http://www.segfault.it/

Lo Staff di Segfault.it

Presenta:

Corso di base di Netfilter/iptables

Versione del documento 1.0 (Aprile 2008)

Sommario

1 Introduzione: Giorno 1

La prerogativa principale di un firewall è senza ombra di dubbio quella di proteggere un PC o una rete da attacchi o traffico indesiderato. E' insomma lo strumento difensivo che sta più in prima linea tra tutti quelli che è possibile schierare a favore della propria sicurezza. A causa di questa sua **caratteristica** è però anche il sistema che merita più attenzioni, almeno nelle fasi iniziali di messa in esercizio. Vediamo come configurarlo semplicemente disponendo di una comune distribuzione Linux. Questa guida è rivolta principalmente ad un pubblico novizio o con conoscenze intermedie sull'argomento. Naturalmente è consigliata la conoscenza dei rudimenti del TCP/IP.

1.1 Iptables ed i primi rudimenti

Iptables è la componente firewall **standard** di Linux sin dall'introduzione dei kernel **2.4**. Si compone di **Netfilter** (una serie di patch per il kernel) ed alcuni strumenti e librerie che girano in **user-land** (tra le quali anche l'eseguibile iptables). Grazie ad essi è possibile trasformare una postazione casalinga o d'ufficio in un firewall di alto livello che non ha nulla da invidiare alle soluzioni commerciali più competitive presenti oggi sul mercato. Come già detto, iptables viene comunemente installato di default nella maggior parte delle distribuzioni esistenti. Per determinarne la sua presenza nel vostro sistema, digitate da linea di comando:

```
[root@FireWater tmp]# which iptables
/sbin/iptables
```

Viene ritornato il percorso completo in cui si trova il file eseguibile ("`/sbin/iptables`" in questo caso). Il funzionamento di iptables si basa sul meccanismo delle **regole** che istruiscono il sistema sul traffico che deve essere scartato o che deve essere lasciato passare. Per ogni regola da inserire si digita il nome del comando (`iptables`) seguito da una serie di opzioni. La pagina di manuale del programma (`man iptables`) è molto esplicativa in tal senso, ma la troppa carne al fuoco può certamente causare più confusione che altro, soprattutto fra i novizi. Per questo motivo utilizzeremo un approccio passo-passo che vi consentirà di comprendere, attraverso appropriati esempi, come iptables sia in grado di operare. Ciò vi aiuterà in seguito a configurare il vostro firewall Linux in modo efficace ed opportuno contro attacchi o probing di vario genere, agendo direttamente sulle varie **tabelle** e **chain** supportate.

Prima di proseguire introduciamo alcuni concetti di base. Il comando:

```
# iptables –L
```

visualizza a video le regole al momento presenti all'interno delle chain `INPUT`, `OUTPUT` e `FORWARD` della tabella `FILTER` (non preoccupatevi per il momento se quello che viene detto sembra ostrogoto!). Una risposta come quella che segue indica che le chain sono vuote (cioè non contengono ancora nessuna regola):

```
Chain INPUT (policy ACCEPT)
target     prot  opt   source      destination

Chain FORWARD (policy ACCEPT)
Target     prot  opt   source      destination

Chain OUTPUT (policy ACCEPT)
target     prot  opt   source      destination
```

Alcune distribuzioni Linux in fase di post-installazione potrebbero aggiungere delle regole proprie. Per evitare interferenze con gli esempi che più avanti mostreremo, è consigliabile azzerare tutto digitando da linea di comando:

```
# iptables –F
# iptables –F –t nat
# iptables –F –t mangle
```

Nel corso del testo si indicherà con "*Linux box*" o "*postazione linux*" il sistema utilizzato per testare tutti gli esempi qui presentati. Presupporremo che tale *Linux box* sarà dotata di un'interfaccia Ethernet (`eth0`) che la collegherà ad altri eventuali PC presenti nella LAN e di un'interfaccia `ppp0` che gli consentirà di accedere ad Internet.

In alcuni degli esempi che proporremo, considereremo l'indirizzo IP `192.168.0.4` assegnato all'interfaccia `eth0`. Gli esempi che ne faranno uso potranno essere testati nell'ambiente di rete personale semplicemente sostituendo questo indirizzo con il vostro.

1.2 Tabelle (Tables) e Catene (Chain)

Le regole inserite attraverso iptables vengono raggruppate all'interno di **catene** (*chain*) che a loro volta sono contenute dentro delle **tabelle** (*tables*). Mentre le tabelle definiscono il tipo di operazione che è possibile svolgere sui pacchetti, le catene definiscono il modo in cui essi sono trattati. L'idea può essere semplificata immaginando un pacchetto come un'entità che deve attraversare diverse fasi e diverse *catene* di elaborazione prima che il sistema possa decidere cosa farne.

Le catene definite da iptables sono:

- **INPUT**: lavora sui pacchetti che fanno il loro ingresso nel sistema;
- **OUTPUT**: lavora sui pacchetti creati localmente ed in uscita dal sistema;
- **FORWARD**: lavora sui pacchetti che vengono instradati dal sistema, ovvero quelli per il quale il sistema agisce come un router (cioè generati da un PC diverso da quelle locale ed a loro volta destinati ad un altro sistema della rete);
- **PREROUTING**: passano attraverso questa catena i pacchetti in entrata. Le regole su di essi vengono applicate prima che il sistema abbia preso decisioni sul loro instradamento;
- **POSTROUTING**: passano attraverso questa catena i pacchetti in uscita. Le regole su di essi vengono applicate dopo che il sistema ha preso decisioni sul loro instradamento (ovvero dopo aver consultato la tabella di routing);

Le tabelle implementate da Netfilter/iptables sono invece tre:

- **filter**: ogni pacchetto che deve essere filtrato passa da qui. Contiene le catene **INPUT**, **OUTPUT** e **FORWARD**.
- **nat**: ogni pacchetto relativo ad una connessione *mascherata* (si apprenderà di più sul termine in seguito) attraversa questa tabella. Sono valide solo le catene **OUTPUT**, **POSTROUTING** e **PREROUTING**.
- **mangle**: passano da qui i pacchetti le cui opzioni devono essere modificate. Nei kernel più recenti questa tabella contiene tutte le catene.

L'utente ha la possibilità di creare nuove tabelle oltre quelle predefinite, ma questo aspetto di Netfilter/Iptables non verrà preso in considerazione (almeno non in questa prima versione del tutorial).

1.3 I firewall e le policy

Un firewall può essere configurato di base in due modi diversi:

1) Garantendo l'accesso a tutto ciò che non è espressamente negato.
2) Negando l'accesso a tutto ciò che non è esplicitamente consentito.

Queste modalità prendono il nome di "**policy**". Di default iptables imposta la policy di tutte le sue chain ad "`ACCEPT`", rispecchiando così in toto il comportamento descritto nel punto **1**. Ma vediamo pregi e difetti di entrambi i tipi di policy.

Prendiamo come esempio il caso standard di un sistema in cui non è ancora stata inserita alcuna regola:

```
# iptables –L

Chain INPUT (policy ACCEPT)
target      prot  opt   source      destination

Chain FORWARD (policy ACCEPT)
Target      prot  opt   source      destination

Chain OUTPUT (policy ACCEPT)
target      prot  opt   source      destination
```

Accanto al nome di ogni catena (`INPUT`, `FORWARD` ed `OUTPUT`) viene indicato il tipo di policy in essere. Ad esempio la policy `ACCEPT` nelle chain `INPUT` ed `OUTPUT` della tabella `FILTER` sta ad indicare che tutto il traffico in entrata ed in uscita dalla Linux Box verrà accettato. Un ping o una richiesta di collegamento verranno fatti passare senza nessun problema.

