
Università degli studi di Milano

Progetto d'esame per

Sistemi di elaborazione dell'informazione

# Firewall con IpTables

Marco Marconi

Anno Accademico 2009/2010

**Sommario**

Implementare un firewall con **iptables** e linux per inserirlo in una piccola realtà aziendale composta da una intranet, una DMZ, e la rete pubblica. Le problematiche incontrate in questo tipo di implementazione riguardano, il *MASQUERADE* tra la rete intranet e la rete internet, il nat per i servizi dalla rete pubblica alla rete DMZ, la gestione delle connessioni in modalità *statefull* tramite i moduli messi a disposizione da *iptables*. I paramentri del kernel per salvaguardare tutto il traffico, come ad esempio, icmp_echo_ignore_all, icmp_echo_ignore_broadcast, icmp_ignore_bogus_error_responses, ecc...., non verranno analizzati.

# Indice



# Elenco delle figure

# Elenco delle tabelle

# 1 Introduzione

Inseriamoci nella realtà di una piccola azienda che ha la necessità di pubblicare i propri servizi su internet e contestualmente dare ai propri dipendenti la possibilità di navigare sul web e utilizzare i servizi aziendali, che siano o meno pubblicati su internet. Uno dei primi problemi è la gestione dei pochi indirizzi ip pubblici che normalmente i provider assegnano alle piccole aziende. Nel momento in cui un'azienda decide di connettersi ad internet, non solo per la navigazione dei propri dipendenti, ma anche per la pubblicazione dei propri servizi, si trova davanti al problema della sicurezza della rete, che potrebbe essere compromessa da un potenziale attaccante posto su internet o anche sulla rete aziendale stessa. É quindi necessario dotare la propria rete dei servizi (DMZ) di una barriera (Firewall) con cui controllare e regolare il tipo di traffico che viene veicolato all'ingresso della rete DMZ. É altrettanto necessario proteggere la rete dei dipendenti da attacchi provenienti dalla rete internet.

Una possibile soluzione è quella di implementare un firewall con tre interfacce di rete, una per ogni rete logicamente diversa. Il nostro firewall, con una interfaccia sarà connesso ad internet, e sulle altre due rimaste libere, troveranno posto, la rete DMZ, e la rete dei dipendenti(intranet). Saranno utilizzati degli indirizzi privati, sia sulla DMZ che sulla intranet, e quindi, una ulteriore complessità alla configurazione del firewall, è costituita dall'utilizzo della funzionalità di *mascheratura*[1], che consente a tutti i dipendenti di *presentarsi* su internet utilizzando tutti lo stesso indirizzo ip pubblico.

In questo documento saranno analizzate ed implementate le tecniche di mascheratura, **MASQUERADE** e **NAT**[2], e la gestione in modalità statefull del firewall.

## 1.1 struttura del documento

Il documento è diviso in una parte introduttiva e a seguire dei cenni sulle funzionalità di IpTables che utilizzeremo sul nostro firewall (vedi sezione 2). Nella sezione 5 andremo ad implementare le nostre regole sul firewall per poi eseguire dei semplici test nella sezione 6.

[1] MASQUERADE come viene definita in iptables

[2] il NAT si divide anche per SNAT,DNAT,... in questo documento sarà utilizzata solamente la funzionalità di **D**estination**NAT**

# 2 Cenni di IpTables

## 2.1 premessa

Descrivere cos'è e come funziona IpTables, non è lo scopo di questo progetto, e quindi ci limiteremo a spiegare le sole funzionalità che andremo a utilizzare.

## 2.2 Il *connection tracking*

Dalla tabella 6 siamo in grado di capire quale traffico deve transitare per una certa interfaccia di rete, ma questo se sufficiente per la comprensione delle regole di un firewall generico, non lo è per chi deve implementare ogni singola regola su uno specifico firewall. Per fare un esempio banale, se provo a fare un ping da un host $A$ ad un host $B$ ed utilizzo una regola del tipo source:$A$ destination:$B$ protocol:*echo* rule:*accept* il ping continuerà a non funzionare in quanto non è stata specificata la regola per il ritorno del ping (echo reply). Il problema è che la risposta al ping, ma come tutte le connessioni di rete in risposta ad un'altra, fanno uso di numeri di porta casuali sopra la 1023. Aprire tutte le porte dopo la 1023, non è una buona politica di sicurezza, e per fortuna IpTables ci viene in aiuto con il modulo **connection tracking**, che fa diventare il firewall da stateless a stateful.

