Linux Firewall ad opera d'arte

PROGRAMMA

Linux Firewalling

Obiettivo:
Comprendere Iptables:
- La sintassi del comando
- La logica di gestione di un pacchetto nel kernel
- I campi di applicazione e le soluzioni possibili
Approfondimenti su funzionalità sperimentali
Sistemi di logging e gestione
Soluzioni di alta affidabilità

Linux firewalling: Introduzione a Iptables

Overview, gestione, utilizzo di iptables su Linux per packet filtering

iptables - Linux natting e packet mangling

Utilizzo di iptables per natting, masquerading e mangling di pacchetti.

Esempi di configurazione di Iptables

Esempi di configurazioni di un firewall Linux con iptables

Iptables Avanzato

Funzionalità avanzate di iptables.

Linux Firewalling

NetFilter e il firewalling su Linux

Il codice di firewalling nel kernel Linux ha una sua storia:
Nel kernel 2.0 (1996) si usava **ipfwadm**
Nel kernel 2.2 (1999) si sono usate le **ipchains**
Dal kernel 2.4 (2001) sono state introdotte le **iptables**
Tutt'ora utilizzate nel ramo 2.6 (2002)

Il framework netfilter/iptables gestisce tutte le attività di firewalling su Linux:
- Netfilter comprende i moduli del kernel
- **iptables** è il comando con cui si gestisce netfilter

Feature principali:
- Packet filtering stateless e stateful
- Supporto IPv4 e IPv6
- Ogni tipo di natting (NAT/NAPT)
- Infrastruttura flessibile ed estendibile
- Numerosi plug-in e moduli aggiuntivi per funzionalità aggiuntive

Sito ufficiale: **http://www.netfilter.org**

Logica di Iptables: tabelle, catene, regole, target

Iptables lavora su 3 tabelle (**tables**) di default:
`filter` - Regola il firewalling: quali pacchetti accettare, quali bloccare
`nat` - Regola le attività di natting
`mangle` - Interviene sulla alterazione dei pacchetti.

Ogni tabella ha delle catene (**chains**) predefinite (**`INPUT`**, **`OUTPUT`**, **`FORWARD`** ...) a cui possono essere aggiunte catene custom.
Ogni catena è composta da un elenco di regole (**rules**) che identificano pacchetti di rete secono criteri diversi (es: **`-p tcp --dport 80 -d 10.0.0.45`**)
Ogni regola termina con una indicazione (**target**) su cosa fare dei pacchetti identificati dalla regola stessa (es: **`-j ACCEPT`**, **`-j DROP`** ...)

iptables - One command to rule them all

Il comando iptables viene usato per ogni attività di gestione e configurazione.

Inserimento regole:
`iptables -A CATENA ...` - Aggiunge una regola alla fine della catena indicata
`iptables -I CATENA [#] ...` - Inserisce alla riga # (default 1) una regola nella catena indicata
`iptables -N CATENA` - Crea una nuova catena custom
`iptables -P CATENA TARGET` - Imposta il target di default per la catena indicata

Rimozione regole e azzeramenti:
`iptables -F [catena]` - Ripulisce tutte le catene (o quella indicata)
`iptables -X [catena]` - Ripulisce tutte le catene custom (o quella indicata)
`iptables -Z [catena]` - Azzera i contatori sulle catene
`iptables -D catena #` - Cancella la regola numero # dalla catena indicata

Interrogazione:
`iptables -L` - Elenca le regole esistenti
`iptables -L -n -v` - Elenca, senza risolvere gli host, in modo verboso le regole esistenti

Firewall con la tabella filter

E' quella implicita e predefinita (`-t filter`)
Riguarda le attività di filtraggio del traffico.
Ha 3 catene di default:
INPUT - Riguarda tutti i pacchetti destinati al sistema. In entrata da ogni interfaccia.
OUTPUT - Riguarda i pacchetti che sono originati dal sistema e destinati ad uscire.
FORWARD - Riguarda i pacchetti che attraversano il sistema, con IP sorgente e destinazione esterni.

Esempio per permettere accesso alla porta 80 locale:
```
iptables -t filter -I INPUT -p tcp --dport 80 -j ACCEPT
```
Analoga a: `iptables -I INPUT -p tcp --dport 80 -j ACCEPT`

Esempio per permettere ad un pacchetto con IP sorgente 10.0.0.4 di raggiungere il server 192.168.0.1 attraversando il firewall:
```
iptables -I FORWARD -s 10.0.0.4 -d 192.168.0.1 -j ACCEPT
```

Gestione e salvataggio regole

Le regole di iptables vengono immediatamente attivate nel momento in cui si inserisce il comando.
Lo stato del firewall rimane fino a quando non si esegue un riavvio.
Il comando **`iptables-save`** stampa su stdout la configurazione corrente del firewall
Con **`iptables-restore`** si può ripristinare una configurazione salvata con iptables-save

Durante l'avvio le impostazioni di iptables possono essere ripristinate con:

- Uno script shell che esegue in successione tutti i comandi iptables necessari. Ha il vantaggio di permettere commenti fra le regole e poter usare variabili, cicli e impostare regole sulla base di parametri variabili.
Spesso lo si richiama da **`/etc/rc.d/rc.local`** ma sarebbe meglio applicarlo appena attivate le interfacce di rete

- Uno script di init.d che tramite iptables-restore ripristina una configurazione precedentemente salvata. E' la soluzione usata da molte distribuzioni, dove il file di configurazione del firewall non è altro che l'output di un comando come: **`iptables-save > /etc/sysconfig/iptables`**. Ha il vantaggio di applicare lunghi elenchi di regole in modo molto più rapido.

Criteri standard di match

Le possibilità di matching di pacchetti secondo diversi criteri sono molte e possono essere abbinate nella stessa regola.
Il carattere ! si usa come negazione (NOT logico):
Criteri di match standard:
`-p [!] proto` - Protocollo IP. Secondo IP number o nome (es: tcp, udp, gre, ah...)
`-s [!] address[/mask]` - Indirizzo IP sorgente (o network con maschera di sottorete)
`-d [!] address[/mask]` - Indirizzo IP destinazione (o network)
`-i [!] interface[+]` - Interfaccia di rete di entrata ([+] wildcard)
`-o [!] interface[+]` - Interfaccia di rete di uscita ([+] wildcard)
`-f` - Frammento di pacchetto

Estensioni per principali protocolli
TCP (**`-p tcp`**)
`--sport port[:port]` - La porta o il range di porte TCP sorgenti (Per un range da 1 a 1024: **`1:1024`**
`--dport port[:port]` - La porta o il range di porte TCP di destinazione
`--tcp-flags flag` - I flag TCP del pacchetto attivi (SYN, ACK, FIN, RST, URG, PSH)
`--syn` - Pacchetti con solo SYN attivo (nuove connessioni)
`--tcp-option option` - Match per specifiche opzioni della intestazione TCP
Esempio per droppare tutti i pacchetti TCP in entrata su porte privilegiate:

```
iptables -I INPUT -p tcp --syn --dport 0:1024 -j DROP
```

UDP (**`-p udp`**)
`--sport port[:port]` - La porta o il range di porte UDP sorgenti (Per un range da 1 a 1024: **`1:1024`**
`--dport port[:port]` - La porta o il range di porte UDP di destinazione
Esempio per permettere pacchetti UDP per traceroute:

```
iptables -I INPUT -p udp --sport 32769:65535 --dport 33434:33523 -j ACCEPT
```

ICMP (**`-p icmp`**)
`--icmp-type type` - Match sul tipo di pacchetto ICMP in formato numerico o completo (**`iptables -p icmp --help`** per un elenco dei tipi ICMP)
Esempio per permettere pacchetti UDP per traceroute:

```
iptables -I INPUT -p udp --sport 32769:65535 --dport 33434:33523 -j ACCEPT
```

Target e jump

Ogni regola termina con la definizione di un TARGET che indica cosa viene fatto del pacchetto.
Molti dei target principali (ACCEPT, DROP, REJECT...) determinano l'interruzione della catena: il pacchetto matchato segue le indicazioni del Target e non vengono considerate le catene successive.
Come target si può anche impostare una catena custom nella quale "saltare" (jump **`-j`**) per procedere nell'attraversamento delle regole.

Target principali
`-j ACCEPT` - Il pachetto matchato viene accettato e procede verso la sua destinazione. Si usa per definire il traffico permesso.
`-j DROP` - Il pacchetto viene rifiutato e scartato, senza alcuna notifica al mittente. Si usa, in alternativa a REJECT, per bloccare traffico.
`-j REJECT` - Il pacchetto viene rifiutato. Al mittente viene mandato un pacchetto (configurabile) di notifica tipo ICMP port-unreachable (**`--reject-with`** ***`icmp-port-unreachable`***)
`-t LOG` - Il pacchetto viene loggato via syslog e procede l'attraversamento della catena. Opzioni: (**`--log-level`**, **`--log-prefix`**, **`--log-tcp-sequence`**, **`--log-tcp-options`**, **`--log-ip-options`**)
`-j DNAT` - Viene modificato l'IP di destinazione del pacchetto. Target disponibile solo in nat / PREROUTING e nat / OUTPUT. L'opzione **`--to-destination`** ***`IP:porta`*** definisce il nuovo IP di destinazione. Si usa tipicamente su network firewall che nattano server di una DMZ
`-j SNAT` - Viene modificato l'IP sorgente. Solo in nat / POSTROUTING. Prevede l'opzione **`--to-source`** ***`IP:porta`***. Si usa per permettere l'accesso a Internet da una rete locale con IP privati.
`-j MASQUERADE` - Simile a SNAT, si applica quando i pacchetti escono da interfacce con IP dinamico (dialup, adsl, dhcp...). Si usa solo in nat / POSTROUTING e prevede l'opzione **`--to-ports`** ***`porte`***.
`-j REDIRECT` - Redirige il pacchetto ad una porta locale. Usabile solo in nat / PREROUTING e nat / OUTPUT è previsto per fare un transparent proxy (con proxy server in esecuzione sulla macchina con iptables)
`-j RETURN` - Interrompe l'attraversamento della catena. Se questa è una secondaria, il pacchetto torna ad attraversare la catena madre da punto in cui aveva fatto il salto nella secondaria. Se il RETURN è in una delle catene di default, il pacchetto interrompe l'attraversamento e segue la policy di default.
`-j TOS` - Usabile solo nella tabella mangle, permette di cambiare il TOS (Type Of Service) di un pacchetto con l'opzione **`--set-tos`**. Per un elenco dei parametri disponibili: **`iptables -j TOS -h`**
`-j MIRROR` - Curioso e sperimentale, questo target invia un pacchetto speculare al mittente. In pratica è come se facesse da specchio per tutti i pacchetti ricevuti. Da usare con cautela, per evitare attacchi DOS indiretti.

Connection tracking con Iptables

Una delle più potenti e benvenute feature di iptables è la gestione del traffico con la consapevolezza dei flussi e delle connessioni dei protocolli usati.
Il modulo **ip_conntrack** di Netfilter si preoccupa di gestire le connessioni e fornire gli strumenti per poterle controllare.
In **/proc/net/ip_conntrack** il kernel mantiene e aggiorna in tempo reale lo stato delle connessioni gestite (il numero massimo è configurabile modificando **/proc/sys/net/ipv4/ip_conntrack_max**.

La gestione del traffico sulla base delle connessioni è fatta richiamando il modulo di match **state** che permette di dividere il traffico secondo diversi stati:
NEW - Il primo pacchetto relativo ad una nuova connessione (syn TCP o nuovo pacchetto UPD)
ESTABLISHED - Pacchetti relativi a connessioni già stabilite, in cui si è avuto almeno un pacchetto da entrambi i peer
RELATED - Pacchetti in qualche modo correlati a connessioni esistenti ed established. Ticipi esempi il traffico di dati FTP o il trasferimetno DCC in IRC.
INVALID - Pacchetti che non rientrano in alcuno dei suddetti stati, di solito vengono droppati.

Il match **state** risulta particolarmente comodo per gestire il "traffico di ritorno" per connessioni accettate.
Ad esempio, per permettere su un host, tutto il traffico in uscita e, in entrata, solo il traffico correlato a quanto uscito bastano regole tipo:

```
iptables -I INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
itables -I OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
```

Per gestire lo stato di connessione di alcuni protocolli particolari, esistono dei moduli speciali adibili sia alla gestione del connection tracking, sia al natting:
ip_conntrack_ftp - **ip_nat_ftp** : Conntrack e nat per FTP
ip_conntrack_h323 - **ip_nat_h323** : Conntrack e nat per H323
ip_conntrack_irc - **ip_nat_irc** : Conntrack e nat per IRC
ip_conntrack_rtsp - **ip_nat_rtsp** : Conntrack e nat per IRC
ip_conntrack_tftp - **ip_nat_tftp** : Conntrack e nat per TFTP
ip_conntrack_rpc_tcp - **ip_conntrack_rpc_udp** : Conntrack per RPC

Tuning dei parametri del kernel

Tramite l'interfaccia system control è possibile settare, in tempo reale, numerosi parametri che influenzano il tuning del networking.