Proviamo adesso a modificare la policy di default della chain `INPUT` da "`ACCEPT`" a "`DROP`" passando così al caso riportato nel precedente punto **2**:

```
# iptables -P INPUT DROP
```

Un "`iptables -L`" mostra lo stato della modifica appena apportata:

```
Chain INPUT (policy DROP)
target      prot  opt   source      destination

Chain FORWARD (policy ACCEPT)
Target      prot  opt   source      destination

Chain OUTPUT (policy ACCEPT)
target      prot  opt   source      destination
```

Provate adesso a lanciare una sessione ssh verso la vostra linux box da un altro PC:

```
# ssh 192.168.0.4
```

….non sarà possibile connettersi. Dopo un po' verrà mostrato a video il messaggio "`ssh: connect to host 192.168.0.4 port 22: Connection timed out`".

La policy "`DROP`" della chain `INPUT` causa il rifiuto di tutto il traffico in entrata. I problemi comunque si ripercuoteranno anche in uscita, vale a dire non avrete alcuna possibilità di dialogare con il mondo esterno:

```
# ping www.google.it
PING 66.102.9.99 (66.102.9.99) 56(84) bytes of data
…
```

In realtà le richieste arrivano a destinazione e le risposte tornano indietro (date pure un occhiata con `tcpdump` se vi pare), ma la tipologia di policy prescelta fa si che queste vengano bloccate e non lasciate passate in **user-land** al programma `ping` per la corretta elaborazione.

Ma allora quando conviene adottare una policy e quando un'altra?

Quando decidete di impostare la policy di una o più chain del firewall ad "`ACCEPT`", dovete inserire delle regole che specificano chiaramente cosa volete bloccare, altrimenti la vostra linux box sarà raggiungibile a chiunque ed in qualunque forma.. Il pro di questo approccio è certamente la sua semplicità d'utilizzo. Il contro è che un errore nella definizione delle regole o una dimenticanza potrebbero consentire a certo traffico che pensavate di aver bloccato, di passare liberamente il vostro firewall.

Quando decidete viceversa di impostare la policy di una o più chain del vostro firewall a "`DROP`", dovete invece inserire delle regole che specificano esplicitamente cosa volete far passare, altrimenti le connessioni verranno bloccate. Questo approccio è certamente più difficoltoso perché richiede l'esatta conoscenza del tipo di traffico che desiderate far fluire liberamente (e quindi presuppone un'attenta pianificazione preventiva delle regole). Un minimo errore nella definizione delle stesse però precluderà la comunicazione con certi host! Rimane il fatto comunque che questo tipo di policy è l'ideale per raggiungere livelli di blindatura elevati.

Inizialmente è consigliato iniziare con la policy `ACCEPT`. Una volta ottenuto un buon grado di preparazione con **iptables**, varrà la pena migrare verso la policy più restrittiva `DROP`. Per il momento rimettiamo quindi tutto come prima:

```
# iptables -P INPUT ACCEPT
```

`ACCEPT` e `DROP` sono quelli che in gergo vengono definiti target. Un target indica l'effetto finale che la regola avrà sul pacchetto. Come intuibile corrispondono alle parole italiane **Accetta** e **Scarta**.

1.4 Scrivere la prima regola

Adesso è venuto il momento di vedere che forma ha una regola e che influenza può avere sul sistema. Digitate da linea di comando:

```
# iptables –A INPUT –p tcp –s 0/0 –-dport 22 –j DROP
```

Osserviamo in dettaglio le singole opzioni che la compongono:

- l'opzione "**-A**" aggiunge una regola dopo quelle già presenti nella chain "**INPUT**". Nel nostro caso non essendocene ancora alcuna dichiarata, questa viene posizionata in testa.
- l'opzione "**-p**" specifica il protocollo della regola (cioè "**tcp**")
- l'opzione "**-s**" indica l'indirizzo IP sorgente delle regola. "**0/0**" significa "*qualsiasi IP*". Si compone della parte che precede lo slash (il vero e proprio indirizzo IP) e da quella che lo succede (la maschera di sottorete)
- l'opzione "**--dport**" indica la porta di destinazione della regola. In questo caso viene specificato il valore "22" che corrisponde al servizio **ssh**.
- l'opzione "**-j**" indica il target, ovvero cosa fare in caso iptables rilevi un pacchetto che coincida con le caratteristiche definite nella regola. In questo caso "`DROP`" significa "*scarta il pacchetto*". Al contrario "`ACCEPT`" lo accetta.

Questa regola non fa altro che bloccare (`-j DROP`) qualsiasi pacchetto tcp (`-p tcp`) proveniente da qualunque indirizzo IP mittente (`-s 0/0`) che tenti di collegarsi al servizio **ssh** (`--dport 22`) di qualsiasi interfaccia della vostra linux box.

Se lo si desidera (ad esempio perché si dispone di più interfacce di rete) è possibile specificare l'esatto indirizzo IP di destinazione alla quale la regola si applica con l'opzione "`-d`". Ecco l'esempio rivisto:

```
# iptables –A INPUT –p tcp –s 0/0 –d 192.168.0.4 –-dport 22 –j DROP
```

In tal caso verrebbero bloccate solo le sessioni ssh avviate verso l'IP locale "`192.168.0.4`". Questa regola non avrebbe quindi effetto su altri ipotetici indirizzi IP assegnati al sistema.

Con **iptables** è anche possibile specificare direttamente l'interfaccia sulla quale la regola deve avere effetto, omettendo l'indirizzo IP specifico. Ciò diviene particolarmente utile quando si ha a che fare con indirizzi che variano di continuo (magari perchè assegnati dinamicamente dal provider che fornisce la connettività ad Internet). In questo modo si evita di riscrivere tutte le regole cambiando IP di volta in volta (il che di per sé equivale ad un grosso risparmio in salute :)

Quello che segue è un esempio lampante di quanto si è fino ad ora detto::

```
# iptables –A INPUT –p tcp –i eth0 –s 0/0 –-dport 22 –j DROP
```

Anche se l'indirizzo IP assegnato all'interfaccia "`eth0`" (o qualsiasi altra desiderata) dovesse variare di volta in volta, le regole scritte in questo modo continueranno ugualmente a funzionare senza alcuna modifica.

La regola scritta in una qualsiasi delle tre forme finora presentate produce lo stesso identico risultato finale. L'utente deve solo scegliere quella più appropriata per le proprie esigenze. Questa scelta è dettata principalmente dal proprio ambiente di rete e dal grado di praticità che si intende ottenere. Detto questo vale davvero la pena constatare la solidità della regola prescelta.

Per testarne il corretto funzionamento digitate da un altro PC collegato alla rete:

```
# telnet 192.168.0.4 22
```

L'output a video rimarrà bloccato in "`Trying 192.168.0.4...`". Lasciatelo così per qualche secondo. Se il servizio non risponde ciò indica che la vostra linux box blocca correttamente

qualsiasi tentativo di collegamento da parte di qualunque indirizzo IP verso la porta **ssh**. Complimenti! Avete configurato con successo la vostra prima regola! Per tornare al prompt dei comandi, premete Ctrl+C.

Nota: Ogni volta che un'opzione viene omessa (ad esempio `--sport` che indica la porta sorgente della regola o "`-d`" già vista in precedenza) iptables imposta di default il corrispondente valore a "**qualsiasi**".

1.5 Propedeuticità delle regole

Analizziamo adesso dalla tabella delle regole la chain `INPUT` per vedere come il suo contenuto è cambiato a seguito dell'inserimento della prima regola:

```
# iptables –L INPUT –n

Chain INPUT (policy ACCEPT)
target      prot  opt   source      destination
DROP        tcp   --    0.0.0.0/0   0.0.0.0/0          tcp dpt:22
```

Ogni regola inserita viene analizzata da iptables in modo sequenziale, cioè dalla prima all'ultima. All'arrivo di un pacchetto, iptables analizza una ad una tutte le regole contenute nella chain `INPUT` esattamente nell'ordine in cui queste sono dichiarate. Quando ne trova una collimante (ovvero che coincide) con il pacchetto in questione, iptables interrompe la ricerca ed esegue l'azione associata alla regola (blocca, accetta, etc..).

Da ciò si può dedurre chiaramente che alcune regole devono essere necessariamente dichiarate prima di altre, pena il loro corretto funzionamento. Ad esempio se volete che `192.168.0.2` (un altro host della vostra rete locale) ed `x.x.x.x` (l'IP remoto del vostro ufficio) siano in grado di collegarsi al servizio ssh del firewall, potete digitare:

```
# iptables –A INPUT –p tcp –s 192.168.0.2 –-dport 22 –j ACCEPT
# iptables –A INPUT –p tcp –s x.x.x.x –-dport 22 –j ACCEPT
```

Provate adesso a collegarvi da `192.168.0.2` o dall'ufficio al servizio ssh del firewall digitando:

```
# ssh 192.168.0.4
```

oppure:

```
# ssh ip_pubblico_del_firewall
```

non riuscirete a farlo! In questo caso infatti non avete considerato la propedeuticità delle regole. Vediamo cosa contiene, a questo punto della discussione, la tabella delle regole appositamente numerata per l'occasione:

```
# iptables –L INPUT –n

Chain INPUT (policy ACCEPT)
Target      prot  opt   source      destination
1)DROP      tcp   --    0.0.0.0/0   0.0.0.0/0         tcp dpt:22
2)ACCEPT    tcp   --    192.168.0.2 0.0.0.0/0         tcp dpt:22
3)ACCEPT    tcp   --    x.x.x.x     0.0.0.0/0         tcp dpt:22
```

L'ordine delle regole è stato evidentemente inserito in modo sbagliato! Infatti dato che ciascuna di esse viene analizzata sequenzialmente, la numero "**1**" prevarrà su tutte le altre. In pratica per ogni pacchetto in ingresso nella vostra linux box, la regola "`1`" verrà analizzata prima della "`2`" e della "`3`". Ciò non permetterà agli indirizzi IP `192.168.0.2` e `x.x.x.x` di collegarsi attraverso ssh perché ogni tentativo di connessione alla porta 22 verrà bloccato.

L'errore nella dichiarazione delle ultime due regole consiste nell'utilizzo dell'opzione "`-A`". Come già detto questa opzione aggiunge in coda (in gergo "appende") la regola specificata a quelle eventualmente già esistenti. Il problema può essere bypassato rimuovendole nel seguente modo (notate l'utilizzo dell'opzione "`-D`"):

```
# iptables –D INPUT –p tcp –s 192.168.0.2 –-dport 22 –j ACCEPT
# iptables –D INPUT –p tcp –s x.x.x.x –-dport 22 –j ACCEPT
```

e ridigitandole in quest'altro:

```
# iptables –I INPUT –p tcp –s 192.168.0.2 –-dport 22 –j ACCEPT
# iptables –I INPUT –p tcp –s x.x.x.x –-dport 22 –j ACCEPT
```

L'opzione "`-I`" in questo caso mette ciascuna regola specificata in testa a quelle già presenti all'interno della chain "`INPUT`", pur essendo entrambi state inserite successivamente. Provate a collegarvi al servizio ssh adesso da uno degli host abilitati e non noterete alcun problema.

Analizziamo quindi nuovamente la tabella delle regole per la chain "`INPUT`":

```
Chain INPUT (policy ACCEPT)
Target      prot  opt   source      destination
ACCEPT      tcp   --    x.x.x.x     0.0.0.0/0         tcp dpt:22
ACCEPT      tcp   --    192.168.0.2 0.0.0.0/0         tcp dpt:22
DROP        tcp   --    0.0.0.0/0   0.0.0.0/0         tcp dpt:22
```

L'ordine è ora corretto. Ogni tentativo di connessione al servizio **ssh** proveniente dagli indirizzi IP `x.x.x.x` ed `192.168.0.2` verrà fatto transitare. Qualsiasi altro tentativo da un host diverso verso la stessa porta verrà invece bloccato.

1.6 Loggare le connessioni

Bloccare le connessioni indesiderate o l'ingresso di determinato traffico sarebbe un'attività fine a sé stessa se non si avesse la possibilità di verificare "**chi**" ha fatto "**cosa**" in un determinato momento. Fortunatamente iptables è in grado di loggare tutto ciò che si vuole con l'aggiunta di una semplice regola. Anche in questa circostanza comunque bisogna rispettare la propedeuticità.

Vale in ogni caso quanto segue: la regola di "`LOG`" deve sempre precedere la regola di "`DROP`" ed il suo contenuto deve essere identico a quest'ultima.

Per esempio vediamo come loggare su file tutte le connessioni bloccate sulla porta 22. Dapprima svuotate per intero la tabella delle regole:

```
# iptables –F
```

Quindi digitate la regola:

```
# iptables –A INPUT –p tcp –s 0/0 –-dport 22 –j DROP
```

A questo punto facciamo precedere ad essa la regola di "LOG"

```
# iptables –I INPUT –p tcp –s 0/0 –-dport 22 –j LOG –-log-prefix "Connessione
Bloccata SSH: "
```

ed analizziamo il contenuto della tabella delle regole::

```
# iptables –L INPUT -n
chain INPUT (policy ACCEPT)
Target      prot  opt   source      destination
LOG         tcp   --    0.0.0.0/0   0.0.0.0/0         --log-prefix "Connessione
Bloccata SSH: "
DROP        tcp   --    0.0.0.0/0   0.0.0.0/0         tcp dpt:22
```

I tentativi di accesso al servizio SSH vengono solitamente loggati all'interno del file `/var/log/messages`. Questo può variare in base alle impostazioni della distribuzione Linux utilizzata. Provate a simulare uno scenario in cui la connessione viene bloccata. Prima però, da un prompt del vostro firewall, digitate:

```
# tail –f /var/log/messages
```

quindi da un PC diverso avviate una sessione ssh:

```
# ssh 192.168.0.4
```

o nel caso in cui aveste la possibilità di simulare l'esempio da un host esterno digitate:

```
# ssh ip_pubblico_firewall
```

A questo punto nella shell in cui è stato lanciato il comando "`tail`" sarà possibile osservare le connessioni bloccate da iptables ed identificate dalla stringa di testo "*Connessione Bloccata SSH: "*

```
Sep 16 11:06:15 localhost kernel: Connessione Bloccata: IN=eth0 OUT=
MAC=00:30:84:40:9b:ec:00:30:84:40:9e:91:08:00 SRC=192.168.0.18 DST=192.168.0.4
LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=16679 DF PROTO=TCP SPT=2402 DPT=22
WINDOW=65535 RES=0x00 SYN URGP=0

[...]
```

Come si può ben notare, la direttiva "`--log-prefix`" specificata nella regola di "`LOG`" influenza il contenuto del file `/var/log/messages`. Potete inserire una frase o un termine diverso per ogni regola di "`LOG`" che desiderate, magari per rendere più semplice la lettura del file di log stesso Prediligete comunque l'utilizzo di parole corte come ad esempio *intruso*, *conn proxy*, *smtp*, etc.. Ne guadagnerete in salute in fase di analisi e verifica (fatelo ogni tanto un controllo sui log ;)

1.7 Proteggersi dalle tecniche di scanning comuni

Comprese le basi ed il funzionamento di iptables, vediamo ora come ci si può proteggere dai continui tentativi di accesso non autorizzato a cui si è sottoposti quando si è collegati ad Internet. Prima di procedere però azzerate la tabella delle regole:

```
# iptables –F
```

Una delle preoccupazioni principali quando si configura un firewall, dovrebbe essere quella di far sì che il proprio sistema sia il meno rintracciabile possibile, ovvero fare in modo che reagisca in modo da rilasciare il minor numero di informazioni quando sottoposto ad un attacco. Due tra le tecniche di scanning oggi maggiormente utilizzate sono il "*Vanilla*" ed il "*Syn Scanning*". Per proteggersi da entrambi potete digitare da linea di comando:

```
# iptables –I INPUT –p tcp –i ppp0 –s 0/0 –-dport 1:65535 –-syn –j DROP
```

Osserviamo in dettaglio le opzioni ed i parametri che ancora non si conoscono:

- l'opzione "**`--dport`**" indica la porta di destinazione della regola. In questo caso è stato specificato un range che va da "**`1`**" a "**`65535`**". Questa notazione sta ad indicare "*qualsiasi porta*". E' stata utilizzata solo per introdurre la possibilità di dichiarare un range di porte desiderato purché separato dal simbolo "**`:`**". Lo stesso effetto si sarebbe ottenuto omettendo l'intera opzione ed il parametro ad essa passato (ricordate che di default, quando si omette un'opzione, iptables assegna il valore "qualsiasi").

- l'opzione "**`--syn`**" indica il flag TCP che si vuole controllare per ogni pacchetto in entrata. I pacchetti con flag *syn* attivo vengono inizialmente scambiati per stabilire una connessione tra due host (cosiddetto **preludio al collegamento**). Rappresentano la fase antecedente allo scambio di dati vero e proprio.

La regola non fa altro che bloccare (`-j DROP`) qualsiasi pacchetto tcp (`-p tcp`) proveniente da qualsiasi indirizzo IP (`-s 0/0`) che tenti di iniziare una connessione (`--syn`) verso qualunque porta (`--dport 1:65535`) in ascolto presso l'interfaccia esterna della vostra linux box (`-i ppp0`). In parole povere questa regola blocca tutte le connessioni in entrata!

E' possibile testarla anteponendo ad essa una regola di log nel seguente modo:

```
# iptables –I INPUT –p tcp –i ppp0 –s 0/0 –-dport 1:65535 –-syn –j LOG –-log-
prefix "Conn block input: "
```

Da questo momento in poi ogni tentativo di collegamento al firewall perpetrato dall'esterno verrà loggato su `/var/log/messages`" (o file alternativo):

```
Sep 16 12:16:15 localhost kernel: Conn block input: IN=ppp0 OUT=
MAC=00:30:84:40:9b:ec:00:30:84:40:9e:91:08:00 SRC=x.x.x.x DST=
ip_pubblico_firewall LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=13100 DF PROTO=TCP
SPT=32109 DPT=25 WINDOW=8192 RES=0x00 SYN URGP=0
[...]
```

Naturalmente potete cambiare l'interfaccia `ppp0` con quella che corrisponde alla vostra interfaccia esterna o interna (ad esempio `eth0`, `wlan0`, etc…) sulla quale intendete operare.

1.8 Gli stati

La regola scritta in precedenza viene detta **stateless**, ovvero non considera gli stati di una connessione TCP. Essa si basa unicamente su un semplice meccanismo di *packet inspection*.

Iptables comunque può memorizzare gli ***stati*** delle connessioni. Attraverso una funzionalità detta "*connection tracking*" è in grado di capire in quale contesto può essere collocato un qualsiasi pacchetto in ingresso o in uscita. Il vantaggio è di scrivere regole più robuste e meno soggette alla possibilità di essere bypassare da un attacker.

Iptables gestisce gli stati mediante quattro espressioni diverse:

- **`NEW:`** si riferisce ad un pacchetto che sta iniziando una nuova connessione;
- **`INVALID:`** si riferisce ad un pacchetto che non è associato ad alcuna connessione esistente;
- **`ESTABLISHED:`** si riferisce ad un pacchetto relativo ad una connessione già stabilita.
- **`RELATED:`** si riferisce ad un pacchetto che vuole iniziare una nuova connessione ma che a sua volta è associato ad una sessione esistente (ad esempio ftp utilizza due canali diversi: la porta 21 per l'invio di comandi ed un canale alternativo per lo scambio vero e proprio dei dati).

Utilizzando quanto appreso, riscriviamo la regola precedente usufruendo del supporto agli stati offerto dalla funzionalità di **connection tracking**:

```
# iptables -I INPUT -p tcp –i ppp0 -m state -s 0/0 --dport 1:65535 --state NEW -
j DROP
```

L'opzione "`-m`" permette l'aggiunta di nuovi moduli che danno la possibilità all'utente di estendere le caratteristiche e le opzioni base di iptables. In questo caso abbiamo attivato il modulo "`state`" che permette di effettuare dei controlli sullo stato dei pacchetti intercettati. Infatti attraverso la direttiva "`--state NEW`" si istruisce il firewall a bloccare (`-j DROP`) tutti i pacchetti nello stato "`NEW`", vale a dire tutti quelli che cercano di iniziare una nuova connessione. Gli effetti finali prodotti sono praticamente identici a quelli della prima regola vista nel paragrafo precedente.

Potete testarne il funzionamento con nmap, digitando da un host esterno:

```
# nmap –sS –n –p 1-65535 IP_pubblico_firewall
```

Al termine dell'attività, se l'output ottenuto sarà:

`All 65535 scanned ports on` *`IP_pubblico_firewall`* `are: filtered`"

Potrete affermare di aver reso il vostro sistema irraggiungibile ai tentativi di Syn e Vanilla Scanning provenienti dall'esterno.

Nota: Se lo si desidera tutti gli esempi fin qui mostrati possono essere adeguati ad un utilizzo LAN oriented, cambiando "`-i ppp0`" in "`-i eth0`" (o qualsiasi altra interfaccia desiderata) e sostituendo nelle sessioni nmap o telnet d'interesse *proprio_IP_pubblico* con l'indirizzo IP locale del sistema.

1.9 Conclusione primo giorno

In realtà le regole antiscanning mostrate sin qui non vanno bene per qualsiasi caso: sono insufficienti contro altre tecniche più avanzate come il FIN Scanning o l'ACK probing. Ciò è facilmente dimostrabile avviando, da un host esterno, una sessione di nmap diversa:

```
# nmap –sF –n IP_pubblico_firewall
```

```
PORT        STATE SERVICE
22/tcp      open  ssh
6000/tcp    open  X11
[…]
```

Il risultato è lapalissiano: vengono rilevate delle porte aperte. Siete cioè ancora rintracciabili da Internet! Perché? Per adesso riposate il cervello. Domani si vedrà come colmare questa lacuna.

1.10 Appendice A

Se non disponete di un sistema esterno per effettuare le prove di scansione proposte nel tutorial e desiderate sapere se il vostro firewall risulta ben configurato ai probing che arrivano da Internet, potete usufruire di uno dei servizi gratuiti offerti in rete come **http://scan.sygate.com**.

2 Introduzione: Giorno 2

Ieri abbiamo dimostrato come la regola:

"**iptables –I INPUT –p tcp –i ppp0 –m state –s 0/0 –-dport 1:65535 –-state NEW –j DROP**"

fosse insufficiente a garantire la sicurezza di un firewall contro tecniche di scanning avanzate (*FIN Scanning*, *ACK probing*, etc...) provenienti da Internet. In realtà per ergere un muro difensivo più efficace bastano davvero pochi accorgimenti. Vediamo come!

2.1 Filtrare il protocollo tcp: revisione della regola di ieri

Dapprima svuotate la tabella delle regole di iptables per non fare confusione con quelle già presenti:

```
# iptables –F
```

…quindi digitate la nuova:

```
# iptables -I INPUT -p tcp –i ppp0 -m state -s 0/0 --dport 1:65535 --state
INVALID,NEW -j DROP
```

La prima cosa che si può notare è la sostituzione dell'estensione "`--state NEW`" in "`--state INVALID,NEW`". Lo "`--state NEW`" della regola precedente corrispondeva di fatto all'opzione "`--syn`" vista nelle scorse pagine e tendeva a fare match con tutti i pacchetti che avessero il flag TCP SYN attivo. Utilizzando invece la seconda estensione "`--state INVALID,NEW`" si indica ad iptables di bloccare (`-j DROP`) tutti i pacchetti il cui stato della connessione sia "`INVALID`" o "`NEW`", il che applicato alla nuova regola e tradotto in parole semplici significa che:

- Con "**`INVALID`**" vengono bloccati tutti quei pacchetti che non sono associati ad alcuna connessione esistente;
- Con "**`NEW`**" vengono bloccati tutti quei pacchetti che tentano di inizializzare una nuova connessione;

Questa regola offre contemporaneamente due benefici. Il primo consiste nel fatto che il firewall diventa contemporaneamente immune a diversi tipi di scanning. Potete ad esempio avviare vari test con nmap da un altro PC posto all'esterno della vostra rete:

```
# nmap –sF –n –p 1-65535 ip_pubblico_firewall
# nmap –sS –n –p 1-65535 ip_pubblico_firewall
# nmap –sA –n –p 1-65535 ip_pubblico_firewall
```

La prima scansione rileverà 65535 porte aperte. Ciò accade perché il FIN Scanning si distingue dalle comuni tecniche di scansione. Tale output viene infatti generato proprio quando si è in presenza di un sistema che blocca tutti i pacchetti relativi ad un attacco di questo genere. Ciò significa in realtà che la regola è stata configurata in modo corretto. A conclusione delle ultime due scansioni (SYN Scanning ed ACK Scanning) verrà invece riportato a video il seguente messaggio: "`All 65535 scanned ports on 192.168.0.4 are: filtered`". Tutte le porte risultano cioè irraggiungibili ad host esterni. Non c'è che dire, un gran bel risultato!

Il secondo beneficio consiste nel fatto che il traffico uscente dalla Linux box non subirà comunque alcuna interruzione. Nessuna delle connessioni che effettuerete verso Internet verranno bloccate (la funzionalità di **Connection Tracking** fa si che iptables comprenda che i pacchetti in ritorno, legati ad una vostra esplicita richiesta di connessione, si riferiscano in realtà ad una sessione lecita e non ad un tentativo di collegamento esterno non autorizzato).

A dimostrazione di ciò potete effettuare delle prove di navigazione con il vostro browser web preferito o, per essere più sbrigativi, digitare da linea di comando:

```
# telnet www.google.it 80
Trying 66.102.9.99...
Connected to www.google.com.
Escape character is '^]'. <--
```

questo fa capire che la connessione è davvero avvenuta con successo (il server attende l'immissione di comandi). La regola inserita non influenza quindi in alcun modo il traffico da voi generato dall'interno verso l'esterno.

2.2 Filtrare il protocollo icmp

Pur l'efficacia dell'ultima regola inserita, siete ancora rintracciabili con un semplice ping da un qualsiasi pc al di fuori della vostra LAN:

```
# ping ip_pubblico_firewall –c 2
PING 192.168.0.4 (192.168.0.4) 56(84) bytes of data
64 bytes from 192.168.0.4: icmp_seq=1 ttl=64 time=0.194 ms
64 bytes from 192.168.0.4: icmp_seq=2 ttl=64 time=0.216 ms
```

o con un pacchetto ICMP TimeStamp sollecitabile da nmap attraverso l'opzione "`-PP`":

```
# nmap –PP -n ip_pubblico_firewall
[...]
Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

Il vantaggio di non rispondere né ai ping né a qualsiasi altro pacchetto icmp (al di fuori di rare eccezioni) è una pratica che aiuta certamente ad essere meno rintracciabili ed identificabili in rete. Un'ottima regola da applicare in tal senso potrebbe essere:

```
# iptables -I INPUT -p icmp –i ppp0 -m state -s 0/0 --state INVALID,NEW -j DROP
```

L'unica cosa di nuovo qui è il parametro passato all'opzione "`-p`" (vale a dire "`icmp`") che indica appunto a quale protocollo fa riferimento la regola. Questa non fa altro che bloccare tutti quei pacchetti icmp che cercano di raggiungere la vostra linux box senza essere stati richiesti o senza essere legati in qualche modo ad una sessione da voi inizializzata.

Naturalmente, come per il caso precedente, avrete la possibilità di generare in uscita tutto il traffico icmp che desiderate. Anche i messaggi ICMP di errore relativi ad un collegamento da voi stabilito verranno fatti passare senza problemi. Come prova potete lanciare un ping dal firewall ed osservare la risposta ottenuta:

```
# ping www.google.it –c 4
PING www.google.it (66.102.9.99) 56(84) bytes of data
64 bytes from 66.102.9.99: icmp_seq=1 ttl=64 time=0.694 ms
64 bytes from 66.102.9.99: icmp_seq=2 ttl=64 time=0.416 ms
64 bytes from 66.102.9.99: icmp_seq=1 ttl=64 time=0.394 ms
```

```
64 bytes from 66.102.9.99: icmp_seq=2 ttl=64 time=0.716 ms
```

A questo punto della discussione, la tabella delle regole per la chain `INPUT` dovrebbe mostrare le seguenti entry:

```
# iptables –L INPUT
Chain INPUT (policy ACCEPT)
target      prot  opt   source      destination
DROP        icmp  --    anywhere    proprio_ip_pubblico     state INVALID,NEW
DROP        tcp   --    anywhere    proprio_ip_pubblico     state INVALID,NEW tcp
dpts:1:65535
```

La presenza di queste due semplici regole inibisce a tal punto le capacità offensive di un attacker che la maggior parte degli strumenti di sicurezza da lui utilizzati come nmap, si rifiuteranno di scansionare il vostro firewall se non forzati con un'apposita opzione:

```
# nmap –sS –n ip_pubblico_firewall

[...]
Note: Host seems down. If it is really up, but blocking our ping probes, try -P0
Nmap run completed -- 1 IP address (0 hosts up) scanned in 12.109 seconds
```

Notate la frase "*Host seems down*". Dopo i primi probe, nmap non è in grado di determinare se l'host è effettivamente in vita e pertanto si arrende. L'unico modo per farlo proseguire nella scansione (che sarebbe comunque ugualmente infruttuosa) consiste nel forzarlo con l'opzione "`-P0`".

2.3 Filtrare il protocollo udp

Pur gli accorgimenti intrapresi, il firewall non è ancora sufficientemente protetto rispetto a tutte quelle tecniche che si basano sull'ottenimento di informazioni attraverso lo sfruttamento del protocollo udp. Seppure sia molto improbabile che un utente "*casalingo*" offra al mondo esterno servizi che si basino su porte udp, è buona norma comunque filtrare in ingresso questo protocollo. Per l'occasione potete utilizzare la seguente regola:

```
# iptables -I INPUT -p udp –i ppp0 -m state -s 0/0 --state INVALID,NEW -j DROP
```

I benefici sono gli stessi già ampiamente discussi nei precedenti casi. Sarete al solito liberi di inviare pacchetti udp dalla vostra linux box verso l'esterno, senza rischiare che vengano bloccati.. Nel seguente esempio si dimostra come sia possibile continuare normalmente a dialogare con un server dns:

```
# nslookup www.altavista.com
Server:         xxx.xxx.xxx.xxx
Address:        xxx.xxx.xxx.xxx#53

www.altavista.com       canonical name = avatw.search.yahoo2.akadns.net.
Name:   avatw.search.yahoo2.akadns.net
Address: 216.155.200.155
```

Allo stesso tempo eviterete però che il firewall risponda a probe UDP perpetrati dall'esterno:

```
# nmap -sU ip_pubblico_firewall -n -p 10-20
```

```
Interesting ports on  (ip_pubblico_firewall):
Port       State       Service
10/udp     open        unknown
11/udp     open        systat
12/udp     open        unknown
13/udp     open        daytime
14/udp     open        unknown
15/udp     open        unknown
16/udp     open        unknown
17/udp     open        qotd
18/udp     open        msp
19/udp     open        chargen
20/udp     open        ftp-data
```

Le tecniche di scansione/probing che si basano su protocollo UDP, differiscono nella forma e nel risultato finale da quelle classiche TCP (un po' come accadeva con il FIN Scanning). L'output appena riportato, in cui si evince che tutte le porte udp analizzate sono nello stato "open" (in realtà sono tutte chiuse), permette di comprendere che la regola iptables è stata configurata correttamente. Qualsiasi porta analizzata verrà in pratica indicata come aperta, non offrendo all'attacker alcuno spunto per determinare informazioni utili alla sua attività intrusiva.

2.4 Creare regole meno restrittive

Dopo l'inserimento dell'ultima regola, la chain `INPUT` conterrà le seguenti entry:

```
# iptables –L INPUT
Chain INPUT (policy ACCEPT)
target      prot  opt   source      destination
1) DROP     udp   --    anywhere    proprio_ip_pubblico     state INVALID,NEW
2) DROP     icmp  --    anywhere    proprio_ip_pubblico     state INVALID,NEW
3) DROP     tcp   --    anywhere    proprio_ip_pubblico     state INVALID,NEW tcp
dpts:1:65535
```

I benefici di queste tre semplici regole (numerate per praticità) si estendono anche all'immunità da pratiche di **Fingerprint** oltre che, come già visto, alle più svariate tecniche di scanning. Per esempio, l'esecuzione del seguente comando da un PC esterno alla LAN:

```
# nmap –O –n ip_pubblico_firewall
```

non fornirà alcuna informazione sul sistema operativo della vostra postazione Linux.

Queste semplici tre regole rappresentano **la base di qualsiasi Firewall** ed in quanto tali vanno adattate alle proprie esigenze. Potrebbe essere utile, tanto per citare un caso, permettere a due indirizzi IP esterni ("`y.y.y.y`" e "`z.z.z.z`") di pingarvi. Affinché le nuove regole che intendete aggiungere funzionino a dovere, dovrete collocarle nella chain `INPUT` prima della numero "`2`", altrimenti questa prevarrà su tutte le altre (il rispetto della propedeuticità delle regole è stato già trattato). Per farlo sarà sufficiente digitare:

```
# iptables -I INPUT -p icmp –i ppp0 -s y.y.y.y --icmp-type echo-request -j
ACCEPT

# iptables -I INPUT -p icmp –i ppp0 -s z.z.z.z --icmp-type echo-request -j
ACCEPT
```

L’opzione “`--icmp-type`” offre la possibilità di specificare il tipo di pacchetto icmp su cui si intende operare. In questo caso il parametro “`echo-request`” corrisponde alla richiesta di ping. Una lista di parametri utilizzabili è consultabile digitando il comando:

```
# iptables -p icmp –h
```

Un’altra necessità che potrebbe sorgere, potrebbe essere quella di consentire a chiunque l’accesso incondizionato a certi servizi. Ad esempio si potrebbe voler rendere accessibili all’esterno i servizi di posta elettronica (porte TCP **25** e **110**), l’ftp pubblico (porta TCP **21**) ed il server web (porta TCP **80**). In questo caso tutte le regole andranno naturalmente inserite prima della numero “`3`”:

```
# iptables –I INPUT –p tcp –i ppp0 –s 0/0 –-dport 21 –j ACCEPT
# iptables –I INPUT –p tcp –i ppp0 –s 0/0 –-dport 25 –j ACCEPT
# iptables –I INPUT –p tcp –i ppp0 –s 0/0 –-dport 80 –j ACCEPT
# iptables –I INPUT –p tcp –i ppp0 –s 0/0 –-dport 110 –j ACCEPT
```

Potete comunque ottimizzare la regola e le performance di iptables, combinandole tutte in un’unica riga:

```
# iptables -I INPUT -p tcp –i ppp0 -m multiport -s 0/0 --dports 21,25,80,110 -j
ACCEPT
```

L’estensione “`-m multiport`” consente di aggiungere il supporto multiporta, attraverso il quale specificare, con l’opzione “`--dports`”, un massimo di 15 porte separate da virgola.

2.5 Conclusione secondo giorno

Anche questo secondo giorno si è avviato a conclusione. Riposate il corpo e la mente per la terza ed ultima sessione del corso che sarà molto più impegnativa!

3 Introduzione: Giorno 3

Evitiamo inutili discussioni e passiamo subito agli argomenti di questo ultimo giorno di corso.

3.1 Condividere il collegamento ad Internet

Se si dispone di più di un PC collegato alla LAN, è possibile configurare la macchina Linux che funge da firewall in modo da condividere il collegamento ad Internet anche in presenza di una sola linea. La tecnica che permette di farlo prende il nome di *masquerading* o più semplicemente **NAT** (per saperne di più vedere l'appendice B). Come l'appellativo stesso dice, non fa altro che "**mascherare**" gli indirizzi IP privati della LAN in modo da stabilire un canale di comunicazione con l'esterno. Gli host esterni vedranno solo l'indirizzo IP pubblico assegnato dal provider, ma i sistemi interni saranno in grado di accedere ad Internet. Come è possibile?

Per ogni collegamento che parte dalla rete interna verso l'esterno, il firewall tiene traccia del flusso dei pacchetti transitanti. Per ciascuna richiesta crea una mappa logica in base alle associazioni indirizzo sorgente/porta sorgete, indirizzo di destinazione/porta di destinazione, che gli permette determinare a quale host della LAN devono essere smistati i dati ritornati.

Iptables consente l'implementazione di una configurazione di questo tipo attraverso il target "`MASQUERADE`" che è consentito solamente nella chain "`POSTROUTING`" della tabella "`nat`". Ad esempio la seguente regola:

```
# iptables –A POSTROUTING –t nat –s 192.168.0.0/24 -j MASQUERADE
```

consente di applicare il meccanismo del *maquerading* a tutti i PC della rete con range di IP che va da "`192.168.0.1`" ad "`192.168.1.254`". Dopo l'inserimento di questa regola il contenuto della chain "`POSTROUTING`" nella tabella "`nat`" sarà il seguente:

```
# iptables -L POSTROUTING -t nat -n

Chain POSTROUTING (policy ACCEPT)
target      prot opt source               destination
MASQUERADE  all  --  192.168.0.0/24       0.0.0.0/0
```

Ma non è finita qui. Per poter garantire l'accesso ad internet ai client della LAN, il firewall deve comportarsi come un router. Niente di più semplice. Su Linux è sufficiente apportare una piccola modifica al filesystem "`/proc`". Al volo potete attivare questa impostazione digitando dal prompt dei comandi del firewall:

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

Potete renderla stabile dopo il reboot aggiungendo la direttiva "`net.ipv4.ip_forward = 1`" nel file di configurazione "`/etc/sysctl.conf`". Alternativamente potete aggiungere "`echo 1 > /proc/sys/net/ipv4/ip_forward`" in uno degli script di caricamento del sistema (ad esempio `/etc/rc.d/rc.local`").

Da questo momento in poi tutti i sistemi della LAN che avranno impostato come gateway nelle configurazioni di rete l'indirizzo IP locale del firewall, saranno in grado di accedere ad Internet.

3.2 Gestire i client della LAN

Se scegliete di nattare gli indirizzi IP privati, la sicurezza del firewall assume una valenza superiore in quanto il suo scopo non è più quello di proteggere solamente sé stesso ma anche i client della rete interna. Ma come minimizzare il rischio di attacchi informatici? Pensare di attuare una politica restrittiva dei collegamenti in ingresso ed una, viceversa, di larghe maniche per quelli in uscita, è un grosso errore. Garantire la sicurezza dei client non significa solo accertarsi che non possano essere attaccati "frontalmente". Gran parte del vostro lavoro di un amministratore di sistema comporterà anche una mediazione dei servizi esterni che i client potranno contattare.

Naturalmente è necessario conoscere con esattezza quali porte devono essere autorizzate, altrimenti si corre il rischio di bloccare applicazioni che negli intenti originari non dovevano essere impattate. Ad esempio si prendano in considerazione queste tre regole:

```
1) # iptables -t nat -A PREROUTING -p tcp -m multiport -s 192.168.0.0/24 –-
dports 20,21,22,25,53,80,110,443 -j ACCEPT

2) # iptables -t nat -A PREROUTING -p udp -s 192.168.0.0/24 --dport 53 -j ACCEPT

3) # iptables -t nat -A PREROUTING -s 192.168.0.0/24 -j DROP
```

Nella prima si specifica che tutti gli indirizzi ip nel range 192.168.0.0/24 (`-s 192.168.0.0/24`) possono collegarsi (`-j ACCEPT`) alle porte tcp (`-p tcp`):

- `20,21,22,25,53,80,110,443`

di qualsiasi indirizzo IP di destinazione.

In questo modo ai client viene offerta la possibilità di contattare i servizi remoti:

- `ftp-data, ftp, ssh, smtp, dns, http, pop3, https`

Con questa regola si è usufruito dell'estensione multiport (`-m multiport`) e nello specifico dell'opzione "`dports`" (`–-dports 20,21,22,25,53,80,110,443`) per evitare di scrivere otto regole separate. La chain utilizzata inoltre è "`PREROUTING`". Essa consente di analizzare i pacchetti in ingresso nella linux box prima che vengano prese decisioni riguardo al loro smistamento (routing).

Nella seconda regola si indica che tutti gli indirizzi ip nel range 192.168.0.0/24 (`-s 192.168.0.0/24`) hanno possibilità di collegarsi (`-j ACCEPT`) alla porta udp 53 (`-p udp`) di qualsiasi indirizzo di destinazione. In questo modo si garantisce l'accesso ai servizi di risoluzione DNS, fondamentali per consentire ai client di navigare ed interagire correttamente con Internet. Anche qui il controllo viene operato in "`PREROUTING`".

Nel terza regola si specifica infine di bloccare (`-j DROP`) qualsiasi connessione verso qualsiasi destinazione contattata dal range di indirizzi IP 192.168.0.0/24 (`-s 192.168.0.0/24`). Quest'ultima regola chiude il cerchio.

Ricapitolando: i client interni possono solamente contattare i servizi **tcp** "`ftp-data, ftp, ssh, smtp, dns, http, pop3, https`" ed il servizio **udp** "`dns`". Qualsiasi tentativo di contattare qualunque altro servizio verrà bloccato dal firewall. Ciò è sufficiente, nella stragrande maggioranza

dei casi, a garantire ai PC della propria **intranet**, una certa autonomia quando navigano in **Internet**. Naturalmente si è liberi di aggiungere o rimuovere le porte relative ai servizi desiderati.

3.3 Port Forwarding

Utilizzando la tecnica del masquerading, il firewall diviene l'unico punto di accesso verso la LAN. Talvolta si può avere la necessità di garantire dall'esterno, la possibilità di collegarsi ai servizi ospitati da un sistema interno. Grazie al target "`DNAT`". iptables offre la possibilità di fare ***port forwarding***.

Ad esempio se si volesse rendere visibile da Internet un server web posto nella LAN (supponiamo che l'indirizzo IP sia "`192.168.0.2`") basterebbe aggiungere la seguente regola:

```
# iptables -t nat -I PREROUTING -p tcp -d ip_pubblico_firewall --dport 80 -j
DNAT --to-destination "192.168.0.2:80"
```

Oppure senza la necessità di specificare l'indirizzo IP pubblico del firewall ma solo l'interfaccia:

```
# iptables -t nat -I PREROUTING -p tcp –i ppp0 --dport 80 -j DNAT --to-
destination "192.168.0.2:80"
```

L'opzione "`--to-destination`" è la più importante: punta al server interno ed alla relativa porta alla quale la connessione andrà dirottata. In entrambi i casi presentati, quando un host esterno contatta la porta 80 dell'indirizzo IP pubblico del firewall, la connessione viene "rigirata" alla porta "`80`" del server interno "`192.168.0.2`". Vediamo un altro esempio:

```
# iptables -t nat -I PREROUTING -p tcp –s y.y.y.y -d proprio_ip_pubblico --dport
8080 -j DNAT --to-destination "192.168.0.2:443"
```

In questo caso solo l'host esterno "`y.y.y.y`" può contattare la porta "`8080`" dell'ip pubblico del firewall. Quando ciò accade la connessione viene dirottata verso la porta "`443`" del server interno "`192.168.0.2`".

3.4 FTP e NAT

Quando il firewall è configurato in modalità **masquerading** (meglio nota come NAT), i client della LAN potrebbero manifestare problemi nei collegamenti ftp remoti (soprattutto in modalità passiva). Questi sono dovuti alla natura stessa del protocollo FTP che opera su più porte, anche allocate dinamicamente. Per risolvere eventuali grane che dovessero manifestarsi, è sufficiente aggiuntare un paio di moduli kernel digitando dal prompt dei comandi del firewall:

```
# modprobe ip_conntrack_ftp
# modprobe ip_nat_ftp
```

Ogni problema dovrebbe ora cessare di manifestarsi.

3.5 Qualche notizia in più su mangle

"`mangle`" è una tabella speciale utilizzata per l'alterazione delle opzioni dei pacchetti che transitano nelle chain "**`OUTPUT`**" e "**`PREROUTING`**". A partire dal kernel 2.4.18 e superiori, è possibile utilizzare questa tabella anche in congiunzione con le chain "**`FORWARD`**", "**`INPUT`**" e "**`POSTROUTING`**". Seppur la sua utilità possa sembrare idonea solamente in fase di scrittura di regole iptables avanzate, una forma di "**mangle**" automatica è praticata in modo totalmente trasparente all'utente attraverso l'alterazione di indirizzi IP e porte, utilizzando ad esempio target come "`MASQUERADE`" o "`DNAT`".

3.6 Conclusione

Il corso è finalmente giunto a conclusione. Per comodità riportiamo nella Appendice C, un piccolo script che raccoglie la maggior parte delle regole fin qui presentate. A voi viene lasciato naturalmente il compito adattarle secondo esigenza.

3.7 Appendice B: Come funziona NAT

NAT sta per **Network Address Translation** ed è principalmente un meccanismo di mascheramento per reti che usufruiscono di indirizzi IP privati (ad esempio `192.168.0.1`, `172.16.1.1`, `10.0.0.1`, etc..) non visibili esternamente. Cosa accade realmente quando iptables viene configurato in modo da operare il masquerading di questi indirizzi? Per comprenderlo avvaliamoci di una sessione iniziale di collegamento intercettata con tcpdump:

```
1) 13:56:35.188403 IP 192.168.0.254.60465 > 66.249.85.104.80: S
1909238142:1909238142(0) win 5840 <mss 1460,sackOK,timestamp 321351154
0,nop,wscale 0>
```

Il firewall riceve un pacchetto da uno dei client della LAN (**`192.168.0.254`**) destinato all'indirizzo **`66.249.85.104`** (*www.google.com*). Il pacchetto reca con sé il flag tcp SYN ("**`S`**") il che indica una richiesta di collegamento.

```
2) 13:56:35.188454 IP 62.121.13.119.60465 > 66.249.85.104.80: S
1909238142:1909238142(0) win 5840 <mss 1460,sackOK,timestamp 321351154
0,nop,wscale 0>
```

Se il firewall lasciasse inalterato l'IP sorgente della richieste, non sarebbe possibile per il client interno comunicare correttamente con la destinazione. Per questo motivo l'indirizzo viene modificato con quello dell'interfaccia pubblica che si affaccia ad Internet (nell'esempio **`62.121.13.119`**). Tutti gli altri campi (porte, opzioni e valori del pacchetto) vengono lasciati inalterati. Da questo momento in poi il firewall memorizza in una specie di tabella i dati relativi alla connessione **nattata**.

```
3) 13:56:35.311652 IP 66.249.85.104.80 > 62.121.13.119.60465: S
2813525196:2813525196(0) ack 1909238143 win 8190 <mss 1460>
```

A questo punto l'host di destinazione replica al pacchetto SYN iniziale. La risposta viene ricevuta dall'interfaccia esterna del firewall (nell'esempio **`62.121.13.119`**). Analizzando la tabella dei pacchetti nattati, il firewall capisce che il pacchetto ricevuto è di competenza di uno dei client della LAN che protegge (**`192.168.0.254`**). E' stato lui infatti a richiedere l'inizializzazione della connessione. A questo punto l'indirizzo IP di destinazione del pacchetto viene sostituito con quello del client che ha fatto la richiesta:

4) 13:56:35.311673 IP **66.249.85.104.80** > **192.168.0.254.60465**: S 2813525196:2813525196(0) ack 1909238143 win 8190 <mss 1460>

Da questo momento in poi il firewall smisterà tutti i pacchetti relativi a questa specifica connessione, all'indirizzo IP interno `192.168.0.254`.

3.8 Appendice C: Firewall Script

```
# Interfaccia esterna del firewall. Modificare secondo la propria configurazione
EXT_INTERFACE="ppp0"

#
# Inserimento dei moduli di connection-tracking.
# Le seguenti righe vanno decommentate se i moduli non vengono caricati
# automaticamente dal sistema operativo. Ciò può essere verificato utilizzando
# il comando "lsmod".
#
# /sbin/modprobe ip_conntrack

# inserimento dei moduli per i collegamenti ftp dietro nat. Decommentare
# se il sistema pratica masquerading.
#
# /sbin/modprobe ip_conntrack_ftp
# /sbin/modprobe ip_ftp_nat

# Questa regola blocca tutto il traffico tcp non richiesto proveniente da
# Internet e destinato direttamente al firewall;
#
iptables -I INPUT -p tcp –i $EXT_INTERFACE -m state -s 0/0 --dport 1:65535 --
state INVALID,NEW -j DROP

# Questa regola blocca tutto il traffico icmp non richiesto proveniente da
# Internet e destinato direttamente al firewall;
#
iptables -I INPUT -p icmp –i $EXT_INTERFACE -m state -s 0/0 --state INVALID,NEW
-j DROP

# Questa regola blocca tutto il traffico udp non richiesto proveniente da
# Internet e destinato direttamente al firewall;
#
iptables -I INPUT -p udp –i $EXT_INTERFACE -m state -s 0/0 --state INVALID,NEW -
j DROP

# Questa regola attiva il masquerade delle connessioni. Decommentare e
# modificare opportunamente il range di IP se si desidera fornire la
# connettività Internet agli altri PC della LAN.
#
# iptables –A POSTROUTING –t nat –s 192.168.0.0/24 -j MASQUERADE

# Limitazioni dei servizi Internet ai client/server della rete interna.
# Decommentare e configurare opportunamente se si desidera imporre delle
# limitazioni all'operato dei client/server della LAN.
#
# iptables -t nat -A PREROUTING -p tcp -m multiport -s 192.168.0.0/24 –-dports
20,21,22,25,53,80,110,443 -j ACCEPT
# iptables -t nat -A PREROUTING -p udp -s 192.168.0.0/24 --dport 53 -j ACCEPT
# iptables -t nat -A PREROUTING -s 192.168.0.0/24 -j DROP
```

```
# Port Forwarding. Decommentare e configurare opportunamente la regola se
# si desidera praticare port forwarding.
#
# iptables -t nat -I PREROUTING -p tcp –s 0/0 –-dport porta_dest_fw –j DNAT --
to-destination "server_interno:porta"
```