Il **connection tracking** è la capacità di mantenere in memoria le informazioni sulle connessioni in transito nel firewall. Con questa caratteristica IpTables è in grado di ricordare lo stato dei flussi e delle connessioni dei protocolli usati. In

```
/proc/net/ip_conntrack
```

il kernel mantiene e aggiorna in tempo reale lo stato delle connessioni gestite, fino ad un numero massimo che è possibile modificare intervenendo su

```
/proc/sys/net/ipv4/ip_conntrack_max
```

La gestione del traffico, basata sulle connessioni, si fa utilizzando il modulo di match (*-m state* con iptables) con **state** che permettere di suddividere il traffico nei seguenti stati:

- NEW - una nuova richiesta di connessione da parte di un client (SYN TCP o nuovo pacchetto UDP);
- ESTABLISHED - pacchetti relativi a connessioni già stabilite;
- RELATED - pacchetti correlati (RELATED) a connessioni esistenti e ESTABILSHED, come ad esempio il traffico ftp;
- INVALID - una connessione che non è nessuna delle precedenti.

Il *connection tracking* è particolarmente comodo per gestire il traffico di risposta per le connessioni già accettate. Ad esempio, per permettere tutto il traffico in uscita (tipico caso di un client), e in entrata solo il traffico RELATED a quanto uscito, è sufficiente una regola del tipo:

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

Per gestire lo stato di connessione di alcuni protocolli, esistono dei moduli speciali adibiti al connection tracking e al natting:

- ip_conntrack_ftp - ip_nat_ftp : Conntrack e nat per FTP
- ip_conntrack_h323 - ip_nat_h323 : Conntrack e nat per H323
- ip_conntrack_irc - ip_nat_irc : Conntrack e nat per IRC
- ip_conntrack_rtsp - ip_nat_rtsp : Conntrack e nat per IRC
- ip_conntrack_tftp - ip_nat_tftp : Conntrack e nat per TFTP
- ip_conntrack_rpc_tcp : Conntrack per RPC
- ip_conntrack_rpc_udp : Conntrack per RPC

I moduli vanno caricati nel kernel con il comando:

```
$ modprobe
```

### 2.2.1 Un esempio con ftp

definiamo un client $A$ ed un server ftp $B$ ed iniziamo la connessione ftp con il comando:

```
$ftp B
```

Questo comando aprirà una NUOVA (NEW) connessione verso il server $B$

```
client A    NEW    Server B
           ---->
```

Per brevità salto tutti i passaggi di autenticazione e passo direttamente al download di un file:

```
$get file.tgz
```

Quando il client chiede al server di eseguire il download di un file, per IpTables, questa è una connessione di tipo **ESTABLISHED**. Nel caso di utilizzo di **modalità passiva**, la porta di connessione del client è la 20, ma la porta per il trasferimento del file può essere una qualsiasi dalla 1023 in poi. Essendo la connessione generata da quella già esistente di ftp, con stessa sorgente IP e stessa destinazione IP, per permettere l'utilizzo della modalita passiva, dobbiamo usare lo stato **RELATED**.

## 2.3 Le tabelle e le catene

IpTables fornisce tre tabelle prestabilite, ognuna delle quali contiene delle catene predefinite. Inizialmente tutte le catene sono vuote e hanno una politica che consente a tutti i pacchetti di passare liberamente attraverso le interfacce di rete. Le catene vanno poi modificate a seconda delle proprie esigenze. Le tabelle predefinite sono le seguenti:

- tabella **filter**
  - filtra il traffico in ingresso,uscita e in transito
  - catene: INPUT, OUTPUT, FORWARD (Figura 1)

