ZEUS News è un notiziario dedicato a quanto avviene nel mondo di Internet, dell'informatica, delle nuove tecnologie e della telefonia fissa e mobile: non è un semplice amplificatore di comunicati stampa ma riserva ampio spazio ai commenti e alle riflessioni, proponendosi quale punto di osservazione libero e indipendente.

Linux in Rete, stop agli intrusi

Ogni computer collegato a Internet è un potenziale bersaglio di attacchi informatici che, se portati a segno, possono avere conseguenze devastanti non solo dal punto di vista tecnico. Proteggere il Pinguino si può, e con notevole efficacia: vediamo come fare.

L'interconnessione tra elaboratori ha assunto, ormai da tempo, importanza fondamentale, tanto da attribuire alle reti il ruolo di infrastruttura critica in ogni sistema informativo. Di conseguenza anche Internet, rete delle reti, è oggi considerata parte integrante del supporto ad ogni attività informatica, comprese quelle, più o meno ludiche, effettuate nelle nostre case di utenti finali.

Se ne comprende facilmente il motivo: la connessione a Internet consente di accedere, con costi estremamente contenuti, a una mole incalcolabile di informazioni e risorse di ogni genere e offre la possibilità di comunicare in tempo reale con persone o sistemi situati praticamente ovunque nel mondo. Tali attrattive, unitamente alla grande facilità di accesso ai servizi reperibili in Rete, portano spesso a dimenticare un fatto importantissimo: se è vero che dal nostro computer possiamo raggiungere tutto il mondo, è altrettanto vera la situazione inversa. In altre parole, da ogni computer connesso a Internet è possibile raggiungere il nostro; per di più, non è assolutamente certo che, qualora ciò effettivamente avvenga, vi siano all'origine intenzioni oneste e amichevoli.

Il problema era meno grave alcuni anni fa, quando la quasi totalità degli accessi casalinghi a Internet avveniva mediante modem relativamente poco performanti e aveva la durata appena indispensabile per navigare qualche sito e scaricare la posta o leggere le *news*. Proprio la durata limitata della connessione e, soprattutto, il fatto che l'indirizzo IP attribuito al computer dell'utente cambiasse ad ogni collegamento, rendevano gli utenti finali un bersaglio poco appetibile per i malintenzionati.

Ma, a causa del rapido evolvere della tecnologia e dell'incremento esponenziale dell'offerta di servizi in Rete, la situazione è ora radicalmente diversa: la durata dei collegamenti aumenta (*chat*, *download*, servizi *streaming* audio o video) e si diffondono tecnologie, quali l'ADSL, che consentono connessioni casalinghe permanenti con indirizzo IP fisso. Una vera manna per i *crackers*, che si vedono enormemente facilitati nel loro obiettivo di individuare computer vulnerabili contro i quali sferrare attacchi, spesso destinati a riuscire con estrema facilità.

Una volta "bucato" il computer, è poi un gioco da ragazzi curiosare tra i segreti della vittima alla ricerca di informazioni interessanti (il numero della carta di credito, la password del servizio di *internet banking*...) o, addirittura, creare una testa di ponte dalla quale attaccare altri sistemi, facendo ricadere ogni resposabilità sull'ignaro utilizzatore. Non si tratta affatto di uno scenario

fantascientifico: molto spesso, infatti, i nostri computer sono configurati come preimpostato dal programma di installazione del sistema e, di conseguenza, può avvenire che molti servizi (telnet, web, ftp, dns, http proxy e altri ancora) siano attivi senza che il proprietario ne sia consapevole. Quando la macchina è collegata alla Rete, basta una password banale (ammettiamolo: lo sono quasi tutte, quasi sempre) o la mancata installazione dell'ultima patch di sicurezza e... la frittata è fatta.

Le conseguenze possono essere davvero gravi, e non solo sotto il profilo tecnico: la legge, infatti, stabilisce responsabilità penali per l'amministratore negligente che si sia reso complice involontario di malversazioni informatiche. In altre parole, si può rischiare la galera. Bisogna considerare, però, che la legge non stabilisce obblighi precisi: questi devono essere, al contrario, commisurati alle caratteristiche del sistema informativo e all'ambito in cui esso viene utilizzato. Un *Internet provider* è tenuto a provvedere alla sicurezza dei propri sistemi con impegno, attenzione e risorse certamente maggiori di quanto richiesto a chi utilizzi un computer in casa propria, per lavoro o per hobby, e con esso si connetta in modo più o meno saltuario alla Rete.

Perciò, la soluzione non sta nel rinunciare a Internet, confinandosi spontanemante e per sempre in un mondo di esclusi dalle sue meraviglie... la prudenza è d'obbligo, ma la paranoia è comunque un eccesso.

Quando si parla di protezione contro le intrusioni, *firewall* è quasi una parola d'ordine. Il firewall (muro tagliafuoco) è un sistema in grado di isolare tra loro due o più reti e consentire che tra le stesse transiti esclusivamente il traffico desiderato. Va detto, a scanso di equivoci, che un firewall non è necessariamente *la* soluzione unica, definitiva e insostituibile al problema delle intrusioni; inoltre la sua complessità e quella della sua configurazione dipendono largamente dall'ambito in cui esso è impiegato: si possono avere firewall software e sistemi costituiti da più componenti hardware e software cooperanti.

Il caso particolare di cui ci stiamo occupando è quello di un computer utilizzato per normali attività di *home computing*, connesso a Internet in modo semipermanente via modem analogico, ISDN o ADSL. Esso, inoltre, potrebbe essere collegato in rete locale a un secondo computer (ad esempio un notebook). Infine, particolare molto importante, si tratta di un computer su cui "gira" GNU/Linux, il quale implementa, già a livello di kernel, la capacità di filtrare il traffico in transito sulle interfacce di rete, consentendo esclusivamente il passaggio del traffico conforme alle regole stabilite dall'amministratore. Per le premesse testè presentate, possiamo affermare di avere a disposizone un vero e proprio firewall software, in grado di proteggere degnamente il nostro Pinguino con un livello di sicurezza assolutamente adeguato a evitare che ogni sessione di lavoro in Rete sia fonte di infinite preoccupazioni. A patto, naturalmente, che se ne comprenda il funzionamento, lo si configuri con attenzione e, sopratutto, non ci si illuda di avere ottenuto l'assoluta invulnerabilità.

L'implementazione di riferimento, detta Netfilter, è una componente del kernel stabile a partire dalla versione 2.4; la capacità di effettuare *packet filtering* è presente nei kernel di Linux già a partire dalla versione 1.1, ma proprio con il kernel 2.4 essa, grazie ad una notevole opera di reingegnerizzazione, ha raggiunto livelli di efficienza, versatilità e efficacia senza precedenti.

Per configurare efficacemente un firewall, è necessario comprenderne, oltre alle principali particolarità sintattiche, la logica di implementazione. Ma, prima ancora, è opportuno approfondire, seppure con parecchie semplificazioni, alcuni aspetti tecnici del protocollo IP, cioè delle regole che in Internet presiedono alla comunicazione tra computer.

Uno sguardo al protocollo IP

Tutte le comunicazioni che si svolgono in Internet si basano sul protocollo IP (Internet Protocol), il quale prevede che ogni computer abbia un indirizzo, unico al mondo, composto di quattro cifre separate da un punto. Ciascuna di esse è espressa mediante 8 bit ed è compresa, di conseguenza, tra 0 e 255. Perciò 193.207.153.121 è un indirizzo IP valido e, in tutta Internet, non può essere attribuito a più di un computer. In realtà, ogni computer può essere dotato di più interfacce di rete, ciascuna delle quali, a sua volta, può avere più di un indirizzo; tuttavia ciascuno di essi, per essere utilizzabile in Internet, deve essere unico.

Ogni indirizzo IP può essere suddiviso in due parti: la prima indica la rete (in pratica, una sottorete di Internet), mentre la seconda identifica il computer nell'ambito di quella rete. E' consuetudine indicare quanti bit dell'indirizzo IP esprimono la rete posponendone il numero all'indirizzo stesso: se l'indirizzo dell'esempio precedente fosse scritto 193.207.153.121/24, si intenderebbe che i primi 24 bit (le prime tre cifre) indicano la sottorete. Tale indirizzo identifica il computer 121 appartenente alla sottorete 193.207.153. Azzerando tutti i bit che esprimono il computer si ottiene l'indirizzo di base della rete (193.207.153.0); ponendoli tutti a 1 si ottiene l'indirizzo di *broadcast* della rete, cioè l'indirizzo che identifica tutti i computer ad essa appartenenti (193.207.153.255).

Alcuni *range* di indirizzi, per convenzione, vengono riservati alle reti private e non sono utilizzabili in Internet. Essi corrispondono alle notazioni 10.0.0.0/8 (in pratica, tutti gli indirizzi la cui prima cifra è 10, altrimenti detti indirizzi di *classe A*), 172.16.0.0/12 (*classe B*; tutti gli indirizzi da 172.16.0.0 a 172.31.255.255), 192.168.0.0/16 (*classe C*; da 192.168.0.0 a 192.168.255.255). A detti gruppi di indirizzi va assimilato il range 127.0.0.0/8 (*loopback*; tutti gli indirizzi la cui prima cifra è 127), riservati alle comunicazioni "confinate" all'interno di un singolo computer: per convenzione, l'indirizzo "interno", o di *loopback*, di un computer è 127.0.0.1. Vi sono poi indirizzi, riservati ad usi particolari, validi unicamente quali destinazione: 224.0.0.0/4 (*classe D*; da 224.0.0.0 a 239.255.255.255); infine, gli indirizzi 240.0.0.0/5 (*classe E*; da 240.0.0.0 a 247.255.255.255) sono riservati per usi futuri.

Il protocollo IP garantisce che per trasmettere informazioni ad un computer è sufficiente conoscerne l'indirizzo IP: l'infrastruttura di rete provvede ad individuare il percorso necessario per raggiungerlo e a trasportare l'informazione. Ma, in Internet, si è soliti utilizzare *nomi* di comupter, piuttosto che i loro indirizzi: *www.zeusnews.it* non è un indirizzo IP, bensì un nome, al quale deve per forza corrispondere almeno un indirizzo IP unico. In Internet è disponibile un servizio, detto DNS (Domain Name Service) deputato proprio alla "risoluzione" dei nomi in indirizzi: qualcosa di analogo a un elenco telefonico, che permette di risalire al numero di telefono a partire dal nome dell'abbonato. Il DNS è utilizzato (solitamente in modo automatico) dai programmi ai quali non si fornisca l'indirizzo IP del computer da contattare, bensì il suo nome.

Per ragioni di efficienza, l'informazione viene spedita suddivisa in piccole porzioni, dette *pacchetti*. Ogni pacchetto dispone di una intestazione (*header*) nella quale sono indicati, tra l'altro, l'indirizzo IP del mittente e quello del destinatario, i quali possono costituire un primo criterio per distinguere i pacchetti leciti da quelli indesiderati. Inoltre, per motivi legati alla tecnologia di implementazione delle infrastrutture di rete, i pacchetti possono essere frammentati lungo il percorso, cioè suddivisi in porzioni di minori dimensioni, per essere poi ricostruiti una volta giunti a destinazione. Dal punto di vista della sicurezza, la frammentazione rappresenta una criticità: infatti, solo il primo frammento è dotato di uno *header* recante tutte le informazioni IP; i successivi frammenti potrebbero facilmente essere "costruiti" o modificati ad arte con intenzioni malevole. Quindi può essere

opportuno che il firewall accetti solo pacchetti integri.

Ma non è tutto. Quando un pacchetto raggiunge un computer, come è possibile sapere a quale programma è destinato, tra tutti quelli eseguiti in quel momento? Se, per esempio, desideriamo visualizzare la home page di Zeus News e ne digitamo l'URL, il nostro browser risale all'indirizzo IP mediante il DNS e spedisce proprio a quel computer i pacchetti contenenti la richiesta. Ma sul computer *www.zeusnews.it* potrebbe non essere attivo soltanto il web server: anzi, è verosimile che altri programmi stiano comunicando in rete o siano in attesa di essere contattati. Ecco allora entrare in gioco altri protocolli di rete, i quali, appoggiandosi all'IP per l'indirizzamento del traffico, presiedono al trasporto delle informazioni da applicazione ad applicazione. I più noti sono il TCP (Transfer Control Protocol) e UDP (User Datagram Protocol). Anche il protocollo di trasporto rappresenta un criterio efficace per discriminare il traffico tra desiderato e indesiderato, tuttavia è possibile operare filtraggi ancora più specifici prendendo in considerazione il concetto di "servizio" o, più tecnicamente, di *porta*: si tratta dello strumento mediante il quale i protocolli TCP e UDP associano pacchetti IP ad una specifica applicazione.

Ogni programma in attesa di informazioni dalla rete si pone in ascolto su una *porta* TCP o UDP che, su quel computer, sarà di suo esclusivo utilizzo: nessun altro programma potrà usare la stessa porta per comunicare. Ogni porta è identificata da un numero compreso tra 0 e 65535; quelle numerate tra 0 e 1023 possono essere utilizzate solo da programmi privilegiati; molte di esse, inoltre, sono convenzionalmente dedicate a utilizzi ben definiti. Ad esempio, ai server HTTP (web) è solitamente riservata la porta TCP 80, ai server SMTP (invio delle email) la porta TCP 25, ai server POP3 (scaricamento delle email dalla casella) la porta TCP 110, ai DNS server la porta UDP 53 (ma i server DNS utilizzano anche la porta 53 su porotocollo TCP per alcune operazioni).

Per tornare al nostro esempio, il web server (il termine "server" indica, qui, il processo che gestisce il servizio HTTP) di Zeus News è in ascolto sulla porta 80: l'indirizzo completo TCP/IP a cui inviare la nostra richiesta, tale da permettere ai pacchetti di raggiungere non solo la macchina, ma il processo in grado di interpretarli correttamente, è perciò 193.207.153.121:80. La comunicazione tra computer è, di per sé, bidirezionale: nel caso dell'esempio, i pacchetti trasportano le richieste dal browser al server, e le pagine web da questo al browser (nulla vieta che, in altri casi, non sia esclusivamente uno dei due attori a inoltrare richieste e l'altro a rispondere). Il web server, perciò usa la porta 80 anche per rispondere e il browser, a sua volta, deve utilizzare una porta che, per quanto sopra, avrà numero compreso tra 1024 e 65535, assegnato dal sistema operativo. Nell'ipotesi che al nostro computer sia stato attribuito dall'Internet provider l'indirizzo IP 150.139.63.212, la comunicazione tra web server e browser (e viceversa) sarà stabilita tra gli indirizzi TCP/IP 193.207.153.121:80 e 150.139.63.212:26752, ove 26752 è un esempio del numero di porta che può essere assegnato al browser dal sistema operativo del nostro computer.

Risulta ora evidente che un ulteriore criterio utile per filtrare il traffico IP è rappresentato dai numeri di porta, cioè dai servizi oggetto della comunicazione.

Una delle differenze principali tra TCP e UDP consiste nell'affidabilità del trasporto: il protocollo UDP non offre alcuna garanzia in tal senso, privilegiando l'efficienza della comunicazione; al contrario, il TCP gestisce una numerazione sequenziale dei pacchetti scambiati e prevede la negoziazione della connessione, al fine di stabilire una vera e propria sessione di lavoro tra i due computer. La negoziazione TCP è, tutto sommato, un processo piuttosto semplice: la macchina che intende aprire un nuovo collegamento invia al computer destinatario un pacchetto che, nel proprio header, presenta valorizzato a 1 un particolare flag, chiamato SYN. Il destinatario risponde con un pacchetto che, sempre nello header, presenta valorizzato a 1, oltre al SYN flag, anche il flag

chiamato ACK. Per concludere la negoziazione, il primo computer risponde con un pacchetto il cui solo flag ACK è 1. Come dire: "Ci sei? - Si, ci sono. - OK, allora cominciamo a dialogare."

La presenza del solo flag SYN nel pacchetto di apertura del collegamento permette di distinguere, in un flusso di traffico TCP, la richiesta di servizio dalla prosecuzione del colloquio: ciò permette al firewall, ad esempio, di bloccare le richieste in arrivo verso il nostro computer e consentire, al tempo stesso, il transito delle risposte alle richieste di servizio da noi effettuate.

Vi sono poi altri protocolli notevoli, basati sull'indirizzamento IP: uno di questi è l'ICMP (Internet Control Message Protocol), utilizzato per segnalazioni di controllo relative al funzionamento della connessione (è usato dal famoso comando *ping*, ad esempio). Come ogni altro protocollo, può essere individuato e filtrato dal firewall: analizzeremo alcune sue peculiarità affrontando la configurazione vera e propria di Netfilter.

A questo punto, senza pretesa di avere esaurito la materia del networking IP, siamo pronti per tornare a Linux e occuparci dell'implementazione di Netfilter o, almeno, delle sue componenti utili all'obiettivo: proteggere il nostro computer casalingo da intrusioni provenienti da Internet.

La logica del firewall Netfilter

La logica di Netfilter prende le sue mosse dalle cosiddette "tavole" (*tables*; non a caso il comando che consente di configurare e amministrare il firewall è *iptables*): ciascuna di esse raggruppa uno o più insiemi di regole applicabili in un ambito elaborativo definito. Ad esempio, la *filter table*, sulla quale concentreremo la nostra attenzione, entra in azione quando è richiesto il packet filtering. Vi sono poi altre tavole, come la *nat* e la *mangle*, sulle quali non ci soffermeremo.

Gli insiemi di regole raccolti nelle *tables* sono detti "catene" (*chains*); alcune hanno nomi e obiettivi predefiniti, ma l'amministratore ne può creare di nuove, secondo necessità. Della *filter table* fanno parte, per default, tre catene, inizialmente vuote: la *INPUT*, le cui regole sono applicate ai pacchetti originati all'esterno del computer che esegue il firewall e ad esso destinati; la *OUTPUT*, contenente le regole per filtrare i pacchetti che, generati dal computer, sono diretti al suo esterno; infine la *FORWARD*, per i pacchetti che, avendo indirizzi di origine e destinazione entrambi differenti da quelli locali, si affidano alle capacità di routing del kernel per essere trasferiti tra due delle reti alle quali il computer è collegato (nel caso in esame le reti sono soltanto due: una è quella costituita dal computer stesso e da un'eventuale LAN; l'altra, naturalmente, è rappresentata da tutta Internet).

Ogni regola inserita nelle *chain* deve indicare i criteri che individuano il pacchetto su cui agire (indirizzo, protocollo, flags, eccetera) e l'azione, o *target*, da applicare allo stesso. Netfilter implementa tre azioni: *ACCEPT*, usata per lasciare passare il pacchetto; *REJECT*, per scartarlo dandone avviso al mittente; *DROP*, per scartarlo senza avvisare (solitamente preferibile ad ACCEPT, in quanto è opportuno non sobbarcarsi lavoro inutile e potenzialmente pericoloso). A dette regole si aggiungono la possibilità di effettuare il *logging* delle caratteristiche del pacchetto (*LOG*) e quella di trasferire il controllo a un'altra *chain*, solitamente definita dall'amministratore.

Per ogni pacchetto in arrivo, in partenza o in transito, le regole della catena applicabile sono esaminate in sequenza. Se le caratteristiche del pacchetto corrispondono a quelle specificate in una delle regole della *chain*, è eseguita l'azione che essa specifica e si passa ad analizzare il pacchetto successivo, altrimenti viene esaminata la regola che segue. Se nessuna delle regole risulta

applicabile, è eseguita l'azione predefinita per quella catena: l'azione di default per tutte le catene è ACCEPT, ma si tratta, ovviamente, di una impostazione modificabile. Portare il default a DROP, cioè vietare tutto ciò che non è esplicitamente consentito, è saggia prudenza, ma può avere alcune controindicazioni: vi sono infatti applicazioni importanti (tra le quali, ad esempio, il sistema grafico XFree) che si appoggiano al trasporto di rete anche per la comunicazione tra componenti locali e potrebbero bloccarsi se questa venisse bloccata anche per breve tempo.

Il rimedio può consistere nel modificare il default dopo avere specificato le regole necessarie ad abilitare il traffico interno (ma sarebbe necessario procedere con estrema attenzione a ulteriori modifiche della configurazione), oppure, come nel caso che presenteremo, nell'accettare la *policy* di default e concludere ogni catena con una regola che scarti tutti i pacchetti non gestiti da quelle precedenti.

La configurazione anti-intrusi

Entriamo ora nel vivo della fase di configurazione. Vedremo come definire le regole indispensabili per proteggersi con un livello di sicurezza ragionevole, commentando, al tempo stesso, la sintassi di *iptables*; si noti che tutti i comandi descritti di seguono devono essere dati dall'utente *root*.