Esistono diversi modi per intervenire su sysctl:
1- Usare normali comandi come cat e echo per visualizzare e modificare parametri all'interno di /proc/sys. Es:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
cat /proc/sys/net/ipv4/ip_forward
```

2- Usare il comando sysctl per visualizzare o settare (con opzione -w) i parametri:

```
sysctl -w net.ipv4.ip_forward="1"
sysctl net.ipv4.ip_forward
```

3- Usare il file di configurazione **/etc/sysctl.conf** che all'avvio viene caricato automaticamente e contiene righe come:

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

Iptables sulle distribuzioni Linux

La gestione di iptables cambia su diverse distribuzioni Linux.
Non è indispenabile usare la logica di una specifica distribuzione: si possono applicare le regole di iptables con metodi propri, disattivando quelli di default.

RedHat - Fedora (e derivate)
Il firewalling è gestito con il servizio **iptables**, gestibile tramite l'init script `/etc/init.d/iptables`. La configurazione delle regole è, in formato iptables-save, in `/etc/sysconfig/iptables`.

SuSE
Yast2 permette di configurare un firewall con una discreta varietà di possibilità. Il file di configurazione è in un formato proprio, orientato alla funzione e senza riferimenti diretti alle regole iptables da applicare: `/etc/sysconfig/SuSEfirewall2`.

Mandriva (Mandrake)
Dal Mandrake Control Center il tool **DrakDirewall** permette una comoda configurazione.

Debian
Debian non fornisce una soluzione di default per gestire iptables. Mette a disposizione diversi strumenti che facilitano la creazione e la gestione delle regole.
Questi programmi sono disponibili e a volte previsti di default anche su altre distribuzioni (**Firestarter**, **Fwbuilder**, **Shorewall**, **Guarddog**, **Knetfilter**, **Lokkit**...)

Gentoo
E' fornito uno script di gestione del firewall `/etc/init.d/firewall` che contiene comandi iptables da applicare in funzione del file di configurazione (custom) `/etc/conf.d/firewall`. Comode funzionalità di accounting con il comando `accounting-report`.

Tool di configurazione di Iptables

Sono disponibili diversi strumenti che rendono più semplice la gestione e la configurazione di Iptables. A volte questi sono installati e integrati di default su una distribuzione, a volte vanno installati autonomamente.

Shorewall - `http://www.shorewall.net/` - Potente strumento di configurazione, che sulla base di propri file di configurazione permette la creazione di regole iptables complesse. Meno semplice di altri strumenti si rivela più adeguato per scenari evoluti.

FwBuilder - `http://www.fwbuilder.org/` - Ottimo tool grafico di configurazione con supporto di firewall diversi (iptables, pix, ipfilter, ipf..). Ha un aspetto e una logica che ricorda quello di Checkpoint e permette anche configurazioni complesse.

Firestarter - `http://www.fs-security.com/` - Semplice e comodo tool grafico di configurazione, gestione e monitoraggio del traffico. Non adeguato ad usi evoluti, è l'ideale come personal firewall o per utenti non esperti.

FireHol - `http://firehol.sourceforge.net/` - Similmente a Shorewall prevede propri file di configurazione, con una logica semplificata, con cui costruisce catene iptables complesse.

KMyFirewall - `http://kmyfirewall.sourceforge.net/` - Tool per KDE che permette una configurazione relativamente semplice ma senza rinunciare al dettaglio e alla "consapevolezza" sulle regole implementate.

Lokkit - E' il tool di default con cui viene gestita la configurazione su RedHat (disponibile anche per altre distro) . Molto semplice permette una configurazione essenziale, ma non ha le funzioni di monitoring e gestione di un Firestarter.

WebMin - `http://www.webmin.org/`Webmin ha un modulo per la gestione diretta di iptables (e uno anche per gestire Shorewall). Richiede comunque una buona comprensione della logica di iptables, ma presenta in modo comodo le opzioni disponibili.

Distribuzioni Firewall dedicate

Esistono varie distribuzioni esplicitamente dedicate per un firewall.
Possono essere LiveCD o installate sul sistema e generalmente prevedono:
- Kernel con iptables e a volta patch sperimentali
- Interfaccia grafica di gestione (generalmente via web)
- Server DHCP per assegnare IP alla rete interna
- Software per VPN varie (IPSec, PPTP)
- Software di Intrusion detection e monitoring della sicurezza
- Proxy filtering e eventuali filtri sulla posta

Le più significative sono:
Astaro Security Linux - **http://www.astaro.com/**
Distribuzione commerciale con comoda interfaccia web e notevoli funzionalità: firewall, VPN, antivirus, antispam, content filtering, antispyware, intrusion detection.

SmoothWall - **http://smoothwall.org/**
Storica distribuzione per firewall con la versione Express (free) e versioni commerciali. Supporto VPN, antivirus, antispam, content filtering, antispyware, intrusion detection.

Clark Connect - **http://www.clarkconnect.com/**
Esplicitamente dedicata per firewall, lan gateway prevede supporto VPN, antivirus, antispam, content filtering, antispyware, intrusion detection oltre alla gestione di normali servizi di rete.

IpCop - **http://www.ipcop.org/**
Funzionale e completa, con bella interfaccia web e gestione traffic shaping e VPN.

Devil Linux - **http://www.devil-linux.org**
Può essere caricata direttamente da CD o USB. Ha un kernel hardened con GRSecurity e una interfaccia testuale per la gestione del sistema.

Monowall - **http://www.m0n0.ch/**
Si cita per riferimento, anche se non è basata su Linux ma su FreeBSD. Ha una comoda interfaccia, supporto DHCPd, IPSEC, PPTP, SNMP. Notare che usa **ipfilter** e non iptables.

Overview di Netfilter e Iptables

Tipo Infobox: **DESCRIPTION** - Skill Level: **2- JUNIOR** - Autore: **Giorgio 'neo' Colombo** - Ultimo Aggiornamento: **2005-05-14 23:11:57**

Netfilter è il framework con cui vengono gestite tutte le operazioni di firewalling, natting e manipolazione di pacchetti nel kernel Linux. Netfilter viene gestito con il comando **iptables** con cui si determinano e configurano tutte le regole con cui gestire il traffico di rete.

Breve storia dei firewall Linux
Il codice di firewalling di Linux ha subito diverse modifiche nella storia del kernel.
Nella versione 2.0.x viene usato il sistema **ipfwadm** che viene sostituito nelle versione 2.2.x dalle **ipchains** che introducono il concetto di catene di regole (access-list) a cui un pacchetto può essere indirizzato.
Dalla versione 2.4 in poi il kernel si basa sull'infrastruttura **netfilter/iptables** che permette con semplicità estrema di configurare un firewall di fare packet filtering (stateless, statefull), diversi tipi di NAT (Network Address Translation) e packet mangling (modifica di alcuni flag dei pacchetti trattati).
Il progetto NETFILTER/IPTABLES nasce da Paul "Rusty" Russell e dal Core Team (i quali tutt'ora continuano il lavoro di sviluppo), nel lontano 1999.

Componenti
Il framework netfilter/iptables si può identificare in diversi componenti:
1- la parte riguardante il kernel, ovvero tutto il codice che lavora in kernel space e che con diversi moduli introduce funzionalità base e avanzate.
In quasi tutte le distribuzioni netfilter è installato, in alcune specifiche sulla sicurezza possono essere anche installati moduli aggiuntivi sperimentali.
2- L'interfaccia utente, ovvero sostanzialmente il comando `iptables` che permette di impostare regole e gestire il firewall.
Solitamente fornita con un pacchetto chiamato... iptables.

LINK: Home Page del progetto Netfiltering/Iptables - http://www.netfilter.org/
HOWTO: Elenco di How-To ufficiali Netfiltering/Iptables - http://www.netfilter.org/documentation/index.html#HOWTO
LINK: Info, link utili da linuxGuruz.org - http://www.linuxguruz.org/iptables/

Logica di Iptables

Tipo Infobox: **DESCRIPTION** - Skill Level: **2- JUNIOR** - Autore: **Alessandro 'al' Franceschi** - Ultimo Aggiornamento: **2005-05-15 13:32:22**

E' essenziale capire e familiarizzare con la logica di iptables prima di cimentarsi nella stesura delle policy di un firewall, altrimenti si rischia di non ottenere i risultati voluti o comprometterne la funzionalità.

Logica di tabelle, catene, regole.
Di default iptables lavora su 3 **tabelle (tables)** (filter, nat, mangle) che prevedono diverse **catene (chains)** (INPUT, OUTPUT, FORWARD, PREROUTING, POSTROUTING...) all'interno delle quali esistono elenchi di **regole (rules)**.
Ogni regola identifica pacchetti di rete secondo diversi criteri sulla base di variegati **match** (tcp, udp, state, pkttype e molti altri, alcuni sperimentali) con le relative opzioni e termina con un **target** che determina cosa fare del pacchetto matchato (ACCEPT, DROP, REJECT, SNAT, DNAT, LOG...).

Tabella di filter
Quando si usa il comando iptables questa è la tabella di default, implicita.
Prevede 3 catene di default che di fatto corrispondono alle possibili destinazioni del pacchetto rispetto alla macchina che funziona da firewall:
`INPUT` Catena in cui passano tutti i pacchetti in entrata dalle interfacce di rete del sistema. Si usa comunemente per permettere o negare l'accesso da IP remoti a porte locali.
`OUTPUT:` Considera tutti i pacchetti che sono originati ed escono dal proprio sistema. Si usa per controllare il traffico in uscita, sia per le risposte a pacchetti permessi in INPUT che per nuove connessioni in uscita.
`FORWARD:` Riguarda i pacchetti che devono "attraversare" il firewall, avendo IP sorgente e destinazione diversi da quello della macchina su cui gira iptables.
Si usa tipicamente su router o firewall di rete.

Tabella di nat
Questa tabella riguarda alcune alterazioni che si possono fare su un pacchetto, in particolare la variazione degli IP e delle porte sorgenti o di destinazione . Viene richiamata con l'opzione `-t nat` e prevede le seguenti catene di default:
`PREROUTING` - Catena in cui vengono processati i pacchetti in ingresso, prima che il sistema prenda le decisioni di routing. Si usa tipicamente per fare destination natting usando il target `DNAT`.
`POSTROUTING` - Catena in cui passano i pacchetti dopo che sono stati routati sulla interfaccia di destinazione. Si usa per modificare l'IP e/o la porta sorgente di un pacchetto, tipicamente usato su default gateway di una rete con funzioni di Port Address Translation (su Linux definito Masquerading). Prevede i target `SNAT` e `MASQUERADE`.
`OUTPUT` - Catena utilizzata per modificare l'IP sorgente di un pacchetto uscito dal sistema stesso.

Tabella di mangle
Si può utilizzare per modificare vari parametri negli header di un pacchetto. Viene richiamata con `-t mangle` e prevede le catene di default PREROTING, INPUT, FORWARD, OUTPOUT, POSTROUTING. L'uso più comune è per alterare il TOS di un pacchetto IP. Non ci si ritroverà ad usarla spesso.

Il comando iptables
Iptables è sia il nome dell'infrastruttura a catene con cui si definiscono le policy di firewalling di un sistema basato su netfilter che il comando, utilizzabile da shell, con cui si configura e gestisce il tutto.
La sua sintassi è piuttosto variegata e prevede opzioni e impostazioni anche sulla base dei moduli aggiuntivi supportati e permette di: azzerare le catene esistenti, aggiungere regole, rimuovere regole, impostare le policy di default, creare nuove catene custom, azzerare i

contatori ecc.
Comodi comandi ausiliari sono **iptables-save** e **iptables-restore** che si usano rispettivamente per salvare e ripristinare le configurazioni del firewall.
Le impostazioni sono immediatamente applicate sul sistema fino al momento in cui non si riavvia, per ripristinarle dopo un reboot, è necessario salvare su un file.
Le distribuzioni Linux possono avere file diversi di configurazione e diversi script di avvio, con cui le iptables sono gestite come un servizio, ma la sintassi del comando iptables resta comunque comune.

Attraversamento delle catene
La sequenza con cui vengono processate le varie catene dal Kernel ha questa logica, nel caso si faccia forwarding:
RETE A - MANGLE PREROUTING - NAT PREROUTING - ROUTING - FILTER FORWARD - ROUTING - RETE B
Oppure, per pacchetti in entrata o in uscita:
RETE A - MANGLE PREROUTING - NAT PREROUTING - ROUTING - FILTER INPUT - LOCAL PROCESS - MANGLE OUTPUT - NAT OUTPUT - FILTER OUTPUT - ROUTING - RETE B

I pacchetti attraversano le catene che sono destinati a percorrere, secondo l'ordine delle regole impostate. Se un pacchetto matcha le condizioni definite in una regola, allora segue le indicazioni specificate nel target (ACCEPT, DROP, DNAT) e , in molti casi, interrompe l'attraversamento. Se non matcha nessuna regola di una catena, segue la policy di default impostata per quella catena.
Per questo motivo è fondamentale l'ordine con cui sono inserite le regole in una catena.

iptables

Tipo Infobox: **COMMANDS** - Skill Level: **3- INTERMEDIATE** - Autore: **Giorgio 'neo' Colombo** - Ultimo Aggiornamento: **2005-05-23 14:29:18**

Iptables è il comando utilizzato per settare, modificare e cancellare rule riguardanti il packet filtering del kernel linux.
Di seguito verranno riportati le principali opzioni e le sintassi più utilizzate:

Aggiunge una regola alla fine della catena scelta oppure la modifica o cancella:
```
iptables -[ADC] chain rule-specification [options]
```

Sostituisce regola o la inserisce all'inizio della catena
```
iptables -[RI] chain rulenum rule-specification [options]
```

Visualizza, svuota o azzera i contatori della catena selezionata:
```
iptables -[LFZ] [chain] [options]
```

Crea o aggiunge una nuova catena
```
iptables -[NX] chain
```

Setta la policy di default della catena
```
iptables -P chain target [options]
```

COMANDI
-A, --append Aggiunge una regola in fondo alla catena
-D, --delete Cancella una regola
-R, --replace Esegue il replace di una regola
-I, --insert Inserisce la regola all'inizio della catena o alla posizione indicata.
-L, --list Visulizza l'elenco delle regole inserite
-F, --flush Svuota le catene predefinite
-Z, --zero Azzera i contatori di ogni catena
-N, --new-chain Crea una nuova catena custom
-X, --delete-chain Cancella una catena (creata dall'utente)
-P, --policy Setta la policy di default per una catena

PARAMETRI
-p ,--protocol [protocol] Specifica su quale protocollo deve matchare la regola
-s, --source address[/mask], -src Specifica l'IP sorgente.
-d, --destination address[/mask], -dst Specifica l'IP destinazione
-j, --jump Specifica il target a con cui gestire il pacchetto matchato: può essere una catena custom o un target esistente (drop,accept etc..)
-i, --interface Specifica da quale interfaccia è in entrata un pacchetto. Opzione valida solo per INPUT, FORWARD e PREROUTING.
-o, --out-interface Specifica l'interfaccia d'uscita di un pacchetti. Opzione valida per FORWARD, OUTPUT e POSTROUTING.

TARGET
DROP I pacchetti che "matchano" la regola vengono droppati.
ACCEPT I paccheti che "matchano" la regola vengono fatti passare.
RETURN Il pacchetto smette di attraversare la catena e passa a quella successiva o segue il comportamente di default
QUEUE Se il kernel lo supporta il pacchetto passa a livello user-space dove può essere manipolato da programmi vari.
REJECT Il pacchetto viene rifiutato con un messaggio di notifica al mittente (es: icmp destination unreachable)
LOG Il pacchetto viene loggato via syslog.
MARK Il pacchetto viene marcato per essere gestito da programmi in user space.
MASQUERADE L'IP del pacchetto viene mascherato.
MIRROR Viene rimandato al mittente un pacchetto speculare a quello ricevuto
REDIRECT Il pacchetto viene redirezionato ad una porta locale

ALTRE OPZIONI
-v --verbose Abilita il verbose mode

-n Abilita la visualizzazione numerica, senza DNS reverse lookup
--linenumbers Quando viene eseguito il list delle regole viene aggiunto un numero ad ogni regola che corrisponde la posizione della regola nella sua catena.
! Inverte il significato dell'opzione che lo segue

sysctl

Tipo Infobox: **COMMANDS** - Skill Level: **4- ADVANCED** - Autore: **Giorgio 'neo' Colombo** - Ultimo Aggiornamento: **2002-10-03 18:07:14**

Comando su Linux che permette di configurare in run-time e visualizzare alcuni parametri del kernel.
Particolarmente utile per il tuning del sistema, in particolare:
- Tuning dei Parametri della Network
- Tuning dei Parametri della memoria virtuale
- Tuning dei Parametri del filesystem

Per verificare le opzioni:
```
sysctl [-n] [-e] variable ...
sysctl [-A]
```

Per modificare le opzioni:
```
sysctl [-n] [-e] -w variable=values
```

Per rileggere il file di condfigurazione:
```
sysctl [-n] [-e] -p  [file di configurazione]
```

-A Printa in stdout tutti i valori attualmente disponibili
-n Disabilita il print in stdout del nome della variabile
-e Ignora eventuali errori nel nome della variabile
-w Opzione per modificare il settaggio di sysctl
-p Carica le opzioni di sysctl da un file. Default /etc/sysctl.conf

/etc/sysctl.conf

Tipo Infobox: **PATH** - Skill Level: **3- INTERMEDIATE** - Autore: **Giorgio 'neo' Colombo** - Ultimo Aggiornamento: **2004-04-01 12:13:42**

File di configurazione contenente le opzioni del kernel modificabili in run-time.
Di seguito verra' riportato un esempio di questo file con alcune opzioni abilitate di default da RedHat.
E' fondamentale abilitare net.ipv4.ip_forward (forwarding dei pacchetti IP) se il proprio Linux deve fare da firewall/router.

Di fatto vengono modificati i parametri contenuti nel /proc filesystem.
Abilitando (1) ,disabilitando (0), o il Tuning di alcuni parametri del kernel:
Precisamente si possono modificare i valori del kernel in run-time per i seguenti campi
- Virtual memory
- File System
- Network

Ecco il cat del file in configurazione RedHat Standard:
```
[neo@dido neo]$ cat /etc/sysctl.conf
# Disables packet forwarding
net.ipv4.ip_forward = 0
# Enables source route verification
net.ipv4.conf.default.rp_filter = 1
# Disables the magic-sysrq key
kernel.sysrq = 0
```

A causa di un bug di alcuni router, inoltre, se l'ECN (Explicit Congestation Notification) è abilitato, ciò che sta dietro a tali router NON sarà raggiungibile.
Il tutto si risolve disabilitando l'ECN nel kernel, intervenendo in /etc/sysctl.conf con una riga tipo:
```
net.ipv4.tcp_ecn= 0
```

Network address translation con la tabella nat

La tabella nat (`-t nat`) si usa per modificare IP e porte sorgenti e destinazione.
Ha 3 catene di default:
PREROUTING - Viene attraversata prima che il sistema decida come fare routing. Si usa per fare DESTINATION NATTING (target **DNAT**)
POSTROUTING - Viene usata dopo che il sistema ha deciso su che interfaccia fa uscire un pacchetto. Si usa per SOURCE NATTING (target **SNAT**).
OUTPUT - Si applica a pacchetti in uscita dal sistema stesso per eventuali modifica dell'IP sorgente.

Esempio per nattare una rete completa con un unico IP pubblico (PAT) su interfaccia dialup:

```
iptables -t nat -I POSTROUTING -s 10.0.0.0/24 -o ppp0 -j MASQUERADE
```

Analogo esempio preferibile su interfacce con ip fisso (es: 80.80.14.14):

```
iptables -t nat -I POSTROUTING -s 10.0.0.0/24 -j SNAT --to-source 80.80.14.14
```

Esempio per nattare un host con IP privato su IP pubblico :

```
iptables -t nat -I PREROUTING -d 213.198.151.5 -j DNAT --to-dest 10.10.10.19
```

Alterazione di pacchetti con la tabella mangle

La tabella mangle (`-t mangle`) permette la modifica di vari header IP o TCP di un pacchetto.
Viene tipicamente usata per alterare il TOS e prevede le catene di default: PREROTING, INPUT, FORWARD, OUTPOUT, POSTROUTING.
Ha usi rari e sporadici come il tentativo di ottimizzare la velocità del traffico in uscita impostando il TOS o, più in voga, il DSCP di un pacchetto.

Esempio di modifica TOS per ridurre la latenza di pacchetti DNS:

```
iptables -t mangle -A OUTPUT -p udp --dport 53 -j TOS --set-tos Minimize-Delay
```

Esempio di impostazione del massimo throughput per protocolli di trasferimento file:

```
iptables -t mangle -A FORWARD -p tcp -m multiport --dport 21,20,43,88,109,110,143,873 -j TOS --set-tos Maximize-Throughput
```

Logica di attraversamento di iptables

Il kernel gestisce ogni pacchetto secondo logiche e seguenze ben precise, che determinano il modo con cui vanno processate, e configurate, le iptables.

Pacchetti destinati al sistema locale
1 - Un pacchetto entra da una interfaccia (es: eth0)
2 - Attraversa la tabella/catena **mangle / PREROUTING** (set TOS ecc)
3 - Attraversa la tabella/catena **nat / PREROUTING** (DNAT)
4 - Viene fatta la decisione di routing sulla base dell'IP destinazione attuale.
5 - Attraversa la tabella/catena **mangle / INPUT**
6 - Attraversa la tabella/catena **filter / INPUT**
7 - Viene processato da una applicazione locale (in userland)

Pacchetti in uscita dal sistema locale
1 - Il pacchetto viene processato/generato da una applicazione locale
2 - Il kernel decide dove routare, sulla base dell'IP destinazione.
3 - Attraversa la tabella/catena **mangle / OUTPUT**
4 - Attraversa la tabella/catena **nat / OUTPUT**
5 - Attraversa la tabella/catena **filter / OUTPUT**
6 - Attraversa la tabella/catena **mangle / POSTROUTING**
7 - Attraversa la tabella/catena **nat / POSTROUTING** (SNAT)
8 - Esce dall'interfaccia per raggiungere la sua destinazione (es: eth1)

Pacchetti che attraversano il firewall
1 - Un pacchetto entra da una interfaccia (es: eth0)
2 - Attraversa la tabella/catena **mangle / PREROUTING**
3 - Attraversa la tabella/catena **nat / PREROUTING**
4 - Viene routato verso l'IP di destinazione (in questo caso remoto).
5 - Attraversa la tabella/catena **mangle / FORWARD**
6 - Attraversa la tabella/catena **filter / FORWARD**
7 - Attraversa la tabella/catena **mangle / POSTROUTING**
8 - Attraversa la tabella/catena **nat / POSTROUTING**
9 - Esce dall'interfaccia per raggiungere la sua destinazione (es: eth1)

Linux Network Address Translation

Tipo Infobox: **DESCRIPTION** - Skill Level: **3- INTERMEDIATE** - Autore: **Giorgio 'neo' Colombo** - Ultimo Aggiornamento: **2004-06-04 10:47:19**

NAT ovvero Network Address Tranlation permette di manipolare un pacchetto IP modificando l'indirizzo IP sorgente o di destinazione. Ovviamente il dispositivo che esegue il NAT quando riceve il pacchetto di ritorno esegue l'operazione inversa, sulla base di una tabella di natting che si tiene in memoria.

Il Nat viene utilizzato in varie occasioni, dove si ha la necessità ad esempio di redirezionare il traffico su un unico IP, oppure forwardare il traffico con una certa porta di destinazione ad un'altro host oppure su un'altra porta.
Su Linux si usa definire il natting con due modalità specifiche:
SNAT: Source NAT, cioè l'alterazione dell'IP sorgente del primo pacchetto che apre la connessione. Avviene sempre dopo che il pacchetto ha subito il routing (post-routing).
Un esempio di SNAT è il **masquerading**, con cui tutti gli IP sorgenti di una rete locale vengono convertiti in un unico IP sorgente (del dispositivo che fa masquerading) con cui i pacchetti vengono mandati in rete.
DNAT: Destination NAT, cioè l'alterazione dell'IP di destinazione del primo pacchetto.
A differenza del SNAT il DNAT avviene sempre prima che il pacchetto subisca il routing (pre-routing).
Una forma di DNAT è il port-forwarding e trasparent proxy.
Per quanto riguarda iptables, le catene (chain) da considerare per i vari tipi di NAT sono:
PREROUTING (DNAT, per i pacchetti in arrivo).
Esiste inoltre un "caso speciale" chiamato redirection. E' un DNAT effettuato esclusivamente sull'interfaccia di ingresso del pacchetto. Ovvero si può eseguire un redirect di tuti i pacchetti provenienti su eth0 con destination port 80 e redirezionarli sempre su eth0 ma sulla porta 12345.
OUTPUT (DNAT, per i pacchetti generati della macchina locale)
POSTROUTING (SNAT, per i pacchetti in uscita)

Esempi di NAT:
Port forwarding
```
iptables -A PREROUTING -t nat -p tcp -d 10.0.0.150 --dport 8080  -j DNAT --to 172.16.1.128:80
```
Ovvero tutti i pacchetti che hanno destinazione 10.0.0.150 sulla porta 8080vengono riderizionati al'ip 172.16.1.128 alla porta 80.
Source Natting
```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 10.0.0.1
```
Tutti i pacchetti che escono dall'interfaccia eth0, subiscono una variazione dell'ip sorgente in 10.0.0.1
```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 10.0.0.1-10.0.0.254
```
Come sopra, con la differenza che l'ip sorgente può essere alterato sia in 10.0.0.1 o 10.0.0.154
Masquerading
```
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```
Tutto ciò che passa in uscita dal proprio modem viene mascherato con l'ip pubblico assegnato dal proprio ISP.
Destination Natting
```
iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth1 -j DNAT --to 10.0.0.1:8080
```
Tutti i pacchetti TCP, arrivati dall'interfaccia eth1 con destinazione porta 80 vengono "dirottati" all'ip 10.0.0.1 alla porta 8080.

LINK: How-to NAT in italiano - http://www.netfilter.org/documentation/HOWTO/it/NAT-HOWTO.txt

Masquerading e destination natting con Linux - Esempio

Tipo Infobox: **DESCRIPTION** - Skill Level: **4- ADVANCED** - Autore: **Alessandro 'al' Franceschi** - Ultimo Aggiornamento: **2005-05-22 20:39:41**

Segue un esempio di script che esegue il masquerading di una rete locale e il destination natting di un server PPTP interno a dei limitati IP pubblici.
Può essere modificato e adattato ai propri scopi.

```
#!/bin/sh
#
# masquerading.sh - Version 20040531 - Coresis
#
#### DEBUGGING ###
set -x

### FLUSHING CHAIN ### Azzera e pulisce ogni regola esistente
/sbin/iptables -F
/sbin/iptables -F -t nat
/sbin/iptables -X
/sbin/iptables -Z

### DEFAULT CHAIN ### Imposta le policy di default
/sbin/iptables -P INPUT DROP
/sbin/iptables -P FORWARD DROP
/sbin/iptables -P OUTPUT DROP
/sbin/iptables -t nat -P POSTROUTING DROP
/sbin/iptables -t nat -P PREROUTING DROP

### SETTING IPFORWARDING ### Abilita il forwarding di pacchetti non locali - FONDAMENTALE
/bin/echo "1" > /proc/sys/net/ipv4/ip_forward

### DISABLE RESPOND TO BROADCAST ### Non risponde ai ping inviati al browadcast della subnet
/bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

### ENABLE BAD ERROR MESSAGE PROTECTION ### Ignora finti messaggi di errore ICMP
```

```
/bin/echo "1" > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses

### DISABLE ICMP REDIRECT ACCEPTANCE ### Non accetta pacchetti ICMP di route redirection
/bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_redirects

### SETTING ANTISPOOFING PROTECTION ###
/bin/echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter

### DON'T RESPOND TO BROADCAST PINGS ###
/bin/echo "1" > /proc/sys/net/ipv4/conf/all/log_martians

### Qui vengono definite alcune variabili che successivamente sono usate nelle regole - MODIFICARE SECONDO I PROPRI PARAMETRI
# External Public Interface
EXTIF="eth1"
# Internal Private Interface
INTIF="eth0"
# Host Public IP (su eth1)
EGO="211.121.111.111"
# Internal LAN IP (su eth0)
LANIN="10.0.0.0/24"
# Trusted public network (da cui si permettono collegamenti SSH)
TRUSTED="13.18.151.0/24"
# IP/rete di un utente esterno abilitato a connetterci al server VPN interno
USER="112.56.10.32/28"
# IP Interno del server VPN
VPNSERVER="10.0.0.77"
# RFC IPs Classi di indirizzi dedicate a utilizzi privati o particolari e non routate su Internet
LOOPBACK="127.0.0.0/8"
CLASS_A="10.0.0.0/8"
CLASS_B="172.16.0.0/12"
CLASS_C="192.168.0.0/16"
CLASS_D_MULTICAST="224.0.0.0/4"
CLASS_E_RESERVED_NET="240.0.0.0/5"

# ANTISPOOF Adesso iniziano le regole vere e proprie. Le prime sono generiche regole anti-spoof per IP noti dall'interfaccia pubblica.
/sbin/iptables -A INPUT -i $EXTIF -s $EGO -j DROP
/sbin/iptables -A INPUT -i $EXTIF -s $CLASS_A -j DROP
/sbin/iptables -A INPUT -i $EXTIF -s $CLASS_B -j DROP
/sbin/iptables -A INPUT -i $EXTIF -s $CLASS_C -j DROP
/sbin/iptables -A INPUT -i $EXTIF -s $CLASS_D_MULTICAST -j DROP
/sbin/iptables -A INPUT -i $EXTIF -s $CLASS_E_RESERVED_NET -j DROP
/sbin/iptables -A INPUT -i $EXTIF -d $LOOPBACK -j DROP

# LOOP RULE Permettiamo il traffico di loopback
/sbin/iptables -A INPUT -i lo -j ACCEPT
/sbin/iptables -A OUTPUT -o lo -j ACCEPT

# LAN IN ACCESS Regole che permettono l'accesso al firewall Linux dagli IP della rete Interna - Potrebbero essere più ristrette e limitarsi all'IP dell'amministratore
/sbin/iptables -A INPUT -i $INTIF -s $LANIN -j ACCEPT
/sbin/iptables -A OUTPUT -o $INTIF -d $LANIN -j ACCEPT

# LAN IN OUT Seguono le regole che gestiscono il masquerading della rete interna
/sbin/iptables -A FORWARD -s $LANIN -d 0/0 -j ACCEPT Forwarda tutti i pacchetti dalla rete interna a qualsiasi destinazione
/sbin/iptables -A FORWARD -m state --state ESTABLISHED,RELATED -d $LANIN -j ACCEPT Permette il forwarding di tutti i pacchetti correlati a comunicazioni esistenti
/sbin/iptables -t nat -A POSTROUTING -o $EXTIF -s $LANIN -j MASQUERADE Maschera gli IP sorgenti Interni con l'IP dell'interfaccia pubblica

# GENERAL Regole generali per permettere all'host locale di collegarsi a IP remoti e ricevere i pacchetti di risposta (Nota: si riferiscono alle attività che vengono fatte direttamente dalla macchina Linux locale e non dagli host che la usano come firewall)
/sbin/iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
/sbin/iptables -A OUTPUT -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

# SSH Regole per permettere l'accesso al server SSH locale da un IP esterno fidato precedentemente indicato
/sbin/iptables -A INPUT -s $TRUSTED -p TCP --dport 22 -j ACCEPT

# VPN NATTING Esegue un DNAT di un server VPN che usa pptp (TCP porta 1723) e GRE (IP type 47) e permette l'accesso al server solo da un IP sorgente definito
/sbin/iptables -A FORWARD -s $USER  -p TCP --dport 1723 -j ACCEPT
/sbin/iptables -t nat -A PREROUTING -d $EGO -p tcp --dport 1723 -j DNAT --to-dest $VPNSERVER:1723
/sbin/iptables -t nat -A PREROUTING -d $EGO -p 47 -i eth1 -j DNAT --to-dest $VPNSERVER

# LOGGING Log di tutti i pacchetti, esclusi i broadcast, prima di essere droppati dalla regole di default. I logging viene fatto con livello debug per isolarlo da altri log di sistema. Per cui è necessario scrivere in /etc/syslog.conf una riga tipo:
kern.debug        /var/log/iptables.log
/sbin/iptables -A INPUT -m pkttype --pkt-type ! broadcast -j LOG --log-level=DEBUG --log-prefix="[INPUT DROP] : "
/sbin/iptables -A OUTPUT -m pkttype --pkt-type ! broadcast -j LOG --log-level=DEBUG --log-prefix="[OUTPUT DROP] : "
```

Scenario comune: Personal Firewall

Un Personal Firewall, se non ha servizi da esporre, può avere una configurazione essenziale: permette ogni pacchetto in uscita e accetta in entrata solo i pacchetti di ritorno:

```
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

Configurazioni più elaborata possono prevedere logging dei pacchetti droppati:

```
iptables -A INPUT -m pkttype --pkt-type ! broadcast -j LOG --log-prefix "INPUT DROP: "
iptables -A FORWARD -m pkttype --pkt-type ! broadcast -j LOG --log-prefix "FORWARD DROP: "
```

Controllo, se match owner è supportato, degli utenti autorizzati a far traffico in rete:

```
iptables -A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -m owner --uid-owner 501 -j ACCEPT
```

Scenario comune: Server firewall

Se applicate su un host che eroga un servizio pubblico, è bene bloccare tutto di default e abilitare il traffico in uscita:

```
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

Il traffico permesso in entrata, ad esempio per un server web:

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

Eventuale traffico di servizio da IP limitati (ftp, ssh):

```
iptables -A INPUT -p tcp --dport 22 -s 213.25.10.0/24 -j ACCEPT
iptables -A INPUT -p tcp --dport 21 -s 143.20.12.7 -j ACCEPT
```

Stessi risultati si possono avere per configurazioni più "paranoiche" che limitano il traffico in uscita (attivandolo solo per risposte a richieste su porte aperte):

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -j ACCEPT
```

Scenario comune: LAN Gateway

Un firewall che agisce come gateway per una rete locale (es: 10.0.0.0/24), che natta su un unico IP pubblico tutti gli IP interni, richiede:

```
iptables -A FORWARD -s 10.0.0.0/255.255.255.0 -i eth0 -j ACCEPT
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -t nat -A POSTROUTING -s 10.0.0.0/255.255.255.0 -j
MASQUERADE
```

IMPORTANTE: Abilitare l'IP forwarding a livello del kernel:

```
sysctl -w net.ipv4.ip_forward = 1
```

Il controllo su quali host della rete interna possono accedere a Internet può essere molto più fine, agendo a livello di porte concesse:

```
iptables -A FORWARD -s 10.0.0.0/255.255.255.0 -p tcp --dport 80 -i
eth0 -j ACCEPT
```

A livello di MAC address degli host abilitati:

```
iptables -A FORWARD -m mac  --mac-source 00:50:8B:4D:55:BB -i eth0
-j ACCEPT
```

Introducendo un limite sul traffico che un host può avere:

```
iptables -A FORWARD -s 10.0.0.120 iptables -m connrate --
connrate :36000 -j ACCEPT
```

Scenario comune: DMZ Firewall

Un firewall di rete classico permette di filtrare il traffico esterno prima che raggiuna dei server in una DMZ. Ha almeno 2 interfacce (una esterna, esposta ad Internet, una sulla DMZ, che costituisce il default gateway dei server pubblici).

Può agire in 2 modi principali:
- Routing fra rete esterna e DMZ con IP pubblic
- Natting fra rete esterna e DMZ con IP privati nattati dal firewall
In entrambi i casi l'IP forwarding va attivato.

Su un routing firewall con DROP di default, per quanto riguarda il controllo del traffico in transito bastano regole come:

```
iptables -A FORWARD -d 217.56.217.2 -i eth0 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -d 217.56.217.2 -i eth0 -p tcp --dport 443 -j ACCEPT
iptables -A FORWARD -m tcp -p tcp -s 217.56.217.0/255.255.255.240 -i eth1 -j ACCEPT
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Se viene introdotto il natting dei server pubblici su host con IP privato (10.0.0.x):

```
iptables -A FORWARD -d 10.0.0.2 -i eth0 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -d 10.0.0.2 -i eth0 -p tcp --dport 443 -j ACCEPT
iptables -A FORWARD -m tcp -p tcp -s 10.0.0.0/255.255.255.0 -i eth1 -j ACCEPT
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -t nat -A PREROUTING -i eth0 -d 217.56.217.2 -j DNAT --to-destination 10.0.0.2
iptables -t nat -A POSTROUTING -s 10.0.0.0/255.255.255.0 -j SNAT
```

Notare che per ogni IP di cui il firewall fa destination natting è necessario che il sistema risponda ad arp request, per cui deve avere tutti gli IP nattati configurati come alias sulla sua interfaccia esterna.

Scenario comune: Transparent Proxy

Un transparent proxy è un normale proxy server che viene usato senza la necessità di configurazione di alcun parametro sul client.
Viene gestito, su Linux, operando su due livelli:
- Iptables per redirezionare il traffico dei client
- Squid (o analogo proxy server) per gestire questa funzionalità.