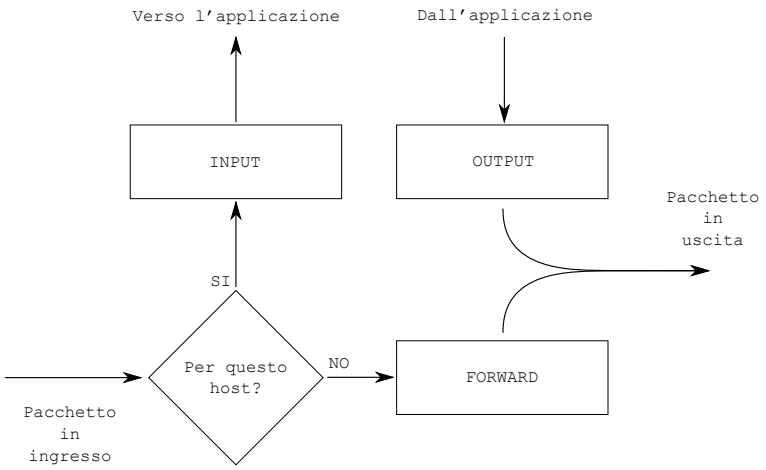


Figura 1: Funzionamento della tabella flter

- tabella **nat**: questa tabella è responsabile dell'impostazione delle regole per la modifica degli indirizzi e porte dei pacchetti. Il primo pacchetto di una connessione passa attraverso questa tabella, e il risultato del passaggio del primo pacchetto determina come tutti gli altri pacchetti della stessa connessione verranno modificati. La tabella nat contiene le seguenti catene predefinite:
  - catena PREROUTING: passano attraverso questa catena i pacchetti in entrata, il passaggio avviene prima che la locale tabella di routing venga consultata per effettuare l'instradamento. Essa è usata per il NAT sulla destinazione o DNAT.
  - catena POSTROUTING: passano attraverso questa catena i pacchetti in uscita dopo che la locale tabella di routing sia stata consultata. Usata per il NAT sulla sorgente o SNAT.
  - catena OUTPUT: permette un DNAT limitato sui pacchetti generati localmente.
- tabella **mangle**: questa tabella è responsabile delle modifiche alle opzioni dei pacchetti, come ad esempio quella che determina la qualità del servizio(TOS, MARK, ...). Tutti i pacchetti passano attraverso questa tabella. Essa contiene tutte le catene predefinite:

  - catena PREROUTING: esamina tutti i pacchetti che in qualche modo entrano nel sistema. Questo processo avviene prima che il routing decida se il pacchetto debba essere inoltrato (catena FORWARD) o se sia destinato al sistema. Viene utilizzata per manipolare l'header del pacchetto (catena INPUT).
  - catena INPUT: tutti i pacchetti destinati al sistema passano per questa catena.
  - catena FORWARD: tutti i pacchetti che vengono instradati dal sistema ma di cui il sistema non è ne sorgente iniziale ne destinazione finale, passano per questa catena.
  - catena OUTPUT: tutti i pacchetti generati dal sistema passano per questa catena.
  - catena POSTROUTING: tutti i pacchetti che lasciano il sistema, sia quelli in OUTPUT sia quelli in FORWARD, passano poi per questa catena.

- tabella **raw**: utilizzata molto raramente e che viene invocata prima delle altre, ha lo scopo di evitare il connection tracking per quei pacchetti che, per una qualche ragione, non si vogliono filtrare in maniera stateful. Le catene previste per la tabella raw sono solo due:
  - catena OUTPUT: in tale catena si andrà ad operare sui pacchetti generati da processi locali.
  - catena PREROUTING: in tale catena si andrà, invece, ad operare sui pacchetti provenienti da qualsiasi interfaccia di rete.

## 2.4 Azioni

Le azioni possibili da intraprendere per una determinata regola, si dividono in:

- **ACCEPT** ⇒ il pacchetto viene accettato;
- **REJECT** ⇒ il pacchetto viene rifiutato informando il mittente;
- **DROP** ⇒ il pacchetto viene completamente ignorato;
- **RETURN** ⇒ il pacchetto ritorna alla catena originale;
- una qualunque altra catena o funzione particolare come *LOG,ULOG,...* per la catena *filter*, o *DNAT,SNAT,...* per la catena *nat*, ...

nel caso di azioni come *LOG*, l'azione può non essere *terminante*[3] e quindi continuare con la regola successiva.

[3]Un'azione è considerata *non terminante* quando, un pacchetto che soddisfa una regola con obiettivo *non terminante*, invece di interromepere la valutazione delle regole, continua a essere confrontato con le regole successive.

# 3 Configurazione utilizzata

Grazie all'utilizzo di 3 macchine virtuali, è stato possibile simulare un'architettura composta da 3 diversi reti.

1. Una rete dedicata alla intranet dove trovano posto le postazioni di lavoro;
2. la rete DMZ, che ospita i server con i servizi da fornire ai dipendenti e alla rete internet;
3. la rete pubblica, che simula la rete internet.

Le macchine virtuali sono state predisposte come da Figura 2 per simulare:

- i computer dei dipendenti;
- i server con i servizi (www, mail, ftp, ...);
- il Firewall con la funzionalità NAT.