Per prima cosa è bene fare un po' di pulizia: con i due comandi seguenti si forza Netfilter a "scaricare" (*flush*) tutte le regole eventualmente attive e, rispettivamente, a eliminare tutte le chain definite dall'amministratore in una certa *table*.

```
iptables -F
iptables -X
```

Si noti che, come del resto nei comandi che seguiranno, non è stata indicata la *table* su cui operare: Netfilter assume perciò che i comandi agiscano sulla tavola *filter*. Un default perfettamente appropriato, dal momento che è proprio nostra intenzione effettuare *packet filtering*.

Ora possiamo definire una nuova *chain*, che chiameremo "STOP": ci consentirà di semplificare la notazione necessaria a richiedere il *logging* dei pacchetti scartati.

```
iptables -N STOP
```

Come si vede, per creare una nuova catena è sufficiente invocare iptables con il comando "-N" e specificarne il nome. Si tratta ora di popolarla con le regole necessarie: vogliamo che la catena STOP effettui il logging di ogni pacchetto esaminato e lo scarti. Per aggiungere una nuova regola ad una catena si deve utilizzare il comando "-A" di iptables, seguito dal nome della chain, dai criteri di *matching* del pacchetto e dal *target*, cioè dall'azione che vogliamo sia eseguita quando il pacchetto esaminato corrisponde ai criteri definiti.

```
iptables -A STOP -j LOG
iptables -A STOP -j DROP
```

E' facile intuire, dall'esame dei due comandi, che l'opzione "-j" (*jump*) specifica l'azione da eseguire. Dal momento che i comandi non specificano alcuna regola esplicita, l'azione indicata con "-j" sarà eseguita per tutti i pacchetti. Nel primo caso, la parola chiave LOG (di cui abbiamo detto sopra) richiede che le caratteristiche del pacchetto siano scritte nel log di sistema, visualizzabile in

qualsiasi momento con il comando seguente:

```
dmesg
```

Il target *LOG* ha un comportamento leggermente particolare: invece di interrompere l'elaborazione della catena e passare all'analisi del pacchetto successivo, come avviene con tutti gli altri target, Netfilter continua ad elaborare la catena. E' perciò eseguita l'azione richiesta dal comando successivo, DROP, che scarta silenziosamente il pacchetto. Scopo della semplice chain appena definita è, evidentemente, quello di scartare i pacchetti dopo averne loggato l'arrivo al firewall.

E' il momento di definire le regole per le catene INPUT, OUTPUT e FORWARD. Per quest'ultima, in particolare, è sufficiente una sola regola:

```
iptables -A FORWARD -j STOP
```

Dal momento che il nostro computer non deve agire come router tra la rete locale di cui fa parte e Internet (e viceversa), tutti i pacchetti dei quali il computer medesimo non risulti mittente o destinatario devono essere scartati. Il target STOP, senza alcun criterio di *matching*, assicura che tutti i pacchetti analizzati dalla catena FORWARD verranno sottoposti ad esame ulteriore secondo le regole della catena STOP, in base alla quale, come ormai sappiamo, essi saranno loggati e scartati. Bloccare i pacchetti in transito non significa impedire l'accesso a Internet ad altri pc connessi alla rete locale, almeno per quel che riguarda la navigazione web: e' sufficiente installare sul computer firewall un *proxy server*; ottimo, allo scopo, è *Squid* (*http://www.squid-cache.org*), incluso in tutte le più diffuse distribuzioni GNU/Linux. Va sottolineato che il routing non sarebbe di per sé sufficiente a consentire ai client locali di presentarsi su Internet senza un intermediario, in quanto, essendo configurati con indirizzi IP necessariamente privati, non sarebbero comunque raggiunti dalle risposte dei server remoti. L'intermediario in questione è, ancora una volta, Netfilter: sarebbe necessario attivarne le funzionalità di *masquerading* (gestite dalla *nat table*), per attribuire a tutti i pacchetti in uscita l'indirizzo del pc connesso direttamente alla Rete quale indirizzo di origine e tenere traccia delle connessioni IP aperte dai computer locali, per recapitare le risposte ai rispettivi destinatari.

Per quel che riguarda INPUT e OUTPUT, è bene loggare e bloccare subito tutto il traffico "sospetto", servendosi anche in questo caso della catena STOP come target:

```
iptables -A INPUT -j STOP -m state --state INVALID
iptables -A OUTPUT -j STOP -m state --state INVALID
iptables -A INPUT -j STOP -f
iptables -A OUTPUT -j STOP -f
```

Il primo e il secondo comando bloccano tutti i pacchetti il cui stato sia INVALID: si tratta di traffico che difficilmente potrebbe risultare di qualche utilità. Si noti che INVALID non è uno stato proprio del protocollo IP, ma è attribuito da Netfilter ai pacchetti che non possono essere ricondotti a un flusso di traffico noto e, comunque, non possono essere identificati. Degli altri stati gestiti da Netfilter ci occuperemo più avanti; qui vale la pena di evidenziare la sintassi della regola che impone il test sullo stato. La notazione "`-m state --state INVALID`" significa che Netfilter deve attivare il proprio modulo ("-m") dedicato all'analisi di stato ("state") e che lo stato da individuare ("--state") è "INVALID". La regola è definita sia per la catena di INPUT che per quella di OUTPUT, con due comandi distinti.

Più semplice è la notazione relativa a terzo e quarto comando, volti a filtrare tutti i frammenti ("-f")

in entrata (catena INPUT) e in uscita (OUTPUT).

Dopo avere eretto le prime, fondamentali, barriere, non dimentichiamo di abilitare il traffico interno al computer: si tratta dei pacchetti scambiati tra processi locali. L'interfaccia (virtuale) di rete coinvolta, o *loopback*, è conosciuta dal sistema col nome *lo*.

```
iptables -A INPUT  -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

Nei comandi presentati la novità sintattica è rappresentata dall'indicazione esplicita dell'interfaccia di rete: le regole da essi stabilite saranno perciò valide esclusivamente per quell'interfaccia. L'opzione "-i" è utilizzata per specificare l'interfaccia di ingresso dei pacchetti (e compare, pertanto, nelle regole inserite nella catena INPUT); "-o" specifica l'interfaccia di uscita (nelle regole della catena OUTPUT).

Occupiamoci, ora, della LAN. Le schede di rete Ethernet sono normalmente identificate mediante il nome *eth*, seguito da un suffisso numerico. La prima scheda è, perciò, *eth0*, la seconda *eth1*, e così via. Dal momento che stiamo configurando il firewall su una macchina dedicata ad attività di *home computing*, connessa a Internet via modem, non è necessario che in essa sia installata una scheda di rete locale. Tuttavia, non è fuori luogo l'ipotesi che, per il collegamento a un secondo pc o ad una piccola LAN, una scheda ethernet sia effettivamente presente: vale la pena di presentare le semplici regole volte a garantire un livello minimale di sicurezza anche verso "l'interno".

Possiamo assumere che la LAN non sia, di per sé, del tutto insicura: in effetti, l'unico punto di contatto con l'esterno è rappresentato proprio dal pc sul quale stiamo configurando Netfilter. Perciò ci limitiamo a verificare, senza ulteriori limitazioni, che tutto il traffico da e verso la LAN abbia indirizzi di origine e destinazione validi. Nell'ipotesi che l'indirizzamento IP della rete locale sia gestito in classe C (192.168.0.0/24) e che il computer di cui ci occupiamo abbia indirizzo 192.168.0.1, si avrà:

```
iptables -A INPUT -i etho -j STOP -s 192.168.0.1
iptables -A INPUT -i eth0 -j STOP -d ! 192.168.0.1
iptables -A INPUT -i eth0 -j STOP -s ! 192.168.0.0/24
iptables -A INPUT -i eth0 -j ACCEPT
iptables -A OUTPUT -o eth0 -j STOP -d 192.168.0.1
iptables -A OUTPUT -o eth0 -j STOP -s ! 192.168.0.1
iptables -A OUTPUT -o eth0 -j STOP -d ! 192.168.0.0/24
iptables -A OUTPUT -o eth0 -j ACCEPT
```

E' ovvio che ciascuno dovrà utilizzare, in luogo di quelli proposti, gli indirizzi reali della propria configurazione. La condizione "-d" specifica un indirizzo (o range di indirizzi) di destinazione, mentre "-s" indica un indirizzo, o un range, di provenienza (*source*). Infine, il punto esclamativo stabilisce la negazione logica della condizione che lo segue, invertendone il significato (lo si può leggere: “non”).

La prima delle otto regole significa perciò: "In ingresso sull'interfaccia della rete locale passa alla catena STOP i pacchetti che *provengono* dall'indirizzo 192.169.0.1 (cioè quello di questo computer)". Si tratta, evidentemente, di un indirizzo falsificato. La seconda, invece, richiede di bloccare il traffico il cui indirizzo di *destinazione* non è 192.168.0.1. La terza trasferisce alla catena STOP i pacchetti la cui provenienza non è il range 192.168.0.0/24, cioè la rete locale. La quinta regola blocca il traffico che esce dal computer e sembra essere destinato allo stesso; sesta e settimaa regola bloccano, sempre in uscita verso la LAN, i pacchetti che non sembrano provenire dal nostro

computer o non sono diretti a indirizzi della rete locale medesima.

Vale la pena di osservare che la *policy* di tutte le catene è "ACCEPT" per default: pertanto viene accettato tutto ciò che non è esplicitamente vietato. Tuttavia la necessità della quarta e ottava regola è spiegata dalle regole molto restrittive che inseriremo in chiusura della configurazione del firewall, tramite le quali, per prudenza, bloccheremo tutto il traffico che non sarà stato già esplicitamente scartato o ammesso.

Dalle regole impostate per la LAN si comprende che per porre più condizioni in modalità "OR" (cioè per fare in modo che sia eseguito lo stesso target quando risulti vera almeno una di esse) è necessario specificarne una alla volta in regole generate da comandi sucessivi. Vedremo in seguito che più condizioni specificate con un solo comando costituiscono una regola sola, nella quale esse sono valutate in modo "AND": il target è eseguito se risultano tutte vere contemporaneamente.

L'insieme delle regole dedicate alla connessione verso Internet è invece più complesso: il mondo esterno deve essere considerato un ambiente potenzialmente ostile, dal quale proteggerci con un robusto ed efficace muro tagliafuoco. Naturalmente, nel muro sarà necessario aprire qualche "breccia": il minimo indispensabile a garantire il regolare passaggio del solo traffico desiderato. Dal momento che il nostro computer non è collegato alla Rete in modo permanente, possiamo ipotizzare che non vi esponga servizi: si assume, in altre parole, che i processi server eventualmente attivi (web, proxy, dbms o quant'altro) siano utilizzati localmente o, al più, da parte delle macchine collegate alla LAN. Dovremo perciò preoccuparci di rifiutare qualsiasi tentativo di connessione proveniente da Internet (salvo una sola eccezione, di cui diremo); sarà prudente, inoltre, bloccare tutti i pacchetti che potrebbero essere stati manipolati ad arte per tentare di "scavalcare" il muro posto a protezione della nostra rete.

Qualche attenzione dovrà essere rivolta anche al traffico in uscita: nella malaugurata ipotesi che un *cracker* riesca ad attivare sul nostro computer un *trojan*, cioè un programma in grado di agire come un "cavallo di Troia", limiteremo così le sue possibilità di compiere in Internet malefatte delle quali potrebbe esserci attribuita la responsabilità. Naturalmente, salvo il caso, disastroso, in cui l'attaccante sia tanto abile da acquisire i privilegi dell'utente *root*, necessari per modificare la configurazione del firewall.

Cominciamo col bloccare tutti i pacchetti i cui indirizzi di origine o destinazione sono sospetti:

```
iptables -A INPUT -i ppp0 -j STOP -s 10.0.0.0/8
iptables -A INPUT -i ppp0 -j STOP -d 10.0.0.0/8
iptables -A INPUT -i ppp0 -j STOP -s 172.16.0.0/12
iptables -A INPUT -i ppp0 -j STOP -d 172.16.0.0/12
iptables -A INPUT -i ppp0 -j STOP -s 192.168.0.0/16
iptables -A INPUT -i ppp0 -j STOP -d 192.168.0.0/16
iptables -A INPUT -i ppp0 -j STOP -s 240.0.0.0/5
iptables -A INPUT -i ppp0 -j STOP -d 240.0.0.0/5
iptables -A INPUT -i ppp0 -j STOP -s 127.0.0.0/8
iptables -A INPUT -i ppp0 -j STOP -d 127.0.0.0/8
iptables -A INPUT -i ppp0 -j STOP -s 0.0.0.0
iptables -A INPUT -i ppp0 -j STOP -d 0.0.0.0
iptables -A INPUT -i ppp0 -j STOP -s 255.255.255.255
iptables -A INPUT -i ppp0 -j STOP -d 255.255.255.255
```

Non ci sono novità sintattiche. Vale però la pena di evidenziare che la connessione a Internet non passa attraverso una vera scheda di rete, bensì è gestita mediante una interfaccia virtuale, il cui nome è *ppp0*. Anche in questo caso, il suffisso numerico è correlato al numero delle connessioni

attive (ciascuna passante per un modem dedicato); ovviamente si ipotizza che al nostro computer ne sia collegato uno soltanto.

In ingresso blocchiamo il traffico proveniente o destinato a tutti gli indirizzi di classe A, B, C ed E, in quanto, trattandosi di range privati o riservati, è bene diffidare: come hanno fatto a giungere fino a noi? Prudenza anche nel caso di pacchetti "targati" *loopback*: essendo per definizione interni ad ogni computer, cosa ci fanno in giro per Internet? Scartiamo anche i pacchetti indirizzati 0.0.0.0, perché si tratta di un indirizzo di rete riservato, e quelli indirizzati 255.255.255.255 perché rappresentano una sorta di *broadcast* globale, destinato a raggiungere (o proveniente da) tutti i computer connessi a Internet: piuttosto strano.

Prudenza anche con gli indirizzi di classe D (*multicast*): essi sono ammessi solo quale *destinazione* dei pacchetti, perciò bloccheremo quelli che sembrano provenire da indirizzi appartenenti a quel range.

```
iptables -A INPUT -i ppp0 -j STOP -s 224.0.0.0/4
```

Il traffico con destinazione multicast è accettato, ma solo se il protocollo di trasporto è UDP e se la loro frequenza non è eccessiva:

```
iptables -A INPUT -i ppp0 -j ACCEPT -d 224.0.0.0/4 -p udp -m limit
                                        --limit 5/second --limit-burst 10
```

La condizione "-p" permette di specificare il protocollo di trasporto (in questo caso UDP, come anticipato), mentre con "-m", questa volta, chiediamo a Netfilter di utilizzare il modulo "limit" per tenere traccia del numero di pacchetti in transito in una data unità di tempo. Dovranno essere accettati, in media ("--limit"), non più di 5 pacchetti al secondo ("5/second"), con picchi massimi ("--limit-burst") di 10: i pacchetti in numero eccedente detti limiti saranno scartati. Come anticipato, perché un pacchetto sia accettato, dovranno essere contemporaneamente vere le condizioni relative all'indirizzo di destinazione, al protocollo e alla frequenza. E' interessante rilevare che, solitamente, il traffico multicast è locale alla rete di origine. Ne segue che l'eventuale traffico multicast ricevuto su *ppp0* è stato probabilmente generato dal nostro Internet provider. Se abbiamo motivo di ritenere che ciò non sia possibile, o desiderabile, potrà essere opportuno bloccarlo sostituendo la regola testè presentata con la seguente:

```
iptables -A INPUT -i ppp0 -j STOP -d 224.0.0.0/4
```

Le regole volte a bloccare il traffico in ingresso sulla base dell'indirizzo di destinazione possono essere considerate ridondanti, a condizione di scartare tutti i pacchetti che appaiono diretti a indirizzi diversi da quello assegnato all'interfaccia *ppp0* dal nostro Internet provider: ciò appare interessante soprattutto per coloro che dispongono di un indirizzo IP internet fisso. La loro configurazione, infatti, può essere semplificata impostando una sola regola in sostituzione di tutte quelle che, nel gruppo precedente, filtrano il traffico sulla base dell'indirizzo di destinazione:

```
iptables -A INPUT -i ppp0 -j STOP -d ! <indirizzo_fisso_di_ppp0>
```

Inutile precisare che *<indirizzo_fisso_di_ppp0>* deve essere sostituito con l'indirizzo IP fisso assegnato dal Provider per il collegamento alla Rete. Se si desidera ammettere il traffico multicast, tale regola bloccante deve essere inserita dopo la "ACCEPT" ad esso relativa; in caso contrario, essa può essere definita in testa al gruppo. Le regole "-d" possono essere eliminate.

Chi dispone di una connessione con indirizzo dinamico è costretto ad inserire la regola relativa al proprio indirizzo solo dopo avere effettivamente attivato la connessione: l'operazione richiede un po' di *shell scripting*, perciò la presentazione di un caso concreto andrebbe oltre le finalità introduttive del presente articolo. Ma... niente paura: non si tratta di una regola irrinunciabile. In chiusura commenteremo, comunque, alcuni comandi riconosciuti da *iptables* per la modifica "al volo" della configurazione di Netfilter.

Del tutto analoghe le regole per la catena OUTPUT, con una sola differenza notevole: dal momento che, per assunzione, la nostra macchina non espone servizi, il traffico multicast in uscita è comunque bloccato, anche se l'indirizzamento è lecito.

```
iptables -A OUTPUT -o ppp0 -j STOP -s 10.0.0.0/8
iptables -A OUTPUT -o ppp0 -j STOP -d 10.0.0.0/8
iptables -A OUTPUT -o ppp0 -j STOP -s 172.16.0.0/12
iptables -A OUTPUT -o ppp0 -j STOP -d 172.16.0.0/12
iptables -A OUTPUT -o ppp0 -j STOP -s 192.168.0.0/16
iptables -A OUTPUT -o ppp0 -j STOP -d 192.168.0.0/16
iptables -A OUTPUT -o ppp0 -j STOP -s 240.0.0.0/5
iptables -A OUTPUT -o ppp0 -j STOP -d 240.0.0.0/5
iptables -A OUTPUT -o ppp0 -j STOP -s 127.0.0.0/8
iptables -A OUTPUT -o ppp0 -j STOP -d 127.0.0.0/8
iptables -A OUTPUT -o ppp0 -j STOP -s 0.0.0.0
iptables -A OUTPUT -o ppp0 -j STOP -d 0.0.0.0
iptables -A OUTPUT -o ppp0 -j STOP -s 255.255.255.255
iptables -A OUTPUT -o ppp0 -j STOP -d 255.255.255.255
iptables -A OUTPUT -o ppp0 -j STOP -s 224.0.0.0/4
iptables -A OUTPUT -o ppp0 -j STOP -d 224.0.0.0/4
```

Anche in questo ambito, chi si connette a Internet con indirizzo IP statico può eliminare tutte le regole "-s", inserendo in testa al gruppo la seguente:

```
iptables -A OUTPUT -o ppp0 -j STOP -s <indirizzo_fisso_di_ppp0>
```

Ancora una volta, disponendo di un indirizzo dinamico, è necessario (se proprio si desidera una configurazione estremamente curata) ricorrere allo *scripting*.

Servono ora regole atte a consentirci di accedere ai servizi Internet di nostro interesse: uno di essi è certamente quello di risoluzione dei nomi (DNS). La definizione di una regola su misura per uno o più servizi deve essere basata sulle caratteristiche peculiari dei servizi stessi: innanzitutto va considerato che la nostra esigenza è unicamente interrogare i server DNS. A tale fine i server "ascoltano" la porta 53 e utilizzano il protocollo di trasporto UDP; i client (come il nostro computer) aprono il colloquio su una propria porta non privilegiata, secondo la norma. Abbiamo tutte le informazioni che ci servono per definire la regola appropriata:

```
iptables -A OUTPUT -o ppp0 -j ACCEPT -p udp --sport 1024:65535
                              --dport 53 -m state --state NEW,ESTABLISHED
```

Il significato della regola, che va digitata su una riga unica ed è presentata su due righe solamente per ragioni tipografiche, è: "In uscita attraverso l'interfaccia *ppp0* accetta tutti i pacchetti su protocollo ("-p") UDP che hanno origine da una porta ("--sport", *source port*) numerata da 1024 a 65535 (1024:65535) e diretti alla porta ("--dport", *destination port*) 53, purché un controllo sul loro stato ("-m state") evidenzi che si tratta ("--state") del pacchetto di apertura di una connessione ("NEW") o (",") facente parte di una connessione già in corso ("ESTABLISHED")". I pacchetti "NEW" sono quelli tramite i quali il nostro computer inizia la connessione col DNS server, mentre

quelli "ESTABLISHED" sono da noi inviati in risposta ad altri da esso ricevuti. Pertanto, in uscita non solo è consentito colloquiare, ma anche *prendere l'inziativa* di richiedere il servizio.

La regola per la catena INPUT è più restrittiva: sono infatti ammessi solo i pacchetti aventi stato "ESTABLISHED".

```
iptables -A INPUT -i ppp0 -j ACCEPT -p udp --dport 1024:65535 --sport 53
                                          -m state --state ESTABLISHED
```

Ciò significa che ogni tentativo di richiedere al nostro computer il servizio DNS sarà bloccato e potranno passare solo i pacchetti inviati dalla porta 53 di un server impegnato in un colloquio già iniziato da noi. Si noti che, in ingresso, porte di destinazione e di origine sono invertite rispetto alla regola definita in output: questa volta, infatti il traffico va inteso diretto dal server (porta 53) al nostro computer (porta non privilegiata).

Abbiamo aperto una prima breccia nel muro: tuttavia, i suoi contorni sono precisi e limitati quanto basta per non temere di avere offerto un varco significativamente utile ad un ipotetico attaccante.

Di quali altri servizi Internet vogliamo fruire? Certamente siamo interessati a HTTP (navigazione web), FTP (trasferimento di files), SMTP (invio di email) e POP3 (ricezione di email), ma anche a HTTP/S (navigazione su canale crittografato per le connessioni, ad esempio, con siti di eCommerce). Sono tutti basati su trasporto TCP e i server sono in ascolto, rispettivamente, sulle porte 80, 21, 25, 110 e 443. Possiamo aprirci la strada con una sola coppia di regole:

```
iptables -A OUTPUT -o ppp0 -j ACCEPT -p tcp --sport 1024:65535 -m
     multiport --dports 21,25,80,110,443 -m state --state NEW,ESTABLISHED
iptables -A INPUT -i ppp0 -j ACCEPT -p tcp --dport 1024:65535 -m
         multiport --sports 21,25,80,110,443 -m state --state ESTABLISHED
```

L'analogia con il servizio DNS, ad eccezione del protocollo, è evidente: la particolarità consiste, piuttosto, nell'avere utilizzato il modulo ("-m") *multiport* per specificare in una regola sola tutte le porte di destinazione ("--dports") desiderate. Anche in questo caso si permette al nostro computer di iniziare e proseguire il colloquio (NEW,ESTABLISHED), mentre in ingresso si accettano solo pacchetti in stato ESTABLISHED, bloccando ogni tentativo di utilizzare dall'esterno i servizi attivi sul computer.

Tuttavia il protocollo FTP richiede, a causa di alcune sue peculiarità, alcune regole dedicate. La connessione aperta dal client (il nostro pc) verso la porta 21 del server FTP è utilizzata per la cosiddetta sessione di controllo, cioè per lo scambio delle informazioni necessarie a guidare l'operazione di trasferimento di files (identificativo dell'utente, password, comandi). Quando, proprio attraverso la sessione di controllo, il client richiede dati (un file, ma anche l'elenco dei file presenti in una directory del server), questi sono inviati dal server su una seconda connessione, anch'essa con trasporto TCP, il cui traffico deve a sua volta essere accettato dal firewall.

Di norma, detta connessione è iniziata dal server, dalla propria porta TCP 20 verso una porta non privilegiata del client, da quest'ultimo indicata nell'ambito della sessione di controllo. E' perciò necessario consentire che da una porta 20 *esterna* al computer sia *iniziata una connessione entrante*: ciò significa aprire nel muro una breccia un poco più seria delle precedenti. Possiamo tuttavia chiedere a Netfilter di verificare che si tratti di una connessione correlata ad un'altra da noi stessi precedentemente aperta verso la porta 21 dello stesso server, in modo da minimizzare il rischio di intrusioni cammuffate da innocue sessioni dati per FTP. Vediamo:

```
iptables -A OUTPUT -o ppp0 -j ACCEPT -p tcp --sport 1024:65535
                                    --dport 20 -m state --state ESTABLISHED
iptables -A INPUT -i ppp0 -j ACCEPT -p tcp --dport 1024:65535 --sport 20
     -m state --state RELATED -m limit --limit 10/minute --limit-burst 15
iptables -A INPUT -i ppp0 -j ACCEPT -p tcp --dport 1024:65535 --sport 20
                                                -m state --state ESTABLISHED
```

Nulla di particolare da segnalare riguardo la regola della catena OUTPUT: è ammesso il traffico TCP diretto da una nostra porta non privilegiata alla porta 20 del server, purché si tratti di pacchetti facenti parte di un colloquio già in corso, in quanto la sessione dati è aperta su iniziativa del server. Notevole, invece, la prima regola di INPUT: sono accettati i pacchetti provenienti da una porta 20 remota, ma solo se si trovano in stato "RELATED". Tale è lo stato attribuito da Netfilter ai pacchetti che richiedono l'apertura di una nuova conessione, quando questa risulti conseguente ad un'altra, aperta in precedenza e non ancora chiusa.

Tale capacità di tracciatura del traffico FTP è implementata in un componente dedicato del kernel di Linux, il modulo *ip_conntrack_ftp*, che deve essere attivato se si desisera che lo stato RELATED sia correttamente riconosciuto e gestito da Netfilter. E' sufficiente un semplice comando, anch'esso lanciato dall'utente *root*:

```
modprobe ip_conntrack_ftp
```

I limiti di frequenza imposti dalla regola evitano che un server male (o maliziosamente) configurato sovraccarichi il nostro kernel inviando una raffica di richieste di connessione. La seconda regola della catena INPUT, infine, abilita la ricezione dei pacchetti in arrivo dal server a connessione ormai operativa: a tale traffico non devono essere applicati vincoli quantitativi.

Ma non è finita: quelle sin qui descritta sono le regole atte alla gestione della sessione dati *attiva*, così chiamata perché spetta al server l'iniziativa di aprirla. Tuttavia, la maggior parte dei server FTP è in grado di gestire anche la sessione *passiva*, aperta dal client verso una porta non privilegiata del server. Dal punto di vista della sicurezza, se ne ricava l'indubbio vantaggio che non è più necessario consentire l'apertura di connessioni in ingresso al firewall; va però sottolineato che tale modalità operativa porta client e server ad operare entrambi su porte non privilegiate, con la conseguente necessità di consentire traffico entrante nel firewall da qualsiasi porta. Ecco le regole necessarie:

```
iptables -A OUTPUT -o ppp0 -j ACCEPT -p tcp --sport 1024:65535
                  --dport 1024:65535 -m state --state RELATED,ESTABLISHED
iptables -A INPUT -i ppp0 -j ACCEPT -p tcp --dport 1024:65535
                          --sport 1024:65535 -m state --state ESTABLISHED
```

La regola di output consente di proseguire connessioni tra porte non privilegiate, o di iniziarne di nuove purché correlate ad altre tuttora aperte: anche in questo caso è indispensabile che il modulo *ip_conntrack_ftp* sia attivo. In input è ammesso solo traffico ricevuto nell'ambito di connessioni già attive.

La scelta se consentire ai client connessioni passive o attive (o, eventualmente, entrambe) spetta all'amministratore. Si tenga presente che, grazie alla tracciatura delle connessioni, la configurazione necessaria al FTP attivo comporta comunque un rischio di intrusione estremamente contenuto ed è la sola a garantire il buon funzionamento del protocollo con tutti i server FTP attivi in Internet.

Dopo avere stabilito le regole necessarie alla gestione dei servizi di nostro interesse, è il momento

di occuparci del protocollo ICMP (Internet Control Message Protocol), dedicato alla trasmissione in Rete di messaggi dedicati al controllo delle connessioni. Su ICMP è basato, ad esempio, l'arcinoto comando *ping*, spesso utilizzato per verificare se un computer remoto sia raggiungibile via IP. Considerato che un utilizzo malizioso di *ping* potrebbe essere volto a determinare un *denial of service*, sovraccaricando il computer bersaglio con l'attività necessaria a rispondere, vogliamo bloccare tutte le *ICMP echo request* in arrivo e ammettere in input esclusivamente i pacchetti ICMP portatori di informazioni (ad esempio, quelli indicanti che un idirizzo non è raggiungibile). Inoltre, non intendiamo privarci dell'opportunità di utilizzare *ping* a nostra volta. Ecco le regole necessarie:

```
iptables -A OUTPUT -o ppp0 -j ACCEPT -p icmp -m state --state NEW
iptables -A INPUT  -i ppp0 -j ACCEPT -p icmp -m state
                                          --state RELATED,ESTABLISHED
```

Come si vede, sono ammessi in uscita esclusivamente pacchetti ICMP in stato NEW: in altre parole, è consentito inziare connessioni ICMP verso macchine remote. Naturalmente, in input devono essere accettati i pacchetti in stato ESTABLISHED, per ricevere le risposte; tuttavia, non è necessario che lo stato ESTABLISHED sia ammesso in output: infatti, ogni "echo request", con relativa risposta, rappresenta una connessione a sé, che nasce con la richiesta medesima e muore all'arrivo della risposta.

In input è necessario però accettare i pacchetti ICMP correlati (RELATED) ad altre connessioni, tramite i quali, ad esempio, si viene informati che un pacchetto precedentemente inviato non può raggiungere il destinatario.

Ancora in tema di comandi di utilità, vale la pena di accennare al comando *traceroute*, che visualizza il percorso compiuto dai nostri dati in viaggio dal computer locale ad uno remoto, elencando tutti i router attraversati: *traceroute* si appoggia al protocollo UDP, utilizzando range specifici di porte. Se desideriamo che il nostro firewall ci consenta di utilizzare *traceroute*, la regola necessaria è:

```
iptables -A OUTPUT -o ppp0 -j ACCEPT -p udp --sport 32769:65535
                                  --dport 33434:33523 -m state --state NEW
```

Sono ammessi in uscita solo pacchetti NEW perché, anche in questo caso, ogni pacchetto di richiesta e relativa risposta esauriscono una connessione. Non è necessaria alcuna regola in input, dal momento che le risposte sono inviate mediante il protocollo ICMP: si tratta di pacchetti evidentemente correlati e, in quanto tali, già ammessi in base alla regola di input ICMP testè descritta.

Ancora un paio di regolette, semplici ma estremamente importanti:

```
iptables -A INPUT  -j STOP
iptables -A OUTPUT -j STOP
```

Grazie ad esse, tutto il traffico non esplicitamente autorizzato dalle regole analizzate sin qui verrà inesorabilmente bloccato da Netfilter, regalandoci un po' di tranquillità nei nostri collegamenti alla Grande Rete. Il nostro firewall è finalmente pronto.

Bootstrap e automazione

Raccogliendo tutti i comandi in uno script, che chiameremo, ad esempio, *rc.firewall*, potremo ricaricare la nostra configurazione in qualunque momento. E' consigliabile che lo script sia accessibile esclusivamente all'amministratore, cosa ottenibile con il comando:

```
chmod 700 rc.firewall
```

Per disattivare tutte le regole in caso di necessità, sarà sufficiente il comando già noto:

```
iptables -F
```

Va sottolineato che Netfilter conserva le regole solo in memoria: ad ogni bootstrap è perciò necessario eseguire lo script per caricare nuovamente la configurazione del firewall. Una soluzione consiste nel modificare lo script *rc.local*, di solito presente nella directory */etc/rc.d*, inserendovi in coda le righe seguenti (nell'ipotesi che anche *rc.firewall* sia installato nella stessa directory):

```
if [ -x /etc/rc.d/rc.firewall ] # se rc.firewall esiste ed e' eseguibile
then                            # allora
        /etc/rc.d/rc.firewall   # eseguilo
fi
```

In alternativa, si può seguire un approccio leggermente più complesso, ma anche più elegante ed efficace, supportato dalla quasi totalità delle distribuzioni GNU/Linux. Dopo avere configurato Netfilter come necessario, si salvino le regole con i seguenti comandi:

```
iptables-save > /etc/sysconfig/iptables
chmod 600 /etc/sysconfig/iptables
```

Il primo comando scrive le regole nel file */etc/sysconfig/iptables*, redirigendovi lo *standard output*; il secondo rende leggibile e modificabile esclusivamente da *root* il file, che verrà automaticamente utilizzato (da *iptables-restore*), in fase di boot, per caricare le regole in Netfilter ancora prima dell'attivazione delle interfacce di rete. Perché il tutto funzioni a dovere, è necessario che l'attivazione del firewall sia abilitata nella configurazione dei *runlevel* con supporto di rete (di norma i *runlevel* 2, 3, 4 e 5). Detta configurazione può essere modificata, in caso di necessità, mediante una delle apposite interfacce grafiche (come *tksysv*), oppure... a manina.

In pratica, si tratta di creare, nelle directory degli script di avviamento dei *runlevel* opportuni, un *symbolic link* allo script di gestione del firewall. Detto link è prefissato con "S", perciò richiede lo startup del servizio. Analogamente, nelle directory degli script di avviamento dei *runlevel* 1 e 6 devono essere creati i link simbolici prefissati con "K", che richiedono di disattivare (*kill*) il servizio.

```
for I in 2 3 4 5
do
cd /etc/rc.d/rc${I}.d
ln -s ../init.d/iptables S01iptables
done

for I in 1 6
do
cd /etc/rc.d/rc${I}.d
ln -s ../init.d/iptables K99iptables
```

```
done
```

I link simbolici sono eseguiti in ordine alfabetico crescente: di conseguenza, il numero che segue il prefisso "S" o "K" determina il momento di esecuzione. In altre parole, il firewall sarà attivato prima di ogni altro servizio (networking compreso) e disattivato dopo tutti gli altri. Inutile dire che se la configurazione dei *runlevel* prevede già la gestione del firewall, i link simbolici sono presenti nelle directory indicate e non vi è necessità di crearli.

Infine, come anticipato, a supporto delle regole impostate in Netfilter deve essere attivato il modulo del kernel che presiede alla tracciatura delle connessioni FTP. Il suo caricamento in fase di bootstrap può essere richiesto inserendone il nome nel file */etc/modules*, in una nuova riga: è possibile farlo aprendo il file con un editor (ad esempio *vi*), oppure con il comando:

```
echo ip_conntrack_ftp >> /etc/modules
```

Possiamo verificare la configurazione del firewall, anche successivamente al bootstrap, con il comando:

```
iptables -L -n -v --line-numbers
```

il quale elenca ("-L") in modo particolareggiato ("-v", *verbose mode*), ma senza effettuare la risoluzione dei nomi host in indirizzi ip ("-n", *no name resolution*), tutte le regole attive, numerandole ("--line-numbers") nell'ambito delle rispettive catene di appartenenza.

Gestire "al volo" le regole

Tutte le regole sin qui esaminate sono state via via aggiunte in modalità *append* alla configurazione di Netfilter: in altri termini, ciascuna di esse è stata inserita in coda a tutte quelle precedentemente definite nell'ambito della catena di pertinenza. Tuttavia, *iptables* consente di inserire una nuova regola in una posizione precisa, nell'ambito di una specifica *chain*. Ad esempio, il comando

```
iptables -I INPUT 3 -j STOP -p tcp
```

inserisce nella terza posizione della catena INPUT la nuova regola. Analogamente, si può sostituire una regola indicandone, anche in questo caso, la posizione, e specificando le nuove condizioni:

```
iptables -R INPUT 3 -j STOP -p tcp
```

sostituisce la terza regola di input. Infine, è possibile eliminare una regola:

```
iptables -D INPUT 3
```

ha l'effetto di cancellare la regola numero 3 della catena di input. In alternativa, è valida la seguente sintassi:

```
iptables -D INPUT -j STOP -p tcp
```

Infatti, se non si indica la posizione della regola, ma la si "riscrive", iptables cerca nella catena una regola specificata in modo identico e la elimina.

I comandi testè presentati possono essere molto utili, ad esempio, quando si desideri modificare la configurazione di Netfilter in conseguenza di particolari eventi: ad esempio, rendere più efficace il *packet filtering* in input e output in relazione all'indirizzo IP assegnato dal Provider al momento della connessione a Internet. In questo caso, uno stratagemma efficace potrebbe consistere nell'inserire nella configurazione iniziale di Netfilter una o più regole "trasparenti" al traffico, in qualità di segnaposto, per poi inserire *nel punto giusto* le regole basate sul nuovo indirizzo IP, conoscibile, ad esempio, col comando

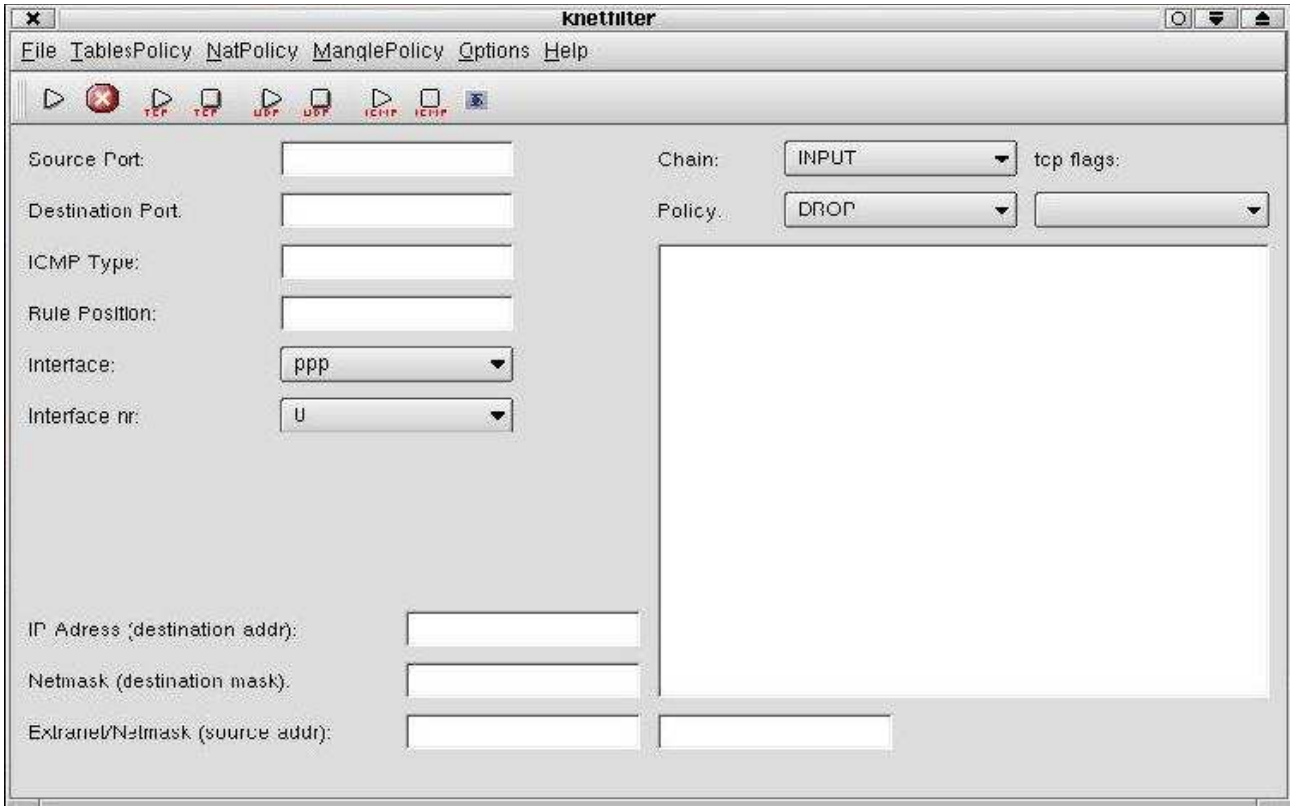
```
/sbin/ifconfig ppp0
```

Il comando *ifconfig* è disponibile anche agli utenti non dotati di privilegi amministrativi.

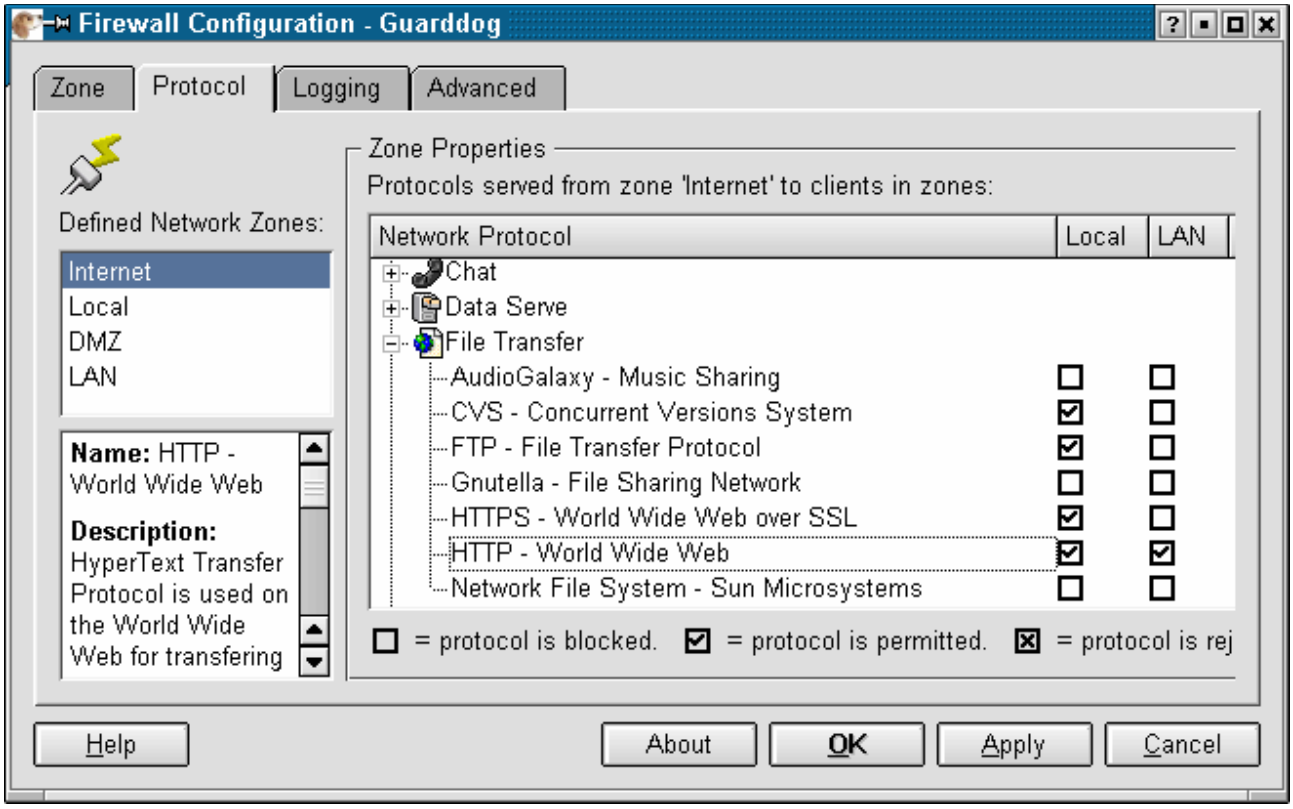
Interfacce grafiche

Irrimediabilmente difficile? No. Quanto sin qui esposto non ha solamente lo scopo di illustrare la sintassi di configurazione di Netfilter, bensì, soprattutto, quello di chiarire la logica con cui è necessario affrontare uno degli aspetti del *firewalling*: il controllo del traffico mediante *packet filtering*. Per chi non ama dedicare tempo all'analisi e preferisce passare immediatamente all'azione, sono disponibili strumenti di configurazione dotati di interfaccia grafica, certo più intuitiva della riga di comando.

Tra questi, vale la pena di citare *Knetfilter*, che consente il completo controllo delle opzioni di Netfilter. La home page si trova all'indirizzo *http://expansa.sns.it/knetfilter*; da essa è stato prelevato lo *screenshot* sottostante.

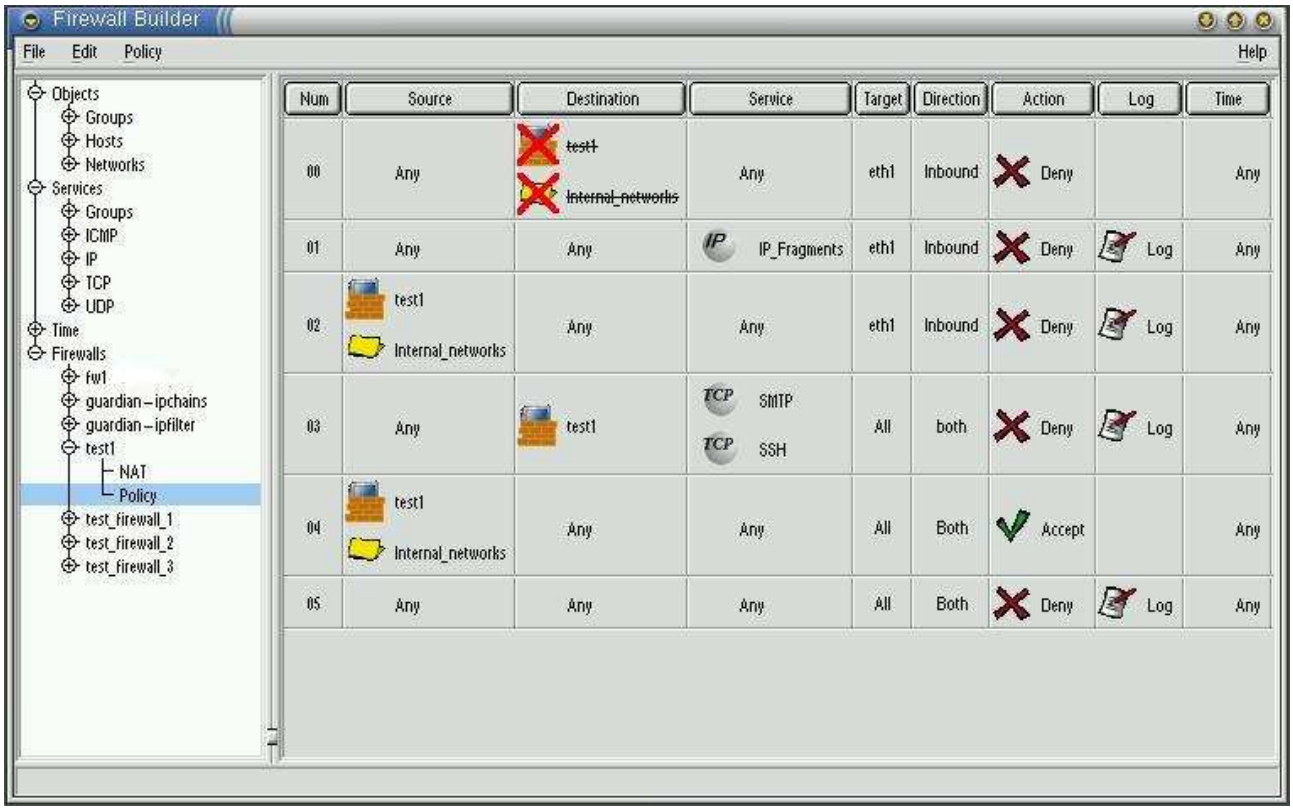


Un'altro interessante strumento di configurazione per Netfilter è *Guarddog*, che implementa una logica a "zone", per ciascuna delle quali è possibile definire regole per la gestione del traffico. *Guarddog* è reperibile all'URL *http://www.simonzone.com/software/guarddog*; anche in questo caso lo screenshot presentato di seguito è stato prelevato direttamente dal sito.



Infine, un cenno merita *FwBuilder*, la cui home page è *http://www.fwbuilder.org*. Si tratta di uno strumento multipiattaforma per la gestione e la configurazione di firewall, operante secondo una logica *object-oriented* e in grado di offrire funzionalità *drag-and-drop*. Tra i firewall supportati, oltre naturalmente a Netfilter, vi sono OpenBSD PF e Cisco PIX.

Si tratta di un programma in grado di gestire diverse implementazioni di firewall, delle quali Netfilter/iptables è una, ma non l'unica. Sono disponibili, oltre ai sorgenti, anche i pacchetti RPM per le distribuzioni più note; oltre che dal sito indicato, il programma è scaricabile anche dal famoso sito SourceForge (*http://sourceforge.net/project/showfiles.php?group_id=5314*). Eccone uno screenshot:



Riferimenti

La documentazione riguardante Netfilter e iptables disponibile in Rete è vastissima, perciò i riferimenti indicati di seguito rappresentano una minima parte di ciò che, con semplici ricerche mirate, è possibile reperire: tra i riferimenti più interessanti si può certamente citare *http://www.netfilter.org*, home page del progetto.

Una vastissima collezione di links a siti e risorse riguardanti Netfilter si trova all'indirizzo *http://www.linuxguruz.org/iptables*, e può valere la pena di cercare anche presso *http://www.linuxdoc.org*, che rappresenta il sito di riferimento per la documentazione in ambito GNU/Linux.

Infine, all'URL *http://iptables-tutorial.frozentux.net/iptables-tutorial.html* è disponibile un tutorial particolareggiato e di facile comprensione.

In conclusione...

Le regole commentate permettono di creare una configurazione di base, sufficiente a garantire una protezione piuttosto efficace contro le intrusioni. Tuttavia, esse non esauriscono l'insieme, vastissimo, delle possibilità operative di Netfilter, né, tantomeno, possono essere considerate un baluardo invalicabile. Esse rappresentano, piuttosto, un efficace punto di partenza per chi intenda approfondire l'esame delle tecniche di *firewalling* e mettere a punto configurazioni più sofisticate e,

soprattutto, costruite *ad hoc*.

Anche dopo essersi dotati di un firewall configurato nel migliore dei modi, comunque, non va dimenticato che la sicurezza di un sistema non risiede mai in una sola componente. In altre parole, non è sufficiente un firewall efficace per ritenersi al sicuro da ogni possibile forma di attacco. Ad esempio, contro gli allegati alla posta portatori di virus o altre soprese, un firewall può ben poco, se non è affiancato da un robusto sistema di scansione delle email. Questo, a sua volta, non rappresenta che una beata illusione se non è tenuto costantemente aggiornato.

Ma, soprattutto, nessun sistema può garantire sicurezza al cento per cento da qualsiasi forma di attacco: la nostra attenzione, unita ad almeno una minima consapevolezza tecnologica, è destinata a rimanere un'arma indispensabile e insostituibile.

Stefano Barni – stefano.barni@zeusnews.com

Appendice: un esempio di rc.firewall

Di seguito è riportato un esempio di *script file* rc.firewall: esso definisce le regole di packet filtering descritte nell'articolo, con particolare riferimento ad una configurazione di collegamento con indirizzo internet assegnato dinamicamente dal Provider.

I comandi inclusi nello script sono raggruppati per funzionalità implementate; ogni gruppo è introdotto da un breve commento esplicativo. Nella parte finale è dato, inoltre, un esempio di comandi atti a valorizzare opportunamente alcune variabili, interne al kernel di GNU/Linux, preposte al controllo di particolari tipologie di traffico: essi hanno l'obiettivo di incrementare il livello di sicurezza garantito dalla configurazione proposta.

```
#!/bin/bash
#############################################################################
#
# rc.firewall
#
# script di configurazione per Netfilter: il computer dispone di una
# connessione non permanente a Internet con indirizzo IP dinamico e di una
# scheda ethernet per il collegamento a una LAN privata
#
# rc.firewall, lanciato senza opzioni, configura Netfilter e kernel; con
# l'opzione -v configura Netfilter e kernel e visualizza la configurazione
#
# nota: i range di porte utilizzati per la definizione delle regole sono qui
# specificati con la notazione abbreviata (gli estremi 0 e 65535 sono omessi)
#
#############################################################################

#############################################################################
#
# lo script deve essere eseguito da root: se le permissions del file non sono
# corrette (700) e ne consentono l'esecuzione anche da parte di utenti non
# privilegiati, entra in azione il controllo che segue
#
#############################################################################

if [ “$(id -u)” -ne “0” ]
then
      echo "$0: You must be root to run this script" >&2
```

```
      exit 1
fi

#############################################################################
#
# variabili di environment: modificare secondo la propria configurazione
#
#############################################################################

LAN_NET="192.168.0.0/24"   # range di indirizzi della LAN
LAN_IP="192.168.0.1"       # indirizzo del computer sulla LAN

LAN_IF=”eth0”              # interfaccia di rete LAN
INET_IF=”ppp0”             # interfaccia virtuale di collegamento a Internet

#############################################################################
#
# setup preliminare
#
#############################################################################

modprobe ip_conntrack_ftp
iptables -F
iptables -X

#############################################################################
#
# definizione della chain STOP (logga e scarta i pacchetti)
#
#############################################################################

iptables -N STOP
iptables -A STOP -j LOG
iptables -A STOP -j DROP

#############################################################################
#
# tutto il traffico che richiede routing viene scartato
#
#############################################################################

iptables -A FORWARD -j STOP

#############################################################################
#
# tutto il traffico sospetto (pacchetti non validi e frammenti) viene scartato
#
#############################################################################

iptables -A INPUT  -j STOP -m state --state INVALID
iptables -A INPUT  -j STOP -f
iptables -A OUTPUT -j STOP -m state --state INVALID
iptables -A OUTPUT -j STOP -f

#############################################################################
#
# loopback: tutto il traffico locale e' abilitato senza limitazioni
#
#############################################################################

iptables -A INPUT  -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

#############################################################################
#
# lan: il traffico da e verso la rete locale e' abilitato con la sola
# esclusione dei pacchetti con indirizzi non lan o sospetti
#
```

```
##############################################################################

iptables -A INPUT  -i $LAN_IF -j STOP -s   $LAN_IP
iptables -A INPUT  -i $LAN_IF -j STOP -d ! $LAN_IP
iptables -A INPUT  -i $LAN_IF -j STOP -s ! $LAN_NET
iptables -A INPUT  -i $LAN_IF -j ACCEPT
iptables -A OUTPUT -o $LAN_IF -j STOP -d   $LAN_IP
iptables -A OUTPUT -o $LAN_IF -j STOP -s ! $LAN_IP
iptables -A OUTPUT -o $LAN_IF -j STOP -d ! $LAN_NET
iptables -A OUTPUT -o $LAN_IF -j ACCEPT

##############################################################################
#
# internet: sono scaratati i pacchetti provenienti o destinati a reti private
# o indirizzi riservati (classi a, b, c, e)
#
##############################################################################

iptables -A INPUT  -i $INET_IF -j STOP -s 10.0.0.0/8
iptables -A INPUT  -i $INET_IF -j STOP -d 10.0.0.0/8
iptables -A INPUT  -i $INET_IF -j STOP -s 172.16.0.0/12
iptables -A INPUT  -i $INET_IF -j STOP -d 172.16.0.0/12
iptables -A INPUT  -i $INET_IF -j STOP -s 192.168.0.0/16
iptables -A INPUT  -i $INET_IF -j STOP -d 192.168.0.0/16
iptables -A INPUT  -i $INET_IF -j STOP -s 240.0.0.0/5
iptables -A INPUT  -i $INET_IF -j STOP -d 240.0.0.0/5
iptables -A OUTPUT -o $INET_IF -j STOP -s 10.0.0.0/8
iptables -A OUTPUT -o $INET_IF -j STOP -d 10.0.0.0/8
iptables -A OUTPUT -o $INET_IF -j STOP -s 172.16.0.0/12
iptables -A OUTPUT -o $INET_IF -j STOP -d 172.16.0.0/12
iptables -A OUTPUT -o $INET_IF -j STOP -s 192.168.0.0/16
iptables -A OUTPUT -o $INET_IF -j STOP -d 192.168.0.0/16
iptables -A OUTPUT -o $INET_IF -j STOP -s 240.0.0.0/5
iptables -A OUTPUT -o $INET_IF -j STOP -d 240.0.0.0/5

##############################################################################
#
# internet: sono bloccati i pacchetti da e per indirizzi di loopback
#
##############################################################################

iptables -A INPUT  -i $INET_IF -j STOP -s 127.0.0.0/8
iptables -A INPUT  -i $INET_IF -j STOP -d 127.0.0.0/8
iptables -A OUTPUT -o $INET_IF -j STOP -s 127.0.0.0/8
iptables -A OUTPUT -o $INET_IF -j STOP -d 127.0.0.0/8

##############################################################################
#
# internet: sono bloccati i pacchetti da e per indirizzi di broadcast atipici
#
##############################################################################

iptables -A INPUT  -i $INET_IF -j STOP -s 0.0.0.0
iptables -A INPUT  -i $INET_IF -j STOP -d 0.0.0.0
iptables -A INPUT  -i $INET_IF -j STOP -s 255.255.255.255
iptables -A INPUT  -i $INET_IF -j STOP -d 255.255.255.255
iptables -A OUTPUT -o $INET_IF -j STOP -s 0.0.0.0
iptables -A OUTPUT -o $INET_IF -j STOP -d 0.0.0.0
iptables -A OUTPUT -o $INET_IF -j STOP -s 255.255.255.255
iptables -A OUTPUT -o $INET_IF -j STOP -d 255.255.255.255

##############################################################################
#
# internet: i multicast (classe d) sono accettati in ingresso se l'indirizzo
# multicast e' quello di destinazione; in uscita sono sempre scartati
#
##############################################################################
```

```
iptables -A INPUT  -i $INET_IF -j STOP   -s 224.0.0.0/4
iptables -A INPUT  -i $INET_IF -j ACCEPT -d 224.0.0.0/4 -p udp \
                                    -m limit --limit 5/second --limit-burst 10
iptables -A OUTPUT -o $INET_IF -j STOP   -s 224.0.0.0/4
iptables -A OUTPUT -o $INET_IF -j STOP   -d 224.0.0.0/4

##############################################################################
#
# internet: bloccati tutti i pacchetti in ingresso diretti a indirizzi diversi
# da quello locale e tutti i pacchetti in uscita con indirizzo di origine
# diverso da quello locale
#
##############################################################################

#iptables -A INPUT  -i $INET_IF -j DROP
#iptables -A OUTPUT -o $INET_IF -j DROP

##############################################################################
#
# internet: dns (solo query)
#
##############################################################################

iptables -A OUTPUT -o $INET_IF -j ACCEPT -p udp --sport 1024: --dport 53 \
                                                -m state --state NEW,ESTABLISHED
iptables -A INPUT  -i $INET_IF -j ACCEPT -p udp --dport 1024: --sport 53 \
                                                   -m state --state ESTABLISHED

##############################################################################
#
# internet: ftp (controllo), smtp, http, pop3, https
#
##############################################################################

iptables -A OUTPUT -o $INET_IF -j ACCEPT -p tcp --sport 1024: \
        -m multiport --dports 21,25,80,110,443 -m state --state NEW,ESTABLISHED
iptables -A INPUT  -i $INET_IF -j ACCEPT -p tcp --dport 1024: \
            -m multiport --sports 21,25,80,110,443 -m state --state ESTABLISHED

##############################################################################
#
# internet: ftp (dati / active); per attivare le regole rimuovere i
# caratteri “#” in testa alle righe
#
##############################################################################

#iptables -A INPUT  -i $INET_IF -j ACCEPT -p tcp --dport 1024: --sport 20 \
#          -m state --state RELATED -m limit --limit 10/minute --limit-burst 15
#iptables -A INPUT  -i $INET_IF -j ACCEPT -p tcp --dport 1024: --sport 20 \
#                                                   -m state --state ESTABLISHED
#iptables -A OUTPUT -o $INET_IF -j ACCEPT -p tcp --sport 1024: --dport 20 \
#                                                   -m state --state ESTABLISHED

##############################################################################
#
# internet: ftp (dati / passive)
#
##############################################################################

iptables -A OUTPUT -o $INET_IF -j ACCEPT -p tcp --sport 1024: --dport 1024: \
                                              -m state --state RELATED,ESTABLISHED
iptables -A INPUT  -i $INET_IF -j ACCEPT -p tcp --dport 1024: --sport 1024: \
                                                   -m state --state ESTABLISHED

##############################################################################
#
# internet: traceroute solo verso host remoti
#
```

```
##############################################################################

iptables -A OUTPUT -o $INET_IF -j ACCEPT -p udp --sport 32769: \
                                        --dport 33434:33523 -m state --state NEW

##############################################################################
#
# internet: icmp ok verso host remoti; in ingresso solo in risposta a
# connessioni gia' attive
#
##############################################################################

iptables -A OUTPUT -o $INET_IF -j ACCEPT -p icmp -m state --state NEW
iptables -A INPUT  -i $INET_IF -j ACCEPT -p icmp \
                                           -m state --state RELATED,ESTABLISHED

##############################################################################
#
# tutto cio' che non e' stato esplicitamente ammesso viene bloccato ora
#
##############################################################################

iptables -A INPUT  -j STOP
iptables -A OUTPUT -j STOP

##############################################################################
#
# kernel settings: i comandi che seguono richiedono al kernel di Linux di
# scartare alcune tipolgie di pacchetti che potrebbero essere utilizzate
# per diversi tipi di attacco. Non si tratta di comandi dati a Netfilter:
# vengono valorizzate variabili di stato interne al kernel scrivendo i
# valori opportuni sul filesystem virtuale /proc
#
##############################################################################

echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
echo 1 > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
echo 0 > /proc/sys/net/ipv4/conf/all/accept_source_route
echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects

##############################################################################
#
# display della configurazione attivata
#
##############################################################################

if [ "$1" = "-v" ]
then
        iptables -L -nv --line-numbers
        echo
        echo kernel: icmp_echo_ignore_broadcasts = \
                          $(cat /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts)
        echo kernel: icmp_ignore_bogus_error_responses = \
                        $(cat /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses)
        echo kernel: accept_source_route = \
                             $(cat /proc/sys/net/ipv4/conf/all/accept_source_route)
        echo kernel: accept_redirects = \
                               $(cat /proc/sys/net/ipv4/conf/all/accept_redirects)
fi

##############################################################################
#
# fine
#
##############################################################################

exit 0
```