Configurazione Iptables
E' necessario redirezionare destinato ad host esterni verso la porta su cui ascolta il proxy.
Per esempio, se si vuole redirezionare il traffico web per un proxy locale in ascolto sulla porta 8080, sul gateway inserire:

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 8080
```

Se il proxy fosse su un altra macchina (LAN: 10.0.0.0/24 . Proxy: 10.0.0.15, Iptables FW: 10.0.0.1) si devono impostare regole tipo:

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -s ! 10.0.0.15 -j DNAT --to-destination 10.0.0.15:8080
iptables -t nat -A POSTROUTING -i eth0 -s 10.0.0.0/24 -d 10.0.0.15 -j SNAT --to-source 10.0.0.1
iptables -A FORWARD -s 10.0.0.0/24 -d 10.0.0.15 -i eth0 -o eth0 -p tcp --dport 8080 -j ACCEPT
iptables -A FORWARD -s 10.0.0.15 -i eth0 -o eth1 -j ACCEPT
```

Configurazione Squid
Se si usa Squid assicurarsi che siano impostati in `squid.conf`:

```
http_port 8080
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

Esempio Iptables: Personal Firewall

Tipo Infobox: **SAMPLE** - Skill Level: **3- INTERMEDIATE** - Autore: **Alessandro 'al' Franceschi** - Ultimo Aggiornamento: **2005-05-09 20:20:11**

Vediamo un esempio pratico di configurazione di Iptables per un personal firewall, quindi da applicare su una macchina desktop che deve poter navigare e liberamente agire in Internet e che non deve esporre alcun servizio.
In un caso simile ci si deve concentrare sulle catene INPUT e FORWARD della tabella FILTER.

Configurazione essenziale
Una configurazione minima può essere molto semplice, l'output che segue è quello di `iptables-save`, per utilizzarlo copiarlo in un file (es: /etc/firewall) e applicarlo con `iptables-restore < /etc/firewall`).
In pratica:
- Di default si blocca il traffico in ingresso e destinato ad attraversa l'host
- Si lascia aperto il traffico in uscita dal proprio host
- Si permette in INPUT tutto il traffico di ritorno relativo a connessioni già esistenti
- Si permette il traffico sulla interfaccia di loopback, necessario per il buon funzionamento di alcuni programma di sistema:

```
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i lo -j ACCEPT
COMMIT
```

Configurazione più elaborata
Un esempio più elaborato e con un controllo più stringente sul traffico può prevedere:
- Default DROP su tutte le catene
- Si permette in INPUT tutto il traffico di ritorno relativo a connessioni già esistenti
- Si permette il traffico sulla interfaccia di loopback (si interviene sia in INPUT che in OUTPUT)
- Si permette in uscita solo il traffico generato da applicativi usati dall'utente con UserID 501 (attenzione, questo limite potrebbe compromettere il funzionamento di alcun programmi non previsti, che magari girano come root (UID 0) o con altri utenti di sistema)
- Si permette l'aggiornamento del sistema (in questo esempio via http all'indirizzo arbitrario 213.215.144.242)
- Si loggano tutti i pacchetti, esclusi quelli di broadcast, droppati.

```
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m pkttype --pkt-type ! broadcast -j LOG --log-prefix "INPUT DROP: "
-A FORWARD -m pkttype --pkt-type ! broadcast -j LOG --log-prefix "FORWARD DROP: "
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -m owner --uid-owner 501 -j ACCEPT
-A OUTPUT -p tcp --dport 80 -d 213.215.144.242 -j ACCEPT
-A OUTPUT -m pkttype --pkt-type ! broadcast -j LOG --log-prefix "OUTPUT DROP: "
COMMIT
```

Esempio Iptables: Server pubblico

Tipo Infobox: **SAMPLE** - Skill Level: **3- INTERMEDIATE** - Autore: **Alessandro 'al' Franceschi** - Ultimo Aggiornamento: **2005-05-14 14:26:53**

Analizziamo la configurazione di iptables per un server pubblico, che deve rendere accessibili dei servizi, con diversi livello di accesso, e che ha un firewall locale.
In questo caso si agisce sulle catene di INPUT e di OUTPUT della tabella FILTER.

Configurazione essenziale
Vediamo un esempio essenziale di una configurazione per un server web pubblico, che deve rendere accessibili a tutto il mondo le porte 80 e 443 e deve poter essere gestito da alcuni IP privilegiati.
La configurazione prevede:
- DROP di default su tutte le catene
- Si lascia aperto il traffico in uscita dal proprio host (OUTPUT) dopo aver sanitificato i pacchetti in uscita (--state)
- Si permette in INPUT tutto il traffico di ritorno relativo a connessioni già esistenti
- Si permette il traffico sulla interfaccia di loopback
- Si permette accesso, da tutta Internet, alle porte 80 e 443
- Si permette l'accesso via SSH (porta 22/TCP) da una arbitraria rete di amministrazione (es: 213.25.10.0/24)
- Si permette l'accesso FTP (porta 21/TCP + gestione del canale dati) da un arbitrario host di un webmaster (es: 143.20.12.7)

```
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp --dport 80 -j ACCEPT
-A INPUT -p tcp --dport 443 -j ACCEPT
-A INPUT -p tcp --dport 22 -s 213.25.10.0/24 -j ACCEPT
-A INPUT -p tcp --dport 21 -s 143.20.12.7 -j ACCEPT
-A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
```

```
COMMIT
```

Notare che per gestire correttamente il protocollo FTP è opportuno assicurarsi che venga caricato il modulo del kernel **ip_conntrack_ftp**. Se non viene fatto automaticamente dalla propria distribuzione, caricarlo, facendo eseguire durante la fase di boot (per esempio nel file `/etc/rcd.d/rc.local`) il seguente comando:

```
modprobe ip_conntrack_ftp
```

Configurazione più elaborata

Un esempio più complesso di un simile sistema può prevedere:
- Default DROP su tutte le catene
- Si permette in INPUT tutto il traffico di ritorno relativo a connessioni già esistenti
- Si permette il traffico sulla interfaccia di loopback (si interviene sia in INPUT che in OUTPUT)
- Si permette in uscita solo il traffico di risposta al traffico in entrata permesso
- Si permette l'aggiornamento del sistema (in questo esempio via http all'indirizzo arbitrario 213.215.144.242)
- Si permette l'invio della posta tramite un unico smtp relay (in questo esempio con IP 213.15.44.22)
- Si loggano tutti i pacchetti di tipo unicast (escludendo quindi broadcast e multicast) con una syslog facility che ci permetta il logging su file separato

```
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp --dport 80 -j ACCEPT
-A INPUT -p tcp --dport 443 -j ACCEPT
-A INPUT -p tcp --dport 22 -s 213.25.10.0/24 -j ACCEPT
-A INPUT -p tcp --dport 21 -s 143.20.12.7 -j ACCEPT
-A INPUT -m pkttype --pkt-type unicast -j LOG --log-prefix "INPUT DROP: " --log-level 7
-A FORWARD -m pkttype --pkt-type unicast  -j LOG --log-prefix "FORWARD DROP: " --log-level 7
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -p tcp --sport 80 -j ACCEPT
-A OUTPUT -p tcp --sport 443 -j ACCEPT
-A OUTPUT -p tcp --sport 22 -d 213.25.10.0/24 -j ACCEPT
-A OUTPUT -p tcp --sport 21 -d 143.20.12.7 -j ACCEPT
-A OUTPUT -p tcp --dport 80 -d 213.215.144.242 -j ACCEPT
-A OUTPUT -p tcp --dport 25 -d 213.15.44.42 -j ACCEPT
-A OUTPUT -m pkttype --pkt-type unicast -j LOG --log-prefix "OUTPUT DROP: " --log-level 7
COMMIT
```

Per visualizzare il log su file separato è necessario configurare Syslog. Editare `/etc/syslog.conf` (o analogo) e aggiungere una riga tipo:

```
kern.debug   /var/log/firewall.log
```

.

Esempi di configurazioni particolari

Vediamo alcuni esempi che usano funzioni di iptables meno comuni e non sempre incluse nel kernel di una distribuzione.

- Aprire diverse porte con **multiport**

Se il nostro server deve esporre più porte può essere comodo il match multiport, che permette di definire più porte TCP o UDP in una singola regola.
Ad esempio per un server di posta completo di webmail, pop3, imap e corrispettivi SSL si può permette pubblico accesso con:

```
iptables -I INPUT -m multiport -p tcp --dport 25,80,110,143,443,993,995 -j ACCEPT
```

- Limitare rischi con tool di analisi log con **owner**

Spesso sul sistema vengono installati strumenti di analisi dei log, di qualsiasi tipo. Questi possono essere semplici programmi singoli o applicazioni più complesse, possono venire schedulati o eseguiti in background. Qualsiasi sia la loro natura, sono programmi che ricevono un input con il log da analizzare, che spesso contiene informazioni su attività generate dall'esterno. Log di server web, di firewall, di posta, per esempio, contengono dati che possono entro certi limiti essere manipolati direttamente da un potenziale intrusore, facendo richieste via HTTP anomale, inviando pacchetti invalidi ecc.
Spesso si sorvola sui potenziali rischi associati al parsing di log, soprattutto se questo viene fatto da programmi semplici e non particolarmente prudenti nel gestire i loro input e se magari vengono pure eseguiti con privilegi di root.
Si possono limitare gli effetti di potenziali vulnerabilità di questi strumenti anche via iptables, impedendo all'utente con cui sono eseguiti sul sistema di comunicare via rete.
Primariamente è opportuno che simili programmi vengano eseguiti con permessi non privilegiati, per cui se non già previsto , è opportuno fare in modo che tutti i log e dati che devono leggere e scrivere siano accessibili dall'utente con cui il programma gira.
Poi, si può usare il metodo di match "owner" con cui le iptables possono gestire il traffico a seconda dell'utente che lo ha generato (per logica questo match si applica solo alla catena di OUTPUT).
Ad esempio se il nostro programma di analisi dei log si chiama fwanalog e viene eseguito con utente logs, con UID 150, possiamo impedire che questo programma comunichi all'esterno (escludendo query DNS che potrebbe fare per reverse DNS lookup) con un comando tipo:

```
iptables -I OUTPUT -p udp --dport ! 53 -m owner --uid-owner 150 -j DROP
```

oppure, nei kernel dove è supportata questa estensione:

```
iptables -I OUTPUT -p udp --dport ! 53 -m owner --cmd-owner fwanalog -j DROP
```

Esempio Iptables: Lan Gateway / Firewall

Tipo Infobox: **SAMPLE** - Skill Level: **3- INTERMEDIATE** - Autore: **Alessandro 'al' Franceschi** - Ultimo Aggiornamento: **2005-05-23 14:41:16**

Analizziamo quali possono essere le configurazioni di iptables per un firewall Linux adibito a default gateway di una rete locale in grado di nattare gli host interni e impedire accessi non autorizzati dall'esterno.

Configurazione essenziale

```
*filter
```

```
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -s 10.0.0.0/255.255.255.0 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A FORWARD -s 10.0.0.0/255.255.255.0 -i eth0 -j ACCEPT
-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
COMMIT
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A POSTROUTING -s 10.0.0.0/255.255.255.0 -j MASQUERADE
COMMIT
```

SOURCE: sito web - http://www.safer-networking.org/it/download/index.html

Catene Custom

E' possibile creare proprie catene con un set di regole autonomo.
La nuova regola può quindi diventare il target di qualsiasi altra regola.

Esempio (tipico) per catena custom che droppa, dopo aver loggato, i pacchetti

```
iptables -N LOGDROP
iptables -A LOGDROP -j LOG
iptables -A LOGDROP -j DROP
```

Per usarla:

```
iptables -A INPUT -p tcp --dport 22 -s ! 10.0.0.1 -j LOGDROP
```

Catene custom possono rendere più modulare e gestibile la configurazione di iptables.
Posso rivelarsi particolarmente utili in firewall complessi o particolarmente elaborati.
Su sistemi con logiche essenziali sono meno comode.

Logging dei pacchetti

E' possibile loggare i pacchetti tramite il target **LOG** (log tramite syslog con facility kern) o **ULOG** (log tramite programmi in userspace).
Entrambi i target, a differenza di molti altri, NON interrompono il preseguimento di una catena.
Per pianificare le logiche di logging è opportuno:
- Decidere cosa si vuole loggare (di solito i pacchetti droppati)
- Definire come sono impostare le regole sul firewall
- Applicare le regole di log al posto giusto.

Se si vuole loggare solo i pacchetti droppati inserire una regola di LOG appena prima di ogni analoga regola di DROP.
Se si ha un DROP di default e si prevedono solo regole di ACCEPT, mettere la regola di LOG alla fine di una catena.

Nel target **`LOG`** è possibile definire:
- il livello di criticità per syslog (**`--log-level level`**);
- un prefisso, fino a 32 caratteri, per distinguere una riga di log (**`--log-prefix prefix`**);
- i dettagli da includere (**`--log-tcp-sequence`** , **`--log-tcp-options`** , **`--log-ip-options`**)

Nel target **`ULOG`** è possibile definire:
- il gruppo netlink (1-32) a cui il kernel manda in multicast i pacchetti (**`--ulog-nlgroup nlgroup`**). Default = 1.
- un prefisso, fino a 32 caratteri, per distinguere una riga di log (**`--ulog-prefix prefix`**);
- numeri di byte da copiare in userspace (Default = 0, tutti i byte del pacchetto) (**`--ulog-cprange size`**)
- numero di pacchetti da mettere in code nel kernel prima di inviarli al netlink. Default = 1 (**`--ulog-qthreshold size`**)

Criteri avanzati di match

Oltre alla possibilità di matchare pacchetti sulla base di logiche comuni, esistono delle estensioni particolarmente potenti e versatili.
Vanno richiamate con **`-m match`** e possono avere delle opzioni specifiche per il modulo di match utilizzato.
ADDRESS TYPE (**`-m addrtype`**) - Match sul tipo di pacchetto sulla base del suo indirizzo (es: UNSPEC, UNICAST, MULTICAST, BROADCAST, ANYCAST, LOCAL, BLACKHOLE...)
Es: **`iptables -I INPUT -m addrtype --src-type UNSPEC --dst-type UNICAST -j LOG`**
CONNECTION RATE (**`-m connrate`**) - Match sulla velocità di connessione in byte/secondo.
Es: **`iptables -I FORWARD -m connrate --connrate :36000 -d 213.215.151.8 -j ACCEPT`**
LIMIT (**`-m limit`**) - Impostazione di limiti sulla base di pacchetti /secondo /minuto /ora /giorno.
Es: **`iptables -I FORWARD -p tcp --syn -m limit --limit 10/second --limit-burst 20/second -d 213.215.151.8 -j ACCEPT`**
MAC ADDRESS (**`-m mac`**) - Match sul mac address di un frame ethernet.
Es: **`iptables -I INPUT -m mac --mac-source 00:50:8B:4D:55:BB -j ACCEPT`**
MULTIPORT (**`-m multiport`**) - Permette di definire più porte (sorgente o destinazaione).
Es: **`iptables -I INPUT -p tcp -m multiport --destination-port 25,80,110,143,443,993,995 -j ACCEPT`**
OWNER (**`-m owner`**) - Match di un pacchetto in uscita secondo onwer, pid o nome del processo che lo ha generato.
Es: **`iptables -I OUTPUT -m onwer --uid-owner 501 -j ACCEPT`**
PACKET TYPE (**`-m pkttype`**) - Match sul tipo di pacchetto a livello di datalink (UNICAST, MULTICAST, BROADCAST)
Es: **`iptables -I INPUT -m pkttype --pkt-type UNICAST -j LOG`**

Moduli sperimentali di iptables

Lo sviluppo di moduli su iptables è costante e ai moduli stabili, integrati nelle distribuzioni comuni, si aggiungono moduli sperimentali di matching, target alternativi o patch per il connection tracking di protocolli particolari.
Lo script **Patch'O'Matic** permette di applicare facilmente le patch sperimentali sui sorgenti del proprio kernel.

Extra MATCH
condition - Match sulla base della presenza di file nel proc filesystem
connlimit - Permette di limitare il numero di connessioni per host
comment - Permette di aggiungere un commento alla regola
geoip - Controllo degli IP secondo la nazione di provenienza
nth - Controllo sulla base di logiche di alternanza configurabili
psd - Post Scan Detector
quota - Controllo delle quote di traffico
random - Gestione di regole sulla base di logiche random
string - Controllo a livello di payload: in pratica layer 7 filtering
time - Controllo sulla base dell'ora o del giorno

Extra TARGET
ACCOUNT - Per fare accounting (ad alta velocità) dei pacchetti matchati
NETMAP - Utilizzabile per nattare 1:1 intere reti
ROUTE - Utile per modificare il routing di determinati pacchetti
TARPIT - Introduce una tarpit, una sorta di buco nero in cui le connessioni tendono a rimanere appese

Strumenti di analisi e monitoring

Analizzare i dati di traffico di iptables può essere utile.
In particolare è comodo verificare tutti i pacchetti droppati dal firewall.
I target LOG e ULOG permettono il logging dei pacchetti.

LOG genera una riga di testo via syslog per ogni pacchetto matchato.
Strumenti che possono analizzare questi log sono:
Iptables Log Analyzer `http://www.gege.org/iptables/`
Usa uno script Perl per processare i log e scriverli su MySQL. L'output è una pagina php con dati in tempo reale.
LogRep `http://logrep.sourceforge.net/`
E' un versatile analizzatore di log di diverso tipo, include il supporto di iptables.
LogWatch `http://www.logwatch.org/`
E' un diffuso analizzatore di log generici. Invia mail con l'analisi e il riassunto dei log. Intepreta anche l'output di iptables.
ULOG tramite ulogd permette una gestione più comoda dei log su database.
Fra i tool che usano ulogd:
Webfwlog Firewall Log Analyzer `http://webfwlog.sourceforge.net`
Fornisce pagine php spartane ma complete e configurabili.
Ulog-php `http://www.inl.fr/article.php3?id_article=7`
Pagina php essenziale ma completa che interroga log salvati su MySQL da ulodg.

Per informazioni sugli strumenti, i metodi e le alternative per la gestione e l'analisi dei log: `http://www.loganalysis.org/`

High Availability Firewall

Ogni soluzione di High Availability esistente su linux per un cluster applicativo si può applicare ad un firewall, che lavora a livello di kernel
Gli scenari sono comuni:
- Cluster Attivo-Passivo (con programmi come **Heartbeat**) o Attivo-Attivo
- Non si sono grossi problemi di filesystem, i dati coinvolti sono pochi e poco variabili: si possono usare sistemi gemelli con meccanismi di sincronizzazione della configurazione del firewall (**brbd**, **rsync**, **ssh**)
- Oltre a cluster, si possono usare soluzioni di load balancing
- Al momento non è supportato uno stateful failover: le connessioni esistenti, in uno switch attivo-passivo, vengono interrotte.

Soluzioni possibili (Alcune)
- **Heartbeat** + rsync|ssh scripts (Cluster HA con sincronizzazione dei file gestita tramite script e fatta via rsync o ssh con chiavi autorizzate)
- **Heartbeat** + **DRBD** (Cluster HA con file system sincronizzato tramite network mirroring con DRBD)
- **Linux Virtual Server (LVS)** Sistema di Load Balancing basato su Linux Virtual Server
- **LVS** + **Ultramonkey** - Sistema LVS con HA su Director gestito con Ultramonkey
- **LVS** + **Keepalived** - Sistema LVS con Keepalived per monitoring e gestione failvover
- **LVS** + **Piranha** - Sistema LVS con HA su Director gestito con Ultramonkey
- **LVS** + **Heartbeat** - Sistema LVS con Heartbeat e sistemi di storage sharing

Links
Linux Virtual Server - **http://www.linuxvirtualserver.org/**
Keepalived - **http://www.keepalived.org/**
Heartbeat - **http://www.linux-ha.org/**

Scenari di Load Balancing

Esistono molti metodi per fare load balancing: alcuni sono semplici e grossolani, altri usano software dedicati per monitoring e failure management.

DNS round robin
E' uno dei metodi più semplici. Il bilanciamento viene fatto assegnando diversi indirizzi allo stesso hostname. Non gestisce guasti e, anche con script di monitoraggio/aggiornamento DNS e TTL bassi non risulta essere una soluzione ideale.

Round robin tramite iptables
Esistono diversi metodi: prevedono la gestione del reindirizzamento (tipicamente tramite PREROUTING) dei pacchetti in entrata sull'IP del firewall / balancer verso diversi server reali.
Per gestire l'alternanza si può usare:
- match **nth**, con alternanza round robin
- match **random**, con alternanza random
Per gestire il controllo dei server e togliere dal pool server guasti si può abbinare il match **condition** per una verifica su file che possono essere gestiti da script di monitoraggio.

Linux Virtual Server
Una soluzione strutturata, più complessa ma più affidabile perchè può essere affiancata da tool di monitoring e gestione.
Richiede: Patch LVS nel kernel (ormai ufficiale) e strumenti di controllo e gestione dei servizi
Il Load Balancer può a sua volta essere ridondato con Keepalived, Ultra Monkey, Piranha, Heartbeat ecc.
LVS inoltre include progetti figli interessanti (e sperimentali):
TCPHA - Supporto di strutture con triangolazione del traffico (Internet->Directory->RealServer->Internet) senza gestire i pacchetti di ritorno dai server tramite il Balancer
KTCPVS (Kernel TCP Virtual Server) implementa load balancing a livello applicativo.

Raccomandazioni per il network tuning del Kernel

Vediamo alcune impostazioni per il kernel che possono rivelarsi interessanti per un firewall.
Possono ovviamente essere applicate via **`/etc/sysctl.conf`** e editando direttamente i file in **`/proc/sys/net`**

`sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1` - Disabilita echo request su ping a brodcast (anti smurf abuse)
`sysctl -w net.ipv4.conf/all/accept_redirects=0` - Impedisce la modifica del routing tramite ICMP redirection.
`sysctl -w net.ipv4.conf/all/send_redirects=0`
`sysctl -w net.ipv4.conf.all.accept_source_route=0` - Disabilita source-routing
`sysctl -w net.ipv4.conf.all.forwarding=0`
`sysctl -w net.ipv4.conf.all.mc_forwarding=0`
`sysctl -w net.ipv4.conf.all.rp_filter=1` - Forza verifiche di congruità sul traffico in ingresso e uscita
`sysctl -w net.ipv4.conf.all.log_martians=1` - Logga e droppa pacchetti "marziani" che non si sa dove routare
`sysctl -w net.ipv4.tcp_syncookies=1` - Abilita i syn-cookie, allocando risorse (TCP buffer) solo dopo aver ricevuto un ACK di un SYN/ACK (Utile per Syn Flood)
`sysctl -w net.ipv4.tcp_max_syn_backlog=1280` - Aumenta la dimensione della coda delle socket allocate per SYN ricevuti
`sysctl -w net.ipv4.tcp_keepalive_time = 1800` - Imposta a 1800 secondi (30 minuti) il tempo di keepalive per una connessione TCP. Al termine di questo tempo una connessione non chiusa ma inattiva, viene chiusa (default 7200)

Design di reti sicure - Scenari e discussioni

Una network disegnato secondo buoni principi di sicurezza dovrebbe:
- Avere firewall che limitano al massimo le porte esposte a Internet
- Avere firewall configurati per bloccare tutto, di default
- Differenziare in diversi network server pubblici, backend, e eventuale LAN interna.
- Separare fisicamente (switch diversi o VLAN) reti IP diverse
- Non avere assolutamente hsot con porte pubbliche a cavallo di due reti
- Limitare al massimo e non lasciare porte pubbliche sui firewall a cavallo di due reti
- Avere sistemi di monitoraggio e IDS isolati e non raggiungibili da Internet
- Avere sistemi di analisi dei pacchetti isolati
- Prevedere Intrusion Prevention Systems packet based e non policy based
- Aprire in modo selettivo e temporalmente limitato le porte di accesso a VPN

Logging con Iptables

Tipo Infobox: **DESCRIPTION** - Skill Level: **3- INTERMEDIATE** - Autore: **Alessandro 'al' Franceschi** - Ultimo Aggiornamento: **2005-05-14 22:11:24**

Iptables mette a disposizione la possibilità di loggare pacchetti per debugging e analisi del traffico.
Di fatto il log generato da iptables può essere utile sia per verificare le funzionalità delle regole inserite sia per essere analizzato da tool per creare statistiche e report.

Il logging viene abilitato tramite target **LOG** che lavora in kernel space e usa syslog per scrivere i log o con **ULOG** per trasmettere a programmi in user space le informazioni di log.
E' buon senso loggare solo i pacchetti che vengono bloccati, in modo da avere informazioni utili per troubleshooting (in fase di configurazione e test del firewall) e sicurezza (per capire i volumi e la qualità del traffico non desiderato).

A differenza di molti altri, i target LOG e ULOG non interrompono l'attraversamento di una catena: i pacchetti che matchano una regola di (U)LOG vengono loggati e continuano ad essere processati venendo gestiti da netfilter secondo le regole successive.

In un firewall, che di default blocca tutto, la regola di logging dovrebbe essere l'ultima prima del DROP di default.

Il target LOG
Il target LOG (**-j LOG**) permette di definire diverse opzioni:
--log-level # - Livello di logging, secondo le logiche di syslog, utile, insieme alla configurazione di syslog.conf, per loggare i dati sui pacchetti su file separato.
--log-prefix stringa - Una stringa, lunga fino a 29 caratteri, che viene messa come prefisso alla riga di log, per renderlo più leggibile e associarlo ad un dato tipo di traffico
--log-tcp-sequence - Logga il TCP sequence naumber. Dato sensibile, se accessibile ad altri utenti locali.
--log-tcp-options - Logga le opzioni presenti nell'intestazione TCP
--log-ip-options - Logga le opzioni presenti nell'intestazione IP (come il precedente può essere utile per strumenti di analisi dei log, altrimenti, se le informazioni aggiuntive non si reputano interessanti, è meglio disattivare)
--log-uid - Logga lo UserID del processo che ha generato il pacchetto (in OUTPUT).

Il target ULOG
Meno diffuso e usato rispetto a LOG, ULOG fornisce maggiore flessibilità ed estende notevolmente il campo di applicazioni possibili.
Redireziona infatti in multicast il pacchetto che ha matchato la regola su una netlink socket a cui possono legarsi uno o più processi in userspace ricevendo i pacchetti senza che un loro eventuale malfunzionamento interrompa la gestione del traffico da parte di netfilter.
I programmi che ricevono i pacchetti li possono quindi trattare per molteplici scopi:
- Logging su database o in formati diversi da quello standard via syslog del target LOG
- Ispezioni dei pacchetti per matching su tipologie di traffico articolate, come le regole di un IDS quale SNORT
- Attivazione di meccanismi di Intrusion Prevention, interrompendo flussi di traffico indesiderati o pericolosi
Le applicazioni possono essere molte e modulari e possono essere sviluppate come componenti esterne al kernel.
Una di queste è **ULogD** che funziona come un servizio e utilizza diversi plugin per gestire i dati in vario modo, tra cui la possibilità di loggare su database MySQL o PostgreSQL
--ulog-nlgroup nlgroup - Definisce il gruppo netlink (1-32) a cui è indirizzato il pacchetto. Di default è 1
--ulog-prefix prefix - Come su LOG, imposta una stringa come prefisso alla riga di log (max 32 caratteri)
--ulog-cprange # - Numero di byte da copiare in user space. Di default è 0 e l'intero pacchetto viene inviato
--ulog-qthreshold # - Numero di pacchetti che il kernel deve avere in coda prima di inviarli al netlink. Default è 1

Esempi
Per loggare i pacchetti sulle catene di filter FORWARD, INPUT e OUTPUT aggiungendo in ogni riga di log adeguato prefisso.
```
iptables -A FORWARD -j LOG --log-prefix="FORWARD: "
iptables -A INPUT -j LOG --log-prefix="INPUT:"
iptables -A OUTPUT -j LOG --log-prefix="OUTPUT:"
```

Per loggare tutti i pacchetti UDP in INPUT con porta sorgente diversa da 137,138 e 139, inserendo come prefisso INPUT UDP:
```
iptables -A INPUT -p UDP --source-port ! 137:139 -j LOG --log-prefix="INPUT UDP:"
```

Per loggare tramite ULOG:
```
iptables -A INPUT -j ULOG --log-prefix="INPUT:"
```
In esecuzione sul sistema deve esserci us servizio come ulogd in grado di intercettare il multicast su netlink.

Un buona regola di log da appendere alla fine di una catena con policy di default DROP, può essere la seguente. Logga solo pacchetti di tipo unicast (comodo per evitare log di drop di multicast o broadcast inoffensivi) e con livello 7 di severity su sysylog (debug):
```
iptables -A INPUT -m pkttype --pkt-type unicast -j LOG --log-prefix "INPUT: " --log-level 7
```

Esempio di output di LOG su syslog
```
Feb 19 11:01:12 GIOVE kernel: FORWARD=eth1 OUT=eth0 SRC=192.168.0.98 DST=213.198.151.2 LEN=66 TOS=0x00 PREC=0x00
TTL=127 ID=165 PROTO=UDP SPT=1047 DPT=53 LEN=46
```

LINK: Netfilter Log Format - http://logi.cc/linux/netfilter-log-format.php3
LINK: Netfilter Log analyzer on the web - http://logi.cc/linux/NetfilterLogAnalyzer.php3
LINK: Iptables Log analyzer - http://www.gege.org/iptables/
LINK: ulogd - http://gnumonks.org/projects/ulogd

Configurazione Kernel Linux per il firewalling

Tipo Infobox: **DESCRIPTION** - Skill Level: **5- SENIOR** - Autore: **Giorgio 'neo' Colombo** - Ultimo Aggiornamento: **2002-10-04 01:20:15**

Per far si che il nostro linux funga da firewall, dovremo compilare il kernel abilitando il netfiltering e alcune sue features a seconda dello scopo e del tipo di traffico che dovra' filtrare.
Come la maggior parte delle features del kernel linux si ha la possibilita' di selezionarle come moduli oppure compilarli all'interno del kernel se si preferisce un kernel monolitico.

Di seguito vengono messe in evidenza le piu' importanti opzioni da considerare nella fase di configurazione del kernel per attivare e gestire il packet filtering:

CONFIG_PACKET
Permette a varie applicazione ed utility di lavorare direttamente sui device di rete
CONFIG_NETFILTER
DA ABILITARE!! Di fatto e' l'opzione base che vi permettera' anche di accedere al menu delle singole opzioni di iptables
CONFIG_IP_NF_CONNTRACK
Modulo che permette il tracking delle connessioni, utilizzato per il NAT e masquerading
CONFIG_IP_NF_FTP
Modulo che permette il tracking delle connessioni FTP, utile se volete fare firewall perimetrale o di contenimento che devono gestire anche traffico ftp. Bene o male FTP si utilizza ovunque, direi DA ABILITARE.
CONFIG_IP_NF_IPTABLES
DA ABILITARE!! Opzione che permette qualsiasi tipo di filtering, NAT e masquerading. Praticamento e' il core delle iptables
CONFIG_IP_NF_MATCH_LIMIT
Opzione non obbligatoria, ma permette di fare filtering sulle connessioni, stabilendo che un certo numero di pacchetti in un certo limite di tempo devono matchare alcune regole. Utile per difendersi da attacchi DoS
CONFIG_IP_NF_MATCH_MAC
Opzione che permette l'utilizzo del MAC address per il matching delle regole. Comodo se si hanno poche macchine da gestire, ma sarete pesantemente dipendenti dall'hardware che utilizzerete
CONFIG_IP_NF_MATCH_MARK
Abilita la possibilita' di fare il matching tramite il MARK dei pacchetti.
CONFIG_IP_NF_MATCH_MULTIPORT
Permette di filtrare le connessioni specificando un range di sorce port e destination port. DA ABILITARE
CONFIG_IP_NF_MATCH_TOS
Permette il filetring tramite TOS (Type of Service). ONLY FOR EXPERT
CONFIG_IP_NF_MATCH_TCPMSS
Abilita' la possibilita del matching del MSS field. ONLY FOR EXPERT
CONFIG_IP_NF_MATCH_STATE
Permette il stateful matching delle connessioni. DA ABILITARE !!
CONFIG_IP_NF_MATCH_UNCLEAN
Opzione che permette di filtrare il traffico TCP,UDP,ICMP che sono considerati invalidi. DA ABILITARE, ma ricordarsi che non funziona del tutto corettamente in tutti i casi
CONFIG_IP_NF_MATCH_OWNER
Ci permette di filtrare le socket, abilitando un'utente specifico per aprire un certo tipo di socket. E' ANCORA IN FASE DI SVILUPPO
CONFIG_IP_NF_FILTER
Modulo base per il packet filetering. DA AGGIUNGERE!!
CONFIG_IP_NF_TARGET_REJECT
Modulo che permette di specificare che i messagi di ICMP error di essere inviati alla sorgente
CONFIG_IP_NF_TARGET_MIRROR
Abilita il bounced back (rimbalzati indietro) dei pacchetti al sender. ES. se si decide di mettere un MIRROR target sulle porte di destinazione di HTTP (80), tutte le richieste verranno replicate al sender e poi seguite dai pacchetti contenenti le informazioni per visualizzare la pagina
CONFIG_IP_NF_NAT
Modulo che abilita il NAT, Port forwarding e masquerading. DA ABILIATRE se si vuole translare gli ip sorgente o di destinazione
CONFIG_IP_NF_TARGET_MASQUERADE
Modulo che permette di aggiungere il TARGET per il masquerading. DA ABILITARE, se si vuole fare masquerading.
CONFIG_IP_NF_TARGET_REDIRECT
Opzione che permette che vi permette di far funzionare il vostro firewall come se fosse un transparent proxy
CONFIG_IP_NF_TARGET_LOG
Permette la possibilita' di loggare i packetti. DA ABILITARE, anche se quello che si logga dipende dalle vostre regole. Se decidete di abilitare questa opzione e di loggare in syslog, utilizzate tools per avere un report del entry registrate nel log sia per rendere la lettura dei log piu' comprensibile sia per la possibilita' di mantenere una history
CONFIG_IP_NF_TARGET_TCPMSS
Questa opzione viene utilizzata per ovviare a vari problemi con host o ISP che bloccano i pacchetti che richiedono l'ICMP Fragmentation.
CONFIG_IP_NF_COMPAT_IPCHAINS
Abilita la compatibilita' con le vecchie IPCHAINS. DA ABILIATRE solo in caso di bisogno
CONFIG_IP_NF_COMPAT_IPFWADM
Abilita la compatibilita' con le vecchie IPCHAINS. DA ABILIATRE solo in caso di bisogno

Esistono altri parametri configurabili in run-time ovvero dopo che il kernel e' stato caricato e il sistema è in funzione.
Si hanno piu' possibilita' per modificare questi parametri, quello piu' corretto e' quello di modificare i vari parametri editando `/etc/sysctl.conf` oppure usare il comando `sysctl` o scrivendo direttamente i valori voluti nel **proc filesystem**.
Questa operazione viene normalmente eseguita tramite script lanciati al boot della macchina.

Segue un elenco delle principali opzioni modificabili in modalita' run-time per quanto riguarda il network, ovviamente nel dubbio è meglio lasciare i valori di default, salvo quelli strettamente necessari per un firewall (ip_forward).
I valori che si possono assumere sono di tipo:
- **BOOLEAN = 0** (disabilita) e 1(abilita)
- **INTEGER = N** , dove N e' un mumero intero Es 64
- **N INTEGERS = X Y Z** , N indica il numero dei valori(interi) da inserire, e X Y Z i valori che acquisisce la variabile

Tutte le variabili fanno riferimento a `/proc/sys/net/ipv4/*`:
ip_forward BOOLEAN - Permette il forwarding dei pacchetti fra le interfacce. DA ABILITARE.
ipfrag_high_thresh INTEGER - La massima memoria utilizzata per riassemblare i pacchetti IP frammentati. Quando questo valore verra'

raggiunto, i pacchetti in surplus verranno scartati finche' non si raggiungera' di nuovo il valore settato in `ip_frag_low_thresh`
ipfrag_time INTEGER - Indica il tempo max in secondi per mantenere in memoria un "IP fragment"
tcp_syn_retries INTEGER - Indica il numero di volte che un SYN deve essere ritrasmesso per i tentativi di una connessione TCP attiva (non puo' essere piu' grande di 255)
tcp_synack_retries INTEGER - Indica il numero delle volte che un SYNACK deve essere ritrasmesso per i tentavi di connessioni TCP passive (non puo' essere piu' grande di 255)
tcp_keepalive_time INTEGER - Indica quanto spesso verra' inviato il messaggio TCP di keepalive. Default 2hdsfsfds
tcp_keepalive_probes INTEGER - Indica quanti pacchetti Keepalive probes verranno inviati, finche' la connessione verra' considerata definitivamente broken
tcp_keepalive_interval INTEGER - Indica quanto frequentemente i pacchetti di probe sono inviati. Moltiplicando per tcp_keepalive_probes si ottiene il tempo per il KILL di una connessione che non risponde ai pacchetti di probes.
tcp_retries1 INTEGER - Indica il numero di tentativi/richieste prima che venga segnalato al network layer che sulla connessione c'e' qualcosa che non va.
tcp_retries2 INTEGER - Indica il numero di tentativi/richieste prima di killare una sessione TCP attiva
tcp_orphan_retries INTEGER - Indica il numero dei tentativi/richieste prima che una connessione TCP venga killata
tcp_fin_timeout INTEGER - Indica il tempo per cui mantenere una socket nello status di FIN-WAIT-2
tcp_max_tw_buckets INTEGER - Indica il numero massimo di socket simultanee in timewait che mantiene il sistema
tcp_max_orphans INTEGER - Numero massimo di socket TCP che possono rimanere non collegati a file handle aperti dal sistema.
tcp_abort_on_overflow BOOLEAN - Permette il reset di una connessione se un servizio in listening e' troppo lento per rispondere.
tcp_sysncookies BOOLEAN - Opzione valida solo se il kernel e' stato compilato con il supporto dei SYNCOOKIES, previene possibili "syn flood attack"
tcp_timestamps BOOLEAN - Abilita o disabilita il timestamp
tcp_ecn BOOLEAN - Abilita o disabilta la notificazione di una possibile congestione della network
ip_local_port_range 2 INTEGERS - Desinisce il range delle porte locali, utilizzate dal protocollo TCP e UDP. Il primo numero indica la porta piu' bassa ed il secondo indica la porta piu' alta. Il valore di default dipende dal quantitativo di ram disponibile sul sistema
ip_dynaddr BOOLEAN - Se settato non a zero abilita il supporto per il dynamic addresses
icmp_echo_ignore_all BOOLEAN - Ignora i Ping (ICMP ECHO Request)
icmp_echo_ignore_broadcasts BOOLEAN - Se settati entrambi il sistema ignora gli ICMP ECHO solo verso indirizzi di broadcast/multicast
log_martians BOOLEAN - Logga i pacchetti con indirizzi impossibili.
accept_redirects BOOLEAN - Accetta o meno gli ICMP redirect
forwarding BOOLEAN - Abilita il forwarding per una specifica interfaccia
proxy_arp BOOLEAN - Abilita il proxy arp.
secure_redirects BOOLEAN - Accetta ICMP redirect message solo dai gw configurati.
La modifica di alcuni parametri potrebbe pregiudicare il buon funzionamento del sistema, ma può essere necessaria per il fine tuning di sistemi sotto particolari carichi di rete.
Per maggiori informazioni e indicazioni utili sui parametri raccomandabili fare riferimento direttamente al file `Documentation/networking/ip-sysctl.txt` nel tarball del kernel.

LINK: IPTables log analyzer - http://www.gege.org/iptables/
LINK: GUI per Iptables - http://www.fwbuilder.org/
HOWTO: Linux Bridge+Firewall Mini-HOWTO version 1.2.0 - http://ldp.openskills.info/HOWTO/Bridge+Firewall.html

Iptables Patch'O'Matic

Tipo Infobox: **DESCRIPTION** - Skill Level: **5- SENIOR** - Autore: **Alessandro 'al' Franceschi** - Ultimo Aggiornamento: **2005-06-01 10:21:27**

Il team di sviluppatori di Netfilter e i loro collaboratori hanno realizzato numerose patch che allargano in modo considerevole le funzionalità del codice base, normalmente incluso nel kernel.
Molte di queste non sono incluse nel codice ufficiale del kernel in quanto considerate sperimentali, non completamente stabili o non necessarie, alcune lo saranno in futuro o sono incluse nel ramo 2.6 ma non nel 2.4, alcune sono incluse nei kernel forniti da specifiche distribuzioni.
Esiste comunque la possibilità di inserirle in un proprio kernel custom utilizzando **Patch'O'Matic** (POM), un comodo strumento realizzato dagli autori di Netfilter.

INTRODUZIONE

Per applicare le patch aggiuntive sul codice del proprio kernel sono necessari:
- I sorgenti del kernel (normalmente scompattati nella directory `/usr/src/linux` o `/usr/src/kernel-`***`versione`*** o simili) su cui verranno applicate le patch
- I sorgenti di iptables, scaricabili dal sito ufficiale http://www.netfilter.org/ che vanno ricompilati per supportare le nuove opzioni introdotte
- I sorgenti di patch'o'matic, anch'essi scaricabili dal sito ufficiale. Notare che dai primi di Gennaio 2005 il team di Netfilter ha smesso di rilasciare release ufficiali di patch'o'matic, per cui è interessato a queste patch si deve basare sulle release "current" rilasciate regolarmente.

Le patch (sono numerose e variegate) sono divise in diversi gruppi:
submitted - Patch proposte e sottoposte per l'inclusione nel kernel ufficiale
pending - Patch che sono in attesa di essere proposte per essere incluse nel kernel ufficiale
base - Patch di base, comunque sperimentali e non testate in modo approfondito
extra - Tutte le patch disponibili, numerose, variegate e sperimentali
obsolete - Codice obsoleto o deprecato.
Notare che esistono patch in conflitto fra loro, che non vanno compilate sullo stesso kernel. Generalmente i conflitti, le dipendenze e le incompatibilità sono indicate per ogni patch.

INSTALLAZIONE

Si scaricano e scompattano i sorgenti. Non essendoci più release ufficiali di POM, in questo esempio si è ricorso all'ultimo snapshot, con tutti i rischi del caso nell'usare software in testing.

```
wget http://ftp.netfilter.org/pub/patch-o-matic-ng/snapshot/patch-o-matic-ng-20050413.tar.bz2
tar -jxvf patch-o-matic-ng-20050413.tar.bz2
cd patch-o-matic-ng-20050413
```

A questo punto si può procedere con la procedura standard di POM:
`export KERNEL_DIR=/usr/src/kerneldir` Indicare la directory dove si trovano i sorgenti del kernel da patchare
`export IPTABLES_DIR=/usr/src/iptablesdir` Dove sono stati scompattati i sorgenti di iptables
`./runme pending`
Alternativamente eseguire `./runme base` o `./runme extra` per includere i diversi gruppi.
Per ogni patch viene data una breve spiegazione e chiesto se si vuole inserirla nel kernel.
Una volta completata questa operazione sarà possibile procedere alla compilazione del kernel secondo le solite procedure. Nel menu di configurazione del kernel tutte le patch aggiuntive sono include nella voce **NetFilter Configuration** del menu **Networking options**.
Oltre ai moduli del kernel, patch'O'matic permette di patchare il mondo userland del comando iptables, in modo da gestire le nuove funzionalità.

PATCH (+) INTERESSANTI
Vediamo alcune delle patch più interessanti che POM mette a disposizione, queste riguardano soprattutto nuovi criteri di matching dei pacchetti e nuovi target:

match: comment
Questa patch, già inclusa nel kernel ufficiale 2.6, permette di aggiungere un commento ad una regola, introducento il modulo di match **comment** con l'unica opzione **--comment**.
Esempio: `-A INPUT -s 213.215.144.242 -m comment --comment "Host Webmaster"`
Voce nel kernel .config: CONFIG_IP_NF_MATCH_COMMENT
Nome modulo: `ipt_comment`

target: NETMAP
Ancora sperimentale, presente nel repository base, introduce il target **NETMAP** che permette di nattare 1 a 1 tutti gli host di intere reti senza specificare i singoli IP. Comodo per firewall che nattano tutti i propri server in una DMZ con IP privati o per fare double natting per far comunicare reti con gli IP privati sovrapposti.
Esempio: `iptables -t nat -A PREROUTING -d 213.215.144.128/25 -j NETMAP --to 10.0.0.128/25`
Si può usare anche per source natting: `iptables -t nat -A POSTROUTING -s 213.215.144.128/25 -j NETMAP --to 10.0.0.128/25`
Voce nel kernel .config: CONFIG_IP_NF_TARGET_NETMAP
Nome modulo: `ipt_NETMAP`

match: connlimit
Patch sperimentale che imposta un numero massimo di connessioni tcp da un singolo arbitrario host IP o da una intera rete. Introduce il match **connlimit** con le opzioni **--connlimit-above** (determina il numero massimo di connessioni) e **--connlimit-mask** (imposta il limite allargandolo ad una intera rete, definendola tramite i subnet bit, ad esempio 24. Di default si limita al singolo host: 32)
Esempi:
Per limitare l'accesso SSH a due sole connessioni per IP sorgente:
`-I INPUT -p tcp --dport 22 --syn -m connlimit --connlimit-above 2 -j DROP`
Per impostare massimo 16 connessioni per rete di 256 indirizzi ad un server web dietro il firewall Linux:
`-A FORWARD -d 213.215.144.242 -p tcp -m tcp --dport 80 --syn -m connlimit --connlimit-above 16 --connlimit-mask 32 -j DROP`
Per impostare a 250 il numero massimo di connessioni contemporanee che possono essere fatte ad un server SMTP:
`-A FORWARD -d 213.215.144.242 -p tcp -m tcp --dport 25 --syn -m connlimit --connlimit-above 250 --connlimit-mask 0 -j DROP`
Voce nel kernel .config: CONFIG_IP_NF_MATCH_CONNLIMIT
Nome modulo: `ipt_connlimit`

match: iprange
Comoda patch che permette di fare il match su un range di IP sorgenti o destinazione. Introduce il modulo di match **iprange** con le opzioni: **--src-range** (range di IP sorgenti, es: 10.0.0.0-10.0.255.255) e **--dst-range**.
Esempio: `iptables -A FORWARD -m iprange --dst-range 10.0.0.0-10.0.255.255 -j ACCEPT`
Voce nel kernel .config: CONFIG_IP_NF_MATCH_IPRANGE
Nome modulo: `ipt_iprange`

match: nth
Curiosa e versatile patch che permette di fare un match ogni n pacchetti sulla base di opzioni varie, mettendo a disposizione di default 16 "contatori" indipendenti. Introduce il match **nth** con le opzioni:
--every Imposta ogni quanti pacchetti eseguire il match
--counter Definisce il contatore da utilizzare (di default 0)
--start Definisce a quale pacchetto iniziare a contare (di default 0)
--packet Definisce quale pacchetto all'interno del contatore matchare. Se viene definito va inserira una regola per ogni --packet da 0 a N-1
Esempio senza l'uso di --packet per loggare un pacchetto ogni 10, in entrata: `iptables -I INPUT -m nth --every 10 -j LOG`
Esempio per un semplice ma efficace load balancer dove tutti i pacchetti alla porta 80 dell'IP pubblico vengono forwardati in modalità round robin a 3 diversi server web nattati (viene usato il contatore 1):

```
iptables -t nat -A PREROUTING -d 213.215.144.242 -p tcp --dport 80 -m nth --counter 1 --every 3 --packet 0 -j DNAT --to-dest 10.0.0.10:80
iptables -t nat -A PREROUTING -d 213.215.144.242 -p tcp --dport 80 -m nth --counter 1 --every 3 --packet 1 -j DNAT --to-dest 10.0.0.11:80
iptables -t nat -A PREROUTING -d 213.215.144.242 -p tcp --dport 80 -m nth --counter 1 --every 3 --packet 2 -j DNAT --to-dest 10.0.0.12:80
```

Voce nel kernel .config: CONFIG_IP_NF_MATCH_NTH
Nome modulo: `ipt_nth`

match: psd
E' possibile identificare dei Port Scan tramite il match **psd** con le seguenti opzioni:
--psd-weight-threshold Peso degli ultimi pacchetti TCP/UDP a porte di destinazione diverse proveniente dallo stesso host.
--psd-delay-threshold Ritardo, in centesimi di secondi, per considerare pacchetti a porte diverse come dovuti ad un port scan
--psd-lo-ports-weight Peso dei pacchetti destinate a porte privilegiate (<=1024).

--psd-hi-ports-weight Peso dei pacchetti destinati a porte non privilegiate.
Un esempio essenziale per aggiungere un log, che può essere verboso, di tutti gli scan (Consigliabile aggiungerlo alla fine della catena di INPUT, con un default DROP finale):
iptables -A INPUT -m psd -j LOG --log-prefix "PORTSCAN: "
Voce nel kernel .config: CONFIG_IP_NF_MATCH_PSD
Nome modulo: **ipt_psd**

match: quota
Implementa il match **quota** con cui si definisce un contatore relativo al traffico di rete che si vuole permettere (o comunque matchare).
Unica opzione: **--quota** (numero di byte massimi).
Esempio: **iptables -A FORWARD -m quota --quota 1024000 -d 10.0.0.5 -j ACCEPT**
Voce nel kernel .config: CONFIG_IP_NF_MATCH_QUOTA
Nome modulo: **ipt_quota**

match: random
Un curioso match **random** che introduce la possibiltà di definire regole con possibilità di match random in base all'opzione: **--average** che indica la percentuale media di match (di default il 50%)
Esempio: **iptables -A INPUT -m random --average 10 -d 10.0.0.5 -j LOG**
Voce nel kernel .config: CONFIG_IP_NF_MATCH_RANDOM
Nome modulo: **ipt_random**

match: time
Permete di creare regole sulla base dell'ora, della data o del giorno della settimana. Introduce il match **time** con le seguenti opzioni:
--timestart value - Match se l'ora locale è successiva a quella indicata (HH:MM, default 00:00).
--timestop value - Match se l'ora è prima di quella indicata (default 23:59).
--days - Giorno della settimana (formato: Mon,Tue,Wed,Thu,Fri,Sat,Sun ; default tutti i giorni)
--datestart - Match se la data è successiva a quella indicatata (Formato: YYYY[:MM[:DD[:hh[:mm[:ss]]]]])
--datestop - Match se la data locale è prima di quella indicata (Formato YYYY[:MM[:DD[:hh[:mm[:ss]]]]])
Esempio per permettere l'accesso solo in orari lavorativi:
iptables -A INPUT -m time --timestart 8:00 --timestop 18:00 --days Mon,Tue,Wed,Thu,Fri -j ACCEPT
Esempio per permettere l'accesso solo fino al 31 Dicembre 2005:
iptables -A INPUT -m time --datestop 2005:12:31:23:59 -j ACCEPT
Voce nel kernel .config: CONFIG_IP_NF_MATCH_TIME
Nome modulo: **ipt_time**

match: condition
Fornisce un meccanismo di matching che può rivelarsi estremamente flessibile per modificare le regole di firewalling secondo il contenuto di un file.
In pratica permette di verificare se un file qualsiasi nella directory /proc/net/ipt_condition/ ha valore 0 o 1.
Esempio per aprire la porta 80 se il file /proc/net/ipt_condition/openweb ha contenuto "1":
iptables -A INPUT -m condition --condition openweb -p tcp --dport 80 -j ACCEPT
Voce nel kernel .config: CONFIG_IP_NF_MATCH_CONDITION
Nome modulo: **ipt_condition**

match: geoip
Rivoluzionario modulo per eseguire filtering sulla base della nazione da cui proviene un IP. Si appoggia ad un database GeoIP, ha bisogno dei file **/var/geoip/geoipdb.bin** e **/var/geoip/geoipdb.idx** su quali eseguire il check sulle nazioni a cui sono associati i vari IP.
I database si possono creare con l'utility csv2bin o scaricare da http://people.netfilter.org/peejix/geoip/database/.
Voce nel kernel .config: CONFIG_IP_NF_MATCH_GEOIP
Nome modulo: **ipt_geoip**

match: string
Permette il match di una stringa nel contenuto di un pacchetto. Di fatto, seppur non è questo l'utilizzo migliore perchè non prevede la conoscenza della logica del protocollo, permette filtering a livello applicativo.
Esempio classico per loggare CodeRed e affini (notare che non esclude falsi positivi): **iptables -I INPUT -p tcp -s 0.0.0.0/0 -m string --string "cmd.exe" -j LOG**
Voce nel kernel .config: CONFIG_IP_NF_MATCH_STRING
Nome modulo: **ipt_string**

SOURCE: Home page path'o'matic sul sito di Netfilter - http://www.netfilter.org/patch-o-matic/
LINK: SecurityFocus: IPTables Linux firewall with packet string-matching support - http://securityfocus.com/infocus/1531