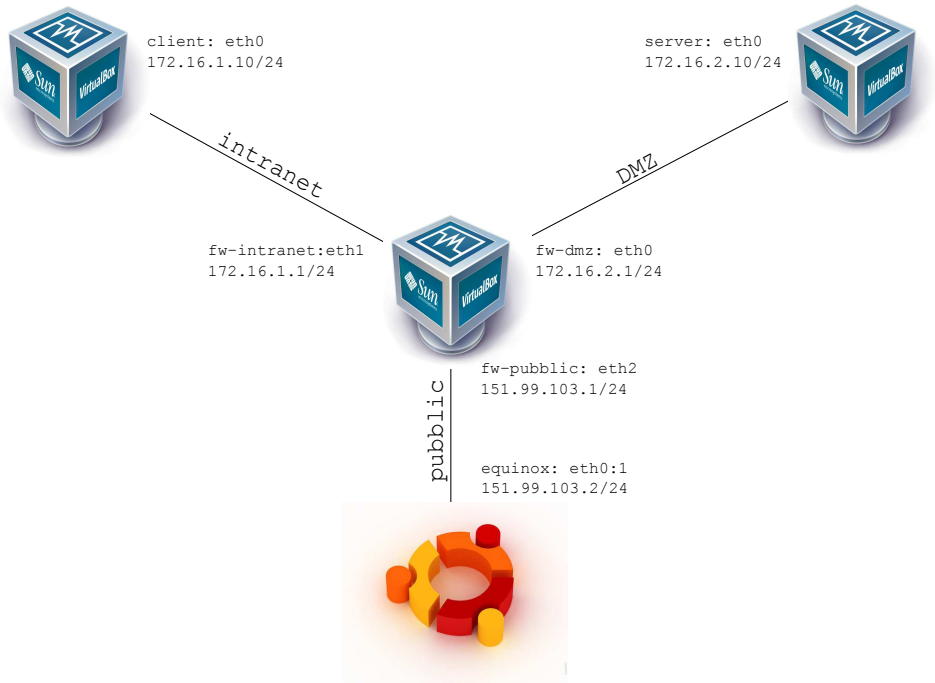


Figura 2: Diagramma delle reti

La macchina fisica ospitante le tre virtuali, farà le veci degli utenti internet, e sarà utilizzata per eseguire delle scansioni dei servizi aperti sul firewall e sulle macchine interne.

## 3.1 S.O. e pacchetti installati

Le tre macchine virtuali sono state installate con Debian. Gli unici pacchetti aggiunti all'installazione minima standard, sono stati:

- `tcpdump` ⇒ (per l'analisi del traffico sulle interfacce di rete);
- `vsftpd` ⇒ come server ftp;
- `sshd` ⇒ come server SSH;
- `lynx` ⇒ come browser web/ftp testuale;
- `nmap` ⇒ per eseguire scansioni sulle porte;
- `apache` ⇒ come server web

## 3.2 Piano d'indirizzamento

Per la macchina virtuale che simula la rete dei servizi, come quella dei dipendenti, è stato predisposto un piano di indirizzamento standard con una sola scheda di rete (vedi tabella 1). Per il firewall si è utilizzata una configurazione con tre schede di rete, una per la rete dei dipendenti, una per la rete dei servizi, ed infine l'ultima a simulare un collegamento con la rete internet (tabella 2). Il firewall utilizza come default gateway la macchina fisica, e conseguentemente, la macchina fisica è configurata per utilizzare l'indirizzo pubblico del firewall come default gateway.

(a) Server

| **server** | **eth0** |
|---|---|
| IP | 172.16.2.10 |
| netmask | 255.255.255.0 |
| gateway | 172.16.2.1 |

(b) Client

| **client** | **eth0** |
|---|---|
| IP | 172.16.1.10 |
| netmask | 255.255.255.0 |
| gateway | 172.16.1.1 |

Tabella 1: Tabelle Server Client

Tabella 2: Firewall

| **firewall** | **eth0** | **eth1** | **eth2** |
|---|---|---|---|
| IP | 172.168.2.1 | 172.168.1.1 | 151.99.103.1 |
| netmask | 255.255.255.0 | 255.255.255.0 | 255.255.255.0 |
| default gw | | | 151.99.103.2 |

# 4 requisiti della rete

Analizzando i bisogni di ogni computer o classi di computer connessi ad una rete, si è in grado di definire delle regole base per il firewall, in modo da consentire solo il traffico lecito da e verso una certa rete. Per fare questo, dobbiamo analizzare ogni singola rete e capire che tipo di traffico viene generato e la sua destinazione. Le analisi seguenti saranno improntate ad analizzare solo il traffico in uscita dalle singole reti. L'unione di tutte le tabelle ci darà modo di capire quale tipo di traffico dovrà essere permesso, andando poi a negare completamente tutto ciò che non è stato esplicitato.

## 4.1 intranet

Dalla rete **intranet** (tabella 3) verso la rete **DMZ**, dovrà essere possibile raggiungere i siti web intranet ed internet, spedire e ricevere la posta, utilizzare un ftp, e accedere al dns. Dalla rete **intranet** si deve anche poter arrivare su qualsiasi server www della rete internet[4]. Da questa specifica si ricava la tabella 3.

Tabella 3: Intranet

| **Source** | **Destination** | **Protocoll** | **Rule** |
|---|---|---|---|
| eth1 | eth0 | http | allow |
| eth1 | eth0 | ftp-data | allow |
| eth1 | eth0 | ftp | allow |
| eth1 | eth0 | smtp | allow |
| eth1 | eth0 | pop3 | allow |
| eth1 | eth0 | dns | allow |
| eth1 | eth2 | http | allow |

## 4.2 DMZ

La rete **DMZ** (tabella 4) per una corretta funzionalità della posta, dovrà fare delle connessioni smtp verso server di posta sulla rete internet. Per il corretto funzionamento dei dns, gli stessi dovranno essere abilitati ad instaurare delle connessioni di tipo dns verso server dns della rete internet.

Tabella 4: DMZ

| **Source** | **Destination** | **Protocoll** | **Rule** |
|---|---|---|---|
| eth0 | eth2 | smtp | allow |
| eth0 | eth2 | dns | allow |

[4]Per questo tipo di funzionalità è necessario prevedere la funzionalità di **MASQUERADE**

## 4.3 Internet

Dalla rete pubblica (tabella 5) non dovrà essere possibile raggiungere alcun computer o apparato, localizzato nella rete intranet. Si dovranno solamente raggiungere i server web e ftp, i server di posta e i dns, con gli opportuni protocolli, localizzati nella rete DMZ.

Tabella 5: Pubblic

| **Source** | **Destination** | **Protocoll** | **Rule** |
|---|---|---|---|
| eth2 | eth0 | http | allow |
| eth2 | eth0 | smtp | allow |
| eth2 | eth0 | dns | allow |
| eth2 | eth0 | ftp | allow |
| eth2 | eth0 | ftp-data | allow |

## 4.4 Conclusioni sulle regole

Dai requisiti delle singole reti si può arrivare a definire una tabella (tabella 6) comprensiva di tutte le regole del traffico permesso da e verso una certa rete.

Tabella 6: Regole complessive

| **Source** | **Destination** | **Protocoll** | **Rule** |
|---|---|---|---|
| eth1 | eth0 | http | allow |
| eth1 | eth0 | ftp-data | allow |
| eth1 | eth0 | ftp | allow |
| eth1 | eth0 | smtp | allow |
| eth1 | eth0 | pop3 | allow |
| eth1 | eth0 | dns | allow |
| eth1 | eth2 | http | allow |
| eth0 | eth2 | smtp | allow |
| eth0 | eth2 | dns | allow |
| eth2 | eth0 | http | allow |
| eth2 | eth0 | smtp | allow |
| eth2 | eth0 | dns | allow |
| eth2 | eth0 | ftp | allow |
| eth2 | eth0 | fftp-data | allow |

# 5 Implementazione

## 5.1 Impostiamo l'ip_forward

Per poter utilizzare la catena **FORWARD** è necessario che lo strato software del sistema operativo che gestisce le connessioni, sia abilitato a far transitare i pacchetti da una interfaccia ad un altra. Le impostazioni possono essere effettuate *on the fly*[5] intervenendo sul file:

```
/proc/sys/net/ipv4/ip_forward
```

Questo flag è utilizzato dal sistema operativo come un *flag* per sapere il comportamento da intraprendere nel caso che un pacchetto in arrivo da un'interfaccia, debba uscire per un'altra. Va impostato a **1** per abilitare il *forward* ed a **0** per disabilitarlo.

## 5.2 Blocchiamo tutto il traffico

Sul firewall impostiamo le azioni di *default* per le catene della tabella *filter*[6].

```
$ iptables -t filