si ha il permesso di cominciare la traversata da qualsiasi catena.

I dettagli del 'pacchetto' sono specificati usando le stessa sintassi usata per specificare le regole firewall. In particolare, sono obbligatori un protocollo ('-p'), un indirizzo di provenienza ('-s'), un indirizzo di destinazione ('-d') e un'interfaccia ('-i'). Se il protocollo è TCP o UDP, allora devono essere specificati un unico indirizzo di provenienza e un unico indirizzo di destinazione, mentre devono essere specificati un tipo e un codice ICMP per il protocollo ICMP (a meno che non sia specificata l'opzione '-f' per indicare una regola sui frammenti, nel qual caso queste opzioni sono illegali).

Se il protocollo è TCP (e non è specificata l'opzione '-f'), può essere specificata l'opzione '-y' per indicare che il pacchetto di test avrà il bit SYN impostato.

Ecco qui un esempio di verifica di un pacchetto SYN TCP dalla porta 60000 di 192.168.1.1 alla porta www di 192.168.1.2, in arrivo sull'interfaccia eth0 e che entra nella catena 'input' (questa è la classica inizializzazione di una connessione WWW):

```
# ipchains -C input -p tcp -y -i eth0 -s 192.168.1.1 60000 -d 192.168.1.2 www
packet accepted
#
```

4.1.8 Più regole in una volta sola e controllare cosa succede

Talvolta un'unica riga di comando può aver effetto su più regole. Ciò capita in due situazioni. La prima: se si può specificare un nome di host che viene risolto (usando il DNS) in diversi indirizzi IP, `ipchains` si comporterà come se si fossero digitati più comandi, uno per ogni combinazione di indirizzi.

Quindi se il nome di host 'www.foo.com' viene risolto in tre indirizzi IP e il nome di host 'www.bar.com' viene risolto in due indirizzi IP, allora il comando 'ipchains -A input -j reject -s www.bar.com -d www.foo.com' aggiungerà sei regole alla catena `input`.

Un altro modo per far sì che `ipchains` effettui azioni multiple è di usare l'opzione '-b' (bidirezionale). Questa opzione fa sì che `ipchains` si comporti come se si fosse digitato due volte il comando, la seconda volta scambiando gli argomenti di '-s' e '-d'. Quindi per evitare l'inoltro sia da che per 192.168.1.1, si potrebbe fare quanto segue:

```
# ipchains -b -A forward -j reject -s 192.168.1.1
#
```

Personalmente, non gradisco molto l'opzione '-b'; se si vuole qualcosa di più utile si veda 4.2.1 (Usare ipchains-save) più avanti.

L'opzione '-b' può essere usato con i comandi di inserimento ('-I'), cancellazione ('-D') (ma non le varianti che accettano un numero di regola), aggiunta ('-A') e verifica ('-C').

Un'altra opzione utile è '-v' (verboso) che mostra esattamente quel che `ipchains` sta facendo con il comando dato. È utile se si ha a che fare con comandi che possono evere effetto su più regole. Per esempio, controlliamo il comportamento dei frammenti tra 192.168.1.1 e 192.168.1.2.

```
# ipchains -v -b -C input -p tcp -f -s 192.168.1.1 -d 192.168.1.2 -i lo
  tcp opt   ---f- tos 0xFF 0x00  via lo    192.168.1.1  -> 192.168.1.2    * ->   *
packet accepted
  tcp opt   ---f- tos 0xFF 0x00  via lo    192.168.1.2  -> 192.168.1.1    * ->   *
packet accepted
#
```

4.2 Un utile esempio

Ho una connessione PPP in dialup (`-i ppp0`). Mi scarico le news (`-p TCP -s news.virtual.net.au nntp`) e la posta (`-p TCP -s mail.virtual.net.au pop-3`) ogni volta che mi connetto. Uso il metodo FTP di Debian per aggiornare regolarmente la mia macchina (`-p TCP -y -s ftp.debian.org.au ftp-data`). Navigo in rete attraverso il proxy del mio ISP mentre tutta la roba precedente è in funzione (`-p TCP -d proxy.virtual.net.au 8080`), ma odio le pubblicità da doubleclick.net nel Dilbert Archive (`-p TCP -y -d 199.95.207.0/24` e `-p TCP -y -d 199.95.208.0/24`).

Non mi preoccupo della gente che prova a fare ftp nella mia macchina mentre sono online (`-p TCP -d $LOCALIP ftp`), ma non voglio che nessuno da fuori pretenda di avere un indirizzo IP della mia rete interna (`-s 192.168.1.0/24`). Ciò è comunemente detto IP spoofing, e c'è un modo migliore per proteggersi nei kernel 2.1 e superiori: si veda 5.7 (Come proteggersi dall'IP spoofing?).

Questa configurazione è piuttosto semplice, perché attualmente non ci sono altre macchine nella mia rete interna.

Non voglio che nessun processo locale (ie. Netscape, lynx ecc.) si connetta a doubleclick.net:

```
# ipchains -A output -d 199.95.207.0/24 -j REJECT
# ipchains -A output -d 199.95.208.0/24 -j REJECT
#
```

Ora voglio impostare le priorità in diversi pacchetti in uscita (non ne vedo l'utilità di farlo nei pacchetti in ingresso). Poiché ho parecchie di questo regole, ha senso metterle tutte in un unica catena, chiamata `ppp-out`.

```
# ipchains -N ppp-out
# ipchains -A output -i ppp0 -j ppp-out
#
```

Ritardo minimo per il traffico web e per telnet.

```
# ipchains -A ppp-out -p TCP -d proxy.virtual.net.au 8080 -t 0x01 0x10
# ipchains -A ppp-out -p TCP -d 0.0.0.0/0 telnet -t 0x01 0x10
#
```

Minimo costo per i dati ftp, nntp e pop-3:

```
# ipchains -A ppp-out -p TCP -d 0.0.0.0/0 ftp-data -t 0x01 0x02
# ipchains -A ppp-out -p TCP -d 0.0.0.0/0 nntp -t 0x01 0x02
# ipchains -A ppp-out -p TCP -d 0.0.0.0/0 pop-3 -t 0x01 0x02
#
```

Ci sono alcune restrizioni sui pacchetti in ingresso dall'interfaccia ppp0: creo una catena chiamata 'ppp-in':

```
# ipchains -N ppp-in
# ipchains -A input -i ppp0 -j ppp-in
#
```

Ora, nessun pacchetto in ingresso da `ppp0` dovrebbe affermare un indirizzo di provenienza di 192.168.1.*, e quindi li registro e li proibisco:

```
# ipchains -A ppp-in -s 192.168.1.0/24 -l -j DENY
#
```

Permetto l'ingresso solo di pacchetti UDP per il DNS (eseguo un caching nameserver che inoltra tutte le richieste a 203.29.16.1, quindi mi aspetto risposte DNS solo da loro), ftp entrante e ritorno di dati ftp (che dovrebbero andare solo verso una porta sopra la 1023, ma non verso le porte X11 attorno a 6000).

```
# ipchains -A ppp-in -p UDP -s 203.29.16.1 -d $LOCALIP dns -j ACCEPT
# ipchains -A ppp-in -p TCP -s 0.0.0.0/0 ftp-data -d $LOCALIP 1024:5999 -j ACCEPT
# ipchains -A ppp-in -p TCP -s 0.0.0.0/0 ftp-data -d $LOCALIP 6010: -j ACCEPT
# ipchains -A ppp-in -p TCP -d $LOCALIP ftp -j ACCEPT
#
```

Permeto il ritorno dei pacchetti TCP di risposta

```
# ipchains -A ppp-in -p TCP ! -y -j ACCEPT
#
```

Per finire, vanno bene i pacchetti local-to-local:

```
# ipchains -A input -i lo -j ACCEPT
#
```

ora, la mia tattica di default per la catena `input` è `DENY`, quindi qualsiasi altra cosa viene scartata:

```
# ipchains -P input DENY
#
```

NOTA: non imposterei le mie catene in questo ordine, in quanto i pacchetti potrebbero passare mentre le imposto. La cosa più sicura è di impostare per prima cosa la tattica a DENY, poi inserire le regole. Naturalmente, se le proprie regole necessitano di ricerche DNS per risolvere i nomi di host, potrebbero esserci problemi.

4.2.1 Usare ipchains-save

Impostare le catene firewall proprio nel modo in cui le si vuole, e poi provare a ricordarsi i comandi usati in modo da porterlo fare anche la volta successiva è una cosa penosa.

`ipchains-save` è uno script che legge l'impostazione corrente delle catene e la salva in un file. Per ora vi lascio in fremente attesa di scoprire cosa fa `ipchains-restore`.

`ipchains-save` può salvare una catena o tutte le catene (se non è specificato un nome di catena). La sola opzione attualmente permessa è '-v' che stampa le regole (in stderr) mentre le salva. Per le catene `input`, `output` e `forward` è salvata anche la tattica.

```
# ipchains-save > my_firewall
Saving 'input'.
Saving 'output'.
Saving 'forward'.
Saving 'ppp-in'.
Saving 'ppp-out'.
#
```

4.2.2 Usare ipchains-restore

`ipchains-restore` ripristina le catene salvate con `ipchains-save`. Accetta due opzioni: '-v' che descrive ogni regola che viene aggiunta, e '-f' che forza lo svuotamento delle catene definite dall'utente se esistono, come descritto nel seguito.

Se nell'input è trovata una catena definita dall'utente, `ipchains-restore` controlla se esiste già. Se esiste, sarà chiesto se la catena debba essere svuotata (ripulita da tutte le regole) o se si debba saltare il ripristino di questa catena. Se si specifica '-f' in riga di comando, non sarà chiesto niente; la catena sarà ripulita.

Per esempio:

```
# ipchains-restore < my_firewall
Restoring 'input'.
Restoring 'output'.
Restoring 'forward'.
Restoring 'ppp-in'.
Chain 'ppp-in' already exists. Skip or flush? [S/f]? s
Skipping 'ppp-in'.
Restoring 'ppp-out'.
Chain 'ppp-out' already exists. Skip or flush? [S/f]? f
Flushing 'ppp-out'.
#
```

5 Miscellanea

Questa sezione contiene tutte le informazioni e le FAQ che non sono riuscito a sistemare nella struttura precedente.

5.1 Come organizzare le proprie regole firewall

Questa domanda richiede qualche riflessione. Si può provare a organizzarle per ottimizzare la velocità (minizzare il numero di verifiche di regole per i pacchetti più comuni) o per incrementare la gestibilità.

Se si ha una connessione intermittente, diciamo una connessione PPP, si può voler impostare la prima regola nella catena input a '-i ppp0 -j DENY' al boot del sistema, e poi mettere qualcosa di simile a questo nello script `ip-up`:

```
# Ricrea la catena 'ppp-in'.
ipchains-restore -f < ppp-in.firewall

# Rimpiazza la regola DENY con un salto alla catena di gestione del ppp.
ipchains -R input 1 -i ppp0 -j ppp-in
```

Lo script `ip-down` potrebbe essere così:

```
ipchains -R input 1 -i ppp0 -j DENY
```

5.2 Cosa non filtrare

Ci sono alcune cosette di cui bisogna essere consci prima di cominciare a filtrare tutto quello che non si vuole.

5.2.1 Pacchetti ICMP

I pacchetti ICMP sono usati (tra le altre cose) per indicare fallimenti negli altri protocolli (come TCP o UDP). In particolare i pacchetti 'destination-unreachable' (destinazione irraggiungibile). Bloccare questi pacchetti significa che non si riceveranno mai gli errori 'Host unreachable' o 'No route to host'; qualsiasi connessione semplicemente attenderà una riposta che non arriverà mai. Ciò è irritante, ma raramente fatale.

Un problema peggiore è la regola dei pacchetti ICMP nel MTU discovery. Tutte le buone implementazioni TCP (inclusa quella di Linux) usano MTU discovery per provare a capire quale sia il pacchetto più grosso che può arrivare a destinazione senza essere frammentato (la frammentazione abbassa le prestazioni. specialmente quando vengono occasionalmente persi dei frammenti). MTU discovery lavora inviando pacchetti con il bit Don't Fragment impostato, inviando poi pacchetti più piccoli se riceve un pacchetto ICMP che indica Fragmentation needed but DF set ('fragmentation-needed'). Questo è un pacchetto tipo 'destination-unreachable', e se non viene mai ricevuto l'host locale non riduce l'MTU e le prestazioni saranno abissali o non esistenti.

Si noti che è comune bloccare tutti i messaggi redirect ICMP (tipo 5); possono essere usati per manipolare l'instradamento (sebbene gli stack IP buoni abbiano delle protezioni), e quindi sono spesso visti un come po' rischiosi.

5.2.2 Connessioni TCP al DNS (nameserver)

Se si sta provando a bloccare tutte le connessioni TCP in uscita, si ricordi che il DNS non sempre usa UDP; se la risposta dal server supera i 512 byte, il client usa una connessione TCP (ancora diretta alla porta numero 53) per ottenere i dati.

Ciò può essere una trappola perché il DNS 'praticamente funzionerà' se si disabilitano tali trasferimenti TCP; comunque se lo si fa possono capitare strani lunghi ritardi e altri occasionali problemi DNS.

Se le proprie interrogazioni DNS sono sempre dirette alla stessa fonte esterna (sia direttamente usando una riga `nameserver` in `/etc/resolv.conf` oppure usando un caching nameserver in modalità forward), allora si devono permettere solo le connessioni TCP alla porta `domain` di quel nameserver dalla porta `domain` locale (se si sta usando un caching nameserver) o da una porta più alta (> 1023) se si sta usando `/etc/resolv.conf`.

5.2.3 Incubi da FTP

FTP presenta un classico problema del filtraggio dei pacchetti. FTP ha due **modalità**; quella tradizionale è detta **modalità attiva** e quella più recente è detta **modalità passiva**. I web browser solitamente usano la modalità passiva, mentre i programmi per FTP a riga di comando solitamente usano la modalità attiva.

In modalità attiva, quando il sito remoto vuole inviare un file (oppure anche il risultato di un comando `ls` o `dir`) apre una connessione TCP verso la macchina locale. Ciò significa che non si possono filtrare queste connessioni TCP senza rompere l'FTP attivo.

Se si ha la possibilità di usare la modalità passiva, allora bene; la modalità passiva crea connessioni dati dal client al server, anche per i dati in ingresso. Altrimenti, è raccomandabile permettere connessioni TCP solamente verso porte superiori alla 1024 ma non tra 6000 e 6010 (la 6000 è usata per X-Windows).

5.3 Filtrare i Ping della Morte

La macchine Linux sono ora immuni ai famosi **Ping della Morte**, che implicano l'invio di un pacchetto ICMP illegalmente grande che fa andare in overflow i buffer nello stack TCP del ricevente con effetti devastanti.

Se vi vogliono proteggere macchine che potrebbero essere ancora vulnerabili, semplicemente si blocchino i frammenti ICMP. Normalmente i pacchetti ICMP non sono abbastanza grandi da richiedere frammentazione, e quindi non si romperà niente se non i grossi ping. Ho sentito (non confermato) che a alcuni sistemi basta anche solo l'ultimo frammento di un pacchetto ICMP fuori misura per corromperli, e quindi non è raccomandabile bloccare solo il primo frammento.

Sebbene i programmi exploit che ho visto usano tutti ICMP, non c'è ragione per non usare frammenti TCP o UDP (o di un protocollo sconosciuto) per questi attacchi, e quindi bloccare i frammenti ICMP è solamente una soluzione temporanea.

5.4 Filtrare Teardrop e Bonk

Teardrop e Bonk sono due attacchi (rivolti principalmente contro macchine Microsoft Windows NT) che si basano sulla sovrapposizione dei frammenti. Le opzioni sono di far sì che il proprio router Linux effettui la deframmentazione oppure disabilitare tutti i frammenti verso le macchine vulnerabili.

5.5 Filtrare i Fragment Bomb

Si dice che alcuni stack TCP meno affidabili hanno problemi a gestire un grande numero di frammenti di pacchetti quando non ricevono mai tutti i frammenti. Linux non ha questo problema. Si possono filtrare tutti i frammenti (il che interrompe pure il loro uso legittimo) oppure compilare il kernel attivando 'IP: always defragment' (solo se la propria macchina Linux è il solo instradamento possibile per questi pacchetti).

5.6 Cambiare le regole firewall

Ci sono alcune questioni temporali coinvolte nella modifica delle regole firewall. Se non si fa attenzione, si possono lasciar passare pacchetti mentre si fanno le modifiche. L'approccio più semplice è il seguente:

```
# ipchains -I input 1 -j DENY
# ipchains -I output 1 -j DENY
# ipchains -I forward 1 -j DENY

... fare le modifiche ...

# ipchains -D input 1
# ipchains -D output 1
# ipchains -D forward 1
#
```

Ciò scarta tutti i pacchetti per la durata delle modifiche.

Se le proprie modifiche sono ristrette a una sola catena, si potrebbe creare una nuova catena con le nuove regole e poi rimpiazzare ('-R') la regola che punta alla vecchia catena con quella che punta a quella nuova: poi si può cancellare la vecchia catena. Questo rimpiazzo sarà atomico.

5.7 Come proteggersi dall'IP Spoofing?

L'IP spoofing è una tecnica nella quale un host invia pacchetti che affermano provenire da un altro host. Poiché il filtraggio dei pacchetti prende decisioni basandosi su questo indirizzo di provenienza, l'IP spoofing è utile solo con filtri di pacchetti un po' stupidi. È usato anche per nascondere l'identità dell'attaccante usando attacchi SYN, Teardrop, Ping della Morte e simili (non ci si preoccupi se non si sa cosa sono).

Il miglior modo per proteggersi dall'IP spoofing è chiamato Source Address Verification (Verifica dell'Indirizzo di Provenienza), ed è fatto dal codice di instradamento e non dal firewall. Si cerchi il file `/proc/sys/net/ipv4/conf/all/rp_filter`. Se esiste, allora attivare il Source Address Verification a ogni avvio è la giusta soluzione. Per farlo, si inseriscano le righe seguenti da qualche parte nei propri script di inizializzazione, prima che sia inizializzata qualsiasi interfaccia di rete:

```
# Questo è il metodo migliore: attivare il Source Address Verification
# e avere così la protezione dallo spoof su tutte le intefacce
# correnti e future.
if [ -e /proc/sys/net/ipv4/conf/all/rp_filter ]; then
  echo -n "Setting up IP spoofing protection..."
  for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
      echo 1 > $f
  done
  echo "done."
else
  echo PROBLEMS SETTING UP IP SPOOFING PROTECTION.  BE WORRIED.
  echo "CONTROL-D will exit from this shell and continue system startup."
  echo
  # Start a single user shell on the console
  /sbin/sulogin $CONSOLE
fi
```

Se non si può fare così, si possono inserire manualmente delle regole per proteggere ogni interfaccia. Ciò richiede conoscenza su qualsiasi interfaccia. I kernel 2.1 automaticamente rifiutano i pacchetti che affermano provenire dagli indirizzi 127.* (riservati per l'interfaccia loopbak locale `lo`).

Per esempio, facciamo il caso che ci siano tre interfacce: `eth0`, `eth1` e `ppp0`. Possiamo usare `ifconfig` per conoscere l'indirizzo e la netmask delle interfacce. Diciamo che `eth0` sia attaccata alla rete 192.168.1.0 con netmask 255.255.255.0, `eth1` sia attaccata alla rete 10.0.0.0 con netmask 255.0.0.0 e che `ppp0` connetta con Internet (dov'è permesso qualsiasi indirizzo tranne quelli riservati come indirizzi IP privati). Inseriremo allora le seguenti regole:

```
# ipchains -A input -i eth0 -s ! 192.168.1.0/255.255.255.0 -j DENY
# ipchains -A input -i ! eth0 -s 192.168.1.0/255.255.255.0 -j DENY
# ipchains -A input -i eth1 -s ! 10.0.0.0/255.0.0.0 -j DENY
# ipchains -A input -i ! eth1 -s 10.0.0.0/255.0.0.0 -j DENY
#
```

Questo approccio non è buono quanto l'approccio con il Source Address Verification, perché se cambia la propria rete, si devono cambiare le regole firewall.

Se si usa un kernel della serie 2.0, si può voler proteggere anche l'interfacia loopback, usando una regola come questa:

```
# ipchains -A input -i ! lo -s 127.0.0.0/255.0.0.0 -j DENY
#
```

5.8 Progetti avanzati

Ho scritto una libreria ('libfw') che funziona dello spazio utente inclusa nei sorgenti. Usa la possibilità offerta da IP Chains 1.3 e superiori di copiare un pacchetto nello spazio utente (usando l'opzione di configurazione IP_FIREWALL_NETLINK).

Il valore marcato può essere usato per specificare il paramentro Quality of Service per i pacchetti o per specificare come fare l'inoltro di porta dei pacchetti. Non li ho mai usati, ma se si vuole scrivere qualcosa in proposito, mi si contatti.

Usando questa libreria possono essere implementate nello spazio utente cose come la **stateful inspection** (preferisco il termine dynamic firewalling). Un'altra idea interessante è il controllo dei pacchetti su base utente facendo delle ricerche in un demone nello spazio utente. Questo dovrebbe essere abbastanza facile.

5.8.1 SPF: Stateful Packet Filtering

ftp://ftp.interlinx.bc.ca/pub/spf `ftp://ftp.interlinx.bc.ca/pub/spf` è il sito del progetto SPF di Brian Murrell, che fa il tracking delle connessioni nello spazio utente. Aggiunge sicurezza significativa per siti a bassa banda.

Attualmente c'è poca documentazione, ma ecco qui un post nella mailing list nel quale Brian spiega un po' di cose:

```
> Credo che faccia esattamente quello che voglio: installare un regola
> temporanea di "arretramento" ("backward" rule) per permettere
> l'ingresso dei pacchetti come fossero una risposta a una richiesta
> in uscita.

Yup, è esattamente quello che fa.  Più protocolli supporta, più la
regola di "arretramento" funziona bene.  Attualmente ha il supporto
(vado a memoria, quindi scusate qualsiasi errore o omissione) per FTP
(sia attivo che passivo, in ingresso e in uscita), RealAudio,
traceroute, ICMP e ICQ basilare (ingresso da un server ICQ e
connessioni TCP dirette, ma non c'è ancora il supporto per le
connessioni TCP dirette secondarie per altre cose come il
trasferimento di file, ecc.).

> È un rimpiazzo per ipchains o un supplemento?

È un supplemento.  Penso a ipchains come al motore per permettere o
prevenire ai pacchetti di viaggiare attraverso la macchina Linux.  SPF
è il pilota che monitorizza costantemente il traffico e dice a
ipchains come cambiare le sue tattiche per rispondere ai cambiamenti
nel traffico.
```

5.8.2 L'ftp-data hack di Michael Hasenstein

Michael Hasenstein della SuSE ha scritto una patch per il kernel che aggiunge a ipchains il tracking delle connessioni ftp. Attualmente può essere trovata a *http://www.suse.de/˜mha/patch.ftp-data-2.gz*

5.9 Estensioni future

Il firewalling e il NAT sono in fase di riprogettazione per il 2.4. I piani e le discussioni sono disponibili nella lista netfilter (si veda *http://lists.samba.org*). Queste estensioni dovrebbero chiarire parecchie questioni insolute sull'usabilità (veramente, il firewalling e il masquerading non dovrebbero essere *così difficili*), e permetteranno la crescita di un firewalling maggiormente flessibile.

6 Problemi comuni

6.1 ipchains -L si pianta!

Probabilmente si stanno bloccando le ricerche DNS; alla fine andrà in timeout. Si provi a passare l'opzione '-n' a ipchains, che sopprime la risoluzione dei nomi.

6.2 Le opzioni negate non funzionano!

Si deve mettere l'opzione '!' da sola, con degli spazi da entrambi i lati. Un errore classico (segnalato dalla versione 1.3.10) è:

```
# ipchains -A input -i !eth0 -j DENY
#
```

Non esisterà mai un'interfaccia chiamata '!eth0', ma ipchains non sa questa cosa.

6.3 Masquerading/Forwarding non funziona!

Ci si assicuri che sia abilitato l'inoltro dei pacchetti (nei kernel recenti è disabilitato per default, il che significa che i pacchetti non proveranno mai ad attraversare la catena 'forward'). Si può venirne a capo digitando (come root):

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

Se funziona, lo si può mettere da qualche parte nei propri script di avvio così che sia abilitato ogni volta; ovviamente è meglio impostare il proprio firewall prima di lanciare questo comando, altrimenti c'è l'opportunità che scappino un po' di pacchetti.

6.4 -j REDIR non funziona!

Si devono permettere i pacchetti di inoltro (si veda sopra) affinché funzioni il dirottamento; diversamente il codice di instradamento scarterà i pacchetti. Quindi se si sta usando solamente il dirottamento e non si usa il forwarding, è bene essere consci di questa cosa.

Si noti che REDIR (sebbene sia nella catena input) non ha effetto sulle connessioni da un processo locale.

6.5 Non funzionano i caratteri jolly nelle interfacce!

C'è un bug nelle versioni 2.1.102 e 2.1.103 del kernel (e in alcune vecchie patch che ho prodotto) che faceva fallire i comandi ipchains che utilizzavano caratteri jolly per specificare interfacce (ad esempio `-i ppp+`).

Ciò è stato corretto nei kernel recenti e nella patch per il 2.0.34 presente sul sito web. Può essere pure corretto a mano nei sorgenti del kernel modificando la riga 63 (più o meno) di include/linux/ip_fw.h:

```
#define IP_FW_F_MASK    0x002F  /* All possible flag bits mask   */
```

Dovrebbe essere "0x003F". Lo si corregga e si ricompili il kernel.

6.6 TOS non funziona!

Questo è stato un mio errore: l'impostazione del campo Type of Service nei kernel dal 2.1.102 al 2.1.111 in realtà non faceva niente. Questo problema è stato corretto nel 2.1.112.

6.7 Non funzionano ipautofw e ipportfw!

Per i 2.0.x, è vero; non ho il tempo per creare e mantenere una patch enorme per ipchains e ipautofw/ipportfw.

Per i 2.1.x, si scarichi l'ipmasqadm di Juan Ciarlante da

```
<htmlurl url="http://juanjox.linuxhq.com/"
        name="http://juanjox.linuxhq.com/">
```

e lo si usi esattamente come si sarebbe usato `ipautofw` o `ipportfw`, tranne per il fatto che invece di `ipportfw` si usa `ipmasqadm portfw`, e invece di `ipautofw` si usa `ipmasqadm autofw`.

6.8 xosview si è rotto!

Si aggiorni alla versione 1.6.0 o superiore, che non richiede nessuna regola firewall per i kernel 2.1.x. Sembra che anche la release 1.6.1 abbia questo problema; lo si segnali all'autore (non è un mio errore!).

6.9 Segmentation Fault con '-j REDIRECT'!

Questo era un bug in ipchains versione 1.3.3. Si aggiorni.

6.10 Non riesco a impostare i timeout del masquerading!

Ciò è vero (per i kernel 2.1.x) fino al 2.1.123. Nel 2.1.124, il tentativo di impostare i timeout del masquerading provoca un blocco del kernel (si modifichi `return` in `ret =` nella riga 1328 di net/ipv4/ip_fw.c). Nel 2.1.125, funziona tutto.

6.11 Voglio dei firewall IPX!

E così molti altri, sembra. Il mio codice gestisce solo IP, sfortunatamente. D'altra parte c'è anche qualcosa di buono: le cose per scrivere un firewall IPX ci sono tutte! Basta semplicemente scrivere il codice; sarò felice di aiutare dove possibile.

7 Un esempio più serio

Questo esempio è estratto dal tutorial scritto da me e Michael Neuling apparso in LinuxWorld nel Marzo 1999; non è il solo modo per risolvere il problema in esame, ma probabilmente è il più semplice. Spero lo si trovi interessante.

7.1 La situazione

- Rete interna con il masquerading (con diversi sistemi operativi) che chiameremo "GOOD".
- Server esposti in una rete separata (chiamata "DMZ" per Demilitarized Zone – Zona Demilitarizzata).
- Connessione PPP a Internet (detta "BAD").

```
        Rete Esterna (BAD)
                |
                |
            ppp0|
         ---------------
         | 192.84.219.1|             Rete dei Server (DMZ)
         |             |eth0
         |             |----------------------------------------------
         |             |192.84.219.250 |             |              |
         |             |               |             |              |
         |192.168.1.250|               |             |              |
         ---------------          --------      -------        -------
                | eth1            | SMTP |      | DNS |        | WWW |
                |                 --------      -------        -------
                |              192.84.219.128  192.84.219.129  192.84.218.130
                |
       Rete Interna (GOOD)
```

7.2 Scopi

Macchina per il filtraggio dei pacchetti:

PING da/per qualsiasi rete

Questo è veramente utile per sapere se una macchina è attiva.

TRACEROUTE da/per qualsiasi rete

Ancora, utile per la diagnostica.

Accesso DNS

Rende ping e DNS più utili.

All'interno di DMZ:

Mail server

- SMTP verso l'esterno
- Accetta SMTP dall'interno e dall'esterno
- Accetta POP-3 dall'interno

Name Server

- Invia richieste DNS all'esterno
- Accetta richieste DNS dall'interno, dall'esterno e dalla macchina per il filtraggio dei pacchetti.

Server web

- Accetta connessioni HTTP dall'interno e dall'esterno
- Accesso rsync dall'interno

Rete interna:

Permettere WWW, ftp, traceroute, ssh verso l'esterno

Queste sono cose abbastanza standard da permettere: alcuni iniziano permettendo di fare tutto alla rete interna, ma qui saremo un po' più restrittivi.

Permettere connessioni SMTP verso il mail server

Ovviamente vogliamo poter inviare mail verso l'esterno.

Permettere connessioni POP-3 verso il mail server

Questo è il modo per leggere la propria posta.

Permettere connessioni DNS verso il name server

Vogliamo essere in grado di cercare i nomi esterni dei siti per il WWW, ftp, traceroute e ssh.

Permettere connessioni rsync verso il server web

Questo è il modo per sincronizzare il server web esterno con quello interno.

Permettere connessioni WWW verso il server web

Ovviamente, vogliamo poterci connettere al nostro server web esterno.

Permettere il ping verso la macchina per il filtraggio dei pacchetti

Questa è una pura cortesia: significa che si può controllare se la macchina firewall è giù.

7.3 Prima del filtraggio dei pacchetti

- Anti-spoofing.

  Poiché non c'è nessun instradamento asimmetrico, semplicemente si può attivare l'anti-spoofing per tutte le interfacce.

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

- Impostare le regole di filtraggio per proibire (DENY) tutto.

  Si permetterà ancora il traffico locale su loopback, ma verrà proibito tutto il resto.

  ```
  # ipchains -A input -i ! lo -j DENY
  # ipchains -A output -i ! lo -j DENY
  # ipchains -A forward -j DENY
  #
  ```

- Configurare le interfacce.

  Solitamente ciò è fatto negli script di boot. Ci si assicuri che i passi precedenti siano fatti prima che vengano configurate le interfacce, per prevenire l'infiltrazione di pacchetti prima di aver impostato le regole.

- Inserire moduli di masquerading per ciascun protocollo. Si devono inserire i moduli di masquerading per FTP, in modo che funzioni l'FTP sia attivo che passivo per la rete interna.

  ```
  # insmod ip_masq_ftp
  #
  ```

7.4 Filtraggio dei pacchetti per i pacchetti di passaggio

Con il masquerading, la cosa migliore è filtrare nella catena forward.

Si suddivida la catena forward in diverse catene utente a seconda delle interfacce di provenienza/destinazione; ciò spacca il problema in tronconi gestibili più facilmente.

```
ipchains -N good-dmz
ipchains -N bad-dmz
ipchains -N good-bad
ipchains -N dmz-good
ipchains -N dmz-bad
ipchains -N bad-good
```

Una cosa comune da fare è di accettare (ACCEPT) errori ICMP standard, e quindi si crea una catena solo per loro.

```
ipchains -N icmp-acc
```

7.4.1 Impostare i salti dalla catena forward

Sfortunatamente, è nota solamente (nella catena forward) l'interfaccia d'uscita. Quindi, per scoprire da quale interfaccia sia entrato un pacchetto useremo l'indirizzo di provenienza (l'anti spoofing previene indirizzi contraffatti).

Si noti che si registra qualsiasi cosa che non soddisfa una di queste regole (ovviamente ciò non dovrebbe mai succedere).

```
ipchains -A forward -s 192.168.1.0/24 -i eth0 -j good-dmz
ipchains -A forward -s 192.168.1.0/24 -i ppp0 -j good-bad
ipchains -A forward -s 192.84.219.0/24 -i ppp0 -j dmz-bad
ipchains -A forward -s 192.84.219.0/24 -i eth1 -j dmz-good
ipchains -A forward -i eth0 -j bad-dmz
ipchains -A forward -i eth1 -j bad-good
ipchains -A forward -j DENY -l
```

7.4.2 Definire la catena icmp-acc

I pacchetti che siano degli errori ICMP sono accettati, altrimenti il controllo passerà alla catena chiamante.

```
ipchains -A icmp-acc -p icmp --icmp-type destination-unreachable -j ACCEPT
ipchains -A icmp-acc -p icmp --icmp-type source-quench -j ACCEPT
ipchains -A icmp-acc -p icmp --icmp-type time-exceeded -j ACCEPT
ipchains -A icmp-acc -p icmp --icmp-type parameter-problem -j ACCEPT
```

7.4.3 Da Good (interno) a DMZ (server)

Restrizioni della rete interna:

- Permettere WWW, ftp, traceroute, ssh verso l'esterno
- **Permettere SMTP verso il Mail server**

- **Permettere POP-3 verso il Mail server**
- **Permettere DNS verso il Name server**
- **Permettere rsync verso il Web server**
- **Permettere WWW verso il Web server**
- Permettere il ping alla macchina filtro

Si potrebbe fare il masquerading dalla rete interna nella DMZ, ma qui non si farà. Poiché nessuno nella rete interna dovrebbe provare a fare cose brutte, registriamo qualsiasi pacchetto che venga proibito.

Si noti che le vecchie versioni di Debian usavano 'pop-3' invece di 'pop3' in /etc/services, cosa che viola l'RFC1700.

```
ipchains -A good-dmz -p tcp -d 192.84.219.128 smtp -j ACCEPT
ipchains -A good-dmz -p tcp -d 192.84.219.128 pop3 -j ACCEPT
ipchains -A good-dmz -p udp -d 192.84.219.129 domain -j ACCEPT
ipchains -A good-dmz -p tcp -d 192.84.219.129 domain -j ACCEPT
ipchains -A good-dmz -p tcp -d 192.84.218.130 www -j ACCEPT
ipchains -A good-dmz -p tcp -d 192.84.218.130 rsync -j ACCEPT
ipchains -A good-dmz -p icmp -j icmp-acc
ipchains -A good-dmz -j DENY -l
```

7.4.4 Da Bad (esterno) a DMZ (server).

- Restrizioni di DMZ:
  - Mail server
    - **SMTP verso l'esterno**
    - **Accetta SMTP da** interno ed **esterno**
    - Accetta POP-3 dall'interno
  - Name server
    - **Invia DNS verso l'esterno**
    - **Accetta DNS da** interno, **esterno** e dalla macchina per il filtraggio dei pacchetti
  - Web server
    - **Accetta HTTP da** interno e **esterno**
    - Accesso rsync dall'interno
- Cose permesse dalla rete esterna verso DMZ
  - Non si registrano le violazioni, in quanto possono succedere.

```
ipchains -A bad-dmz -p tcp -d 192.84.219.128 smtp -j ACCEPT
ipchains -A bad-dmz -p udp -d 192.84.219.129 domain -j ACCEPT
ipchains -A bad-dmz -p tcp -d 192.84.219.129 domain -j ACCEPT
ipchains -A bad-dmz -p tcp -d 192.84.218.130 www -j ACCEPT
ipchains -A bad-dmz -p icmp -j icmp-acc
ipchains -A bad-dmz -j DENY
```

7.4.5 Da Good (interno) a Bad (esterno).

- Restrizioni della rete interna:
  - Permettere WWW, ftp, traceroute, ssh verso l'esterno
  - **Permettere SMTP verso il Mail server**
  - **Permettere POP-3 verso il Mail server**
  - **Permettere DNS verso il Name server**
  - **Permettere rsync verso il Web server**
  - **Permettere WWW verso il Web server**
  - Permettere il ping alla macchina filtro
- Molti permettono qualsiasi cosa dalla rete interna verso quella esterna. Qua facciamo un po' i fascisti.
  - Registrazione delle violazioni.
  - FTP passivo gestito dal modulo del masquerading
  - le porte di destinazione UDP 33434 e successive sono usate da traceroute.

```
ipchains -A good-bad -p tcp --dport www -j MASQ
ipchains -A good-bad -p tcp --dport ssh -j MASQ
ipchains -A good-bad -p udp --dport 33434:33500 -j MASQ
ipchains -A good-bad -p tcp --dport ftp -j MASQ
ipchains -A good-bad -p icmp --icmp-type ping -j MASQ
ipchains -A good-bad -j REJECT -l
```

7.4.6 Da DMZ a Good (interno).

- Restrizioni della rete interna:
  - Permettere WWW, ftp, traceroute, ssh verso l'esterno
  - **Permettere SMTP verso il Mail server**
  - **Permettere POP-3 verso il Mail server**
  - **Permettere DNS verso il Name server**
  - **Permettere rsync verso il Web server**
  - **Permettere WWW verso il Web server**
  - Permettere il ping alla macchina filtro
- Se si fa il masquerading dalla rete interna verso la DMZ, semplicemente si rifiuti qualsiasi pacchetto che proviene nell'altro senso. Ovvero, si permettano solo i pacchetti che possono essere parte di una connessione già stabilita.

```
ipchains -A dmz-good -p tcp ! -y -s 192.84.219.128 smtp -j ACCEPT
ipchains -A dmz-good -p udp -s 192.84.219.129 domain -j ACCEPT
ipchains -A dmz-good -p tcp ! -y -s 192.84.219.129 domain -j ACCEPT
ipchains -A dmz-good -p tcp ! -y -s 192.84.218.130 www -j ACCEPT
ipchains -A dmz-good -p tcp ! -y -s 192.84.218.130 rsync -j ACCEPT
ipchains -A dmz-good -p icmp -j icmp-acc
ipchains -A dmz-good -j DENY -l
```

7.4.7 Da DMZ a Bad (esterno).

- Restrizioni di DMZ:
  - Mail server
    - **SMTP verso l'esterno**
    - **Accetta SMTP da** interno ed **esterno**
    - Accetta POP-3 dall'interno
  - Name server
    - **Invia DNS verso l'esterno**
    - **Accetta DNS da** interno, **esterno** e dalla macchina per il filtraggio dei pacchetti
  - Web server
    - **Accetta HTTP da** interno e **esterno**
    - Accesso rsync dall'interno
-
```
ipchains -A dmz-bad -p tcp -s 192.84.219.128 smtp -j ACCEPT
ipchains -A dmz-bad -p udp -s 192.84.219.129 domain -j ACCEPT
ipchains -A dmz-bad -p tcp -s 192.84.219.129 domain -j ACCEPT
ipchains -A dmz-bad -p tcp ! -y -s 192.84.218.130 www -j ACCEPT
ipchains -A dmz-bad -p icmp -j icmp-acc
ipchains -A dmz-bad -j DENY -l
```

7.4.8 Da Bad (esterno) a Good (interno).

- Non si permette niente (non mascherato) dalla rete esterna verso quella interna

```
ipchains -A bad-good -j REJECT
```

7.4.9 Filtraggio dei pacchetti per la macchina Linux stessa

- Se si vuole usare il filtraggio dei pacchetti sui pacchetti in ingresso alla macchina stessa, è necessario fare il filtraggio sulla catena input. Si crei una catena per ogni interfaccia di destinazione:

```
ipchains -N bad-if
ipchains -N dmz-if
ipchains -N good-if
```

- Creare dei salti a queste:

```
ipchains -A input -d 192.84.219.1 -j bad-if
ipchains -A input -d 192.84.219.250 -j dmz-if
ipchains -A input -d 192.168.1.250 -j good-if
```

Interfaccia di Bad (esterno).

- Macchina per il filtraggio dei pacchetti:
  - **PING verso ogni rete**
  - **TRACEROUTE verso ogni rete**
  - Accesso DNS
- L'interfaccia esterna riceve risposte anche per i pacchetti "mascherati" (il masquerading usa le porte sorgente dalla 61000 alla 65095) oltre a errori ICMP per questi e a risposte al PING.

```
ipchains -A bad-if -i ! ppp0 -j DENY -l
ipchains -A bad-if -p TCP --dport 61000:65095 -j ACCEPT
ipchains -A bad-if -p UDP --dport 61000:65095 -j ACCEPT
ipchains -A bad-if -p ICMP --icmp-type pong -j ACCEPT
ipchains -A bad-if -j icmp-acc
ipchains -A bad-if -j DENY
```

Interfaccia di DMZ.

- Restrizioni della macchina per il filtraggio dei pacchetti:
  - **PING verso ogni rete**
  - **TRACEROUTE verso ogni rete**
  - Accesso DNS
- L'interfaccia DMZ riceve risposte DNS, risposte al ping ed errori ICMP.

```
ipchains -A dmz-if -i ! eth0 -j DENY
ipchains -A dmz-if -p TCP ! -y -s 192.84.219.129 53 -j ACCEPT
ipchains -A dmz-if -p UDP -s 192.84.219.129 53 -j ACCEPT
ipchains -A dmz-if -p ICMP --icmp-type pong -j ACCEPT
ipchains -A dmz-if -j icmp-acc
ipchains -A dmz-if -j DENY -l
```

Interfaccia di Good (interno).

- Restrizioni della macchina per il filtraggio dei pacchetti:
  - **PING verso ogni rete**
  - **TRACEROUTE verso ogni rete**
  - Accesso DNS
- Restrizioni della rete interna:
  - Permettere WWW, ftp, traceroute, ssh verso l'esterno
  - **Permettere SMTP verso il Mail server**
  - **Permettere POP-3 verso il Mail server**
  - **Permettere DNS verso il Name server**
  - **Permettere rsync verso il Web server**
  - **Permettere WWW verso il Web server**
  - Permettere il ping alla macchina filtro
- L'interfaccia interna riceve ping, risposte al ping e errori ICMP.

```
ipchains -A good-if -i ! eth1 -j DENY
ipchains -A good-if -p ICMP --icmp-type ping -j ACCEPT
ipchains -A good-if -p ICMP --icmp-type pong -j ACCEPT
ipchains -A good-if -j icmp-acc
ipchains -A good-if -j DENY -l
```

7.5 Per finire

- Cancellare le regole di bloccaggio:

```
ipchains -D input 1
ipchains -D forward 1
ipchains -D output 1
```

8 Appendice: Differenze tra ipchains e ipfwadm

Alcune di queste modifiche sono il risultato di modifiche nel kernel, e altre dipendono dal fatto che `ipchains` sembra differente da `ipfwadm`.

1. Molti argomenti sono stati rimappati: le maiuscole ora indicano un comando e le minuscole indicano un'opzione.
2. Sono supportate catene arbitrarie e quindi anche le catene predefinite ora hanno un nome invece di essere solamente un opzione (eg. 'input' invece di '-I').
3. L'opzione '-k' non c'è più: si usi '! -y'.
4. L'opzione '-b' inserire/aggiunge/cancella veramente due regole, piuttosto che una singola regola 'bidirezionale'.
5. L'opzione '-b' può essere passata a '-C' per fare due verifiche (una in ogni direzione).
6. L'opzione '-x' a '-l' è stata rimpiazzata da '-v'.
7. Non sono più supportate porte di provenienza e destinazione multiple. Spero che l'essere in grado di negare un intervallo di porte venga in aiuto in questi casi.
8. Le interfacce possono essere specificate solamente attraverso il nome (non l'indirizzo). Comunque, la vecchia semantica è stata silenzionamente cambiata nei kernel della serie 2.1.
9. I frammenti sono esaminati, non lasciati passare automaticamente.
10. Sono state rimosse le catene specifiche per l'accounting.
11. Possono essere testati protocolli arbitrari su IP.
12. Il vecchio comportamente del SYN e ACK matching (che in precedenza era ignorato per i pacchetti non TCP) è cambiato; l'opzione SYN non è valida per regole non specifiche per il TCP.
13. I contatori sono ora a 64 bit su macchine a 32 bit, non più a 32 bit.
14. Ora sono supportate le opzioni inverse.
15. Ora sono supportati i codici ICMP.
16. Sono supportati i caratteri jolly nella specifica dell'interfaccia.
17. Ora è controllata l'integrità delle manipolazioni TOS: il vecchio codice del kernel sileziosamente bloccava le manipolazioni (illegali) del bit TOS 'Must Be Zero'; ipchains ora restituisce un errore se ci si prova, come anche per gli altri casi illegali.

8.1 Tabella di riferimento rapido

[Gli argomenti dei comandi sono in MAIUSCOLO, mentre gli argomenti delle opzioni in minuscolo]

Una cosa da notare, il masquerading è specificato da '-j MASQ'; è completamente diverso da '-j ACCEPT', e non è trattato come un mero effetto collaterale, diversamente da quanto fa `ipfwadm`.

```
======================================================================
| ipfwadm      | ipchains                  | Note
----------------------------------------------------------------------
| -A [both]    | -N acct                   | Crea una catena 'acct'
|              |& -I 1 input -j acct       | e fa sì che i pacchetti
|              |& -I 1 output -j acct      | in ingresso e in uscita
|              |& acct                     | la traversino.
----------------------------------------------------------------------
| -A in        | input                     | Una regola senza tattica
----------------------------------------------------------------------
| -A out       | output                    | Una regola senza tattica
----------------------------------------------------------------------
| -F           | forward                   | Si usi questo come [catena]
----------------------------------------------------------------------
| -I           | input                     | Si usi questo come [catena]
----------------------------------------------------------------------
| -O           | output                    | Si usi questo come [catena]
----------------------------------------------------------------------
| -M -l        | -M -L                     |
----------------------------------------------------------------------
| -M -s        | -M -S                     |
----------------------------------------------------------------------
| -a tattica   | -A [catena] -j TATTICA    | (si veda anche -r e -m).
----------------------------------------------------------------------
| -d tattica   | -D [catena] -j TATTICA    | (si veda anche -r e -m).
----------------------------------------------------------------------
| -i tattica   | -I 1 [catena] -j TATTICA| (si veda anche -r e -m).
----------------------------------------------------------------------
| -l           | -L                        |
----------------------------------------------------------------------
| -z           | -Z                        |
----------------------------------------------------------------------
| -f           | -F                        |
----------------------------------------------------------------------
| -p           | -P                        |
----------------------------------------------------------------------
| -c           | -C                        |
----------------------------------------------------------------------
| -P           | -p                        |
----------------------------------------------------------------------
| -S           | -s                        | Accetta solo una porta o
|              |                           | intervallo.
----------------------------------------------------------------------
| -D           | -d                        | Accetta solo una porta o
```

```
|             |                          | intervallo.
-----------------------------------------------------------------------
| -V          | <nessuna>                | Si usi -i [nome].
-----------------------------------------------------------------------
| -W          | -i                       |
-----------------------------------------------------------------------
| -b          | -b                       | Crea 2 regole.
-----------------------------------------------------------------------
| -e          | -v                       |
-----------------------------------------------------------------------
| -k          | ! -y                     | Non funziona finché non
|             |                          | si specifica anche -p tcp.
-----------------------------------------------------------------------
| -m          | -j MASQ                  |
-----------------------------------------------------------------------
| -n          | -n                       |
-----------------------------------------------------------------------
| -o          | -l                       |
-----------------------------------------------------------------------
| -r [redirpt] | -j REDIRECT [redirpt]   |
-----------------------------------------------------------------------
| -t          | -t                       |
-----------------------------------------------------------------------
| -v          | -v                       |
-----------------------------------------------------------------------
| -x          | -x                       |
-----------------------------------------------------------------------
| -y          | -y                       | Non funziona finché non
|             |                          | si specifica anche -p tcp.
-----------------------------------------------------------------------
```

8.2 Esempi di traduzione di comandi ipfwadm

Vecchio comando: ipfwadm -F -p deny

Nuovo comando: ipchains -P forward DENY

Vecchio comando: ipfwadm -F -a m -S 192.168.0.0/24 -D 0.0.0.0/0

Nuovo comando: ipchains -A forward -j MASQ -s 192.168.0.0/24 -d 0.0.0.0/0

Vecchio comando: ipfwadm -I -a accept -V 10.1.2.1 -S 10.0.0.0/8 -D 0.0.0.0/0

Nuovo comando: ipchains -A input -j ACCEPT -i eth0 -s 10.0.0.0/8 -d 0.0.0.0/0

(Si noti che questi non sono equivalenti specificando l'interfaccia tramite l'indirizzo: si usi il nome di interfaccia. In questa macchina 10.1.2.1 corrisponde a eth0).

9 Appendice: Usare lo script ipfwadm-wrapper.

Lo script shell `ipfwadm-wrapper` dovrebbe essere un plug-in di rimpiazzo di `ipfwadm` per compatibilità all'indietro con ipfwadm 2.3a.

La sola caratteristica che non posso veramente gestire è l'opzione '-V'. Quando è usata, è mostrato un avviso. Se è usata anche l'opzione '-W', l'opzione '-V' è ignorata. Altrimenti, lo script prova a trovare il nome di interfaccia associato con quell'indirizzo, usando `ifconfig`. Se fallisce (come nel caso di un'interfaccia non attiva) allora uscirà con un messaggio d'errore.

Questo avviso può essere soppresso o cambiando il '-V' in un '-W', oppure direzionando lo standard output dello script a /dev/null.

Se si trova un qualsiasi errore nello script, o una qualsiasi differenza tra il vero ipfwadm e questo script, *invito* a segnalarmi il bug: si invii un'email a rusty@linuxcare.com con "BUG-REPORT" nel subject. Prego si segnali la versione del vecchio `ipfwadm` (`ipfwadm -h`), la versione di `ipchains` (`ipchains -version`) e quella dello script ipfwadm wrapper script (`ipfwadm-wrapper -version`). Si invii anche l'output di `ipchains-save`. Grazie in anticipo.

Se si mischia `ipchains` con questo script `ipfwadm-wrapper` lo si fa a proprio rischio e pericolo.

10 Appendice: Ringraziamenti.

Molte grazie a Michael Neuling, che ha scritto la prima cosa rilasciabile del codice IP chains mentre lavorava per me. Mi scuso pubblicamente per aver respinto la sua idea del result-caching, che Alan Cox propose un po' dopo e finalmente mi sono deciso a implementare, rendendomi conto dell'errore che avevo commesso.

Grazie ad Alan Cox per il suo supporto tecnico via email 24 ore su 24 e il suo incoraggiamento.

Grazie a tutti gli autori del codice ipfw e ipfwadm, specialmente a Jos Vos. Come nani sulle spalle dei giganti... Ciò vale anche per Linus Torvalds oltre che a tutti gli hacker del kernel e dello userspace.

Un ringraziamento ai diligenti beta tester e bughunter (cacciatori di bug), specialmente a Jordan Mendelson, Shaw Carruthers, Kevin Moule, Dr. Liviu Daia, Helmut Adams, Franck Sicard, Kevin Littlejohn, Matt Kemner, John D. Hardin, Alexey Kuznetsov, Leos Bitto, Jim Kunzman, Gerard Gerritsen, Serge Sivkov, Andrew Burgess, Steve Schmidtke, Richard Offer, Bernhard Weisshuhn, Larry Auton, Ambrose Li, Pavel Krauz, Steve Chadsey, Francesco Potortì e Alain Knaff.

Il tradottore vuole ringraziare Alvise Bellotti per la sue indispensabili correzioni e i suoi suggerimenti.

10.1 Traduzioni

Quelli che hanno fatto le traduzioni dovrebbero mettersi in *cima* alla pagina dei ringraziamenti, in questo modo: «grazie a XXX per aver tradotto esattamente il tutto dal mio inglese». Poi dovrebbero farmi sapere della loro traduzione cosicché io possa includerli qui.

Arnaud Launay, asl@launay.org:

http://www.freenix.fr/unix/linux/HOWTO/IPCHAINS-HOWTO.html `http://www.freenix.fr/unix/linux/HOWTO/IPCHAINS-HOWTO.html`

Giovanni Bortolozzo, borto@pluto.linux.it: *http://www.pluto.linux.it/ildp/HOWTO/IPCHAINS-HOWTO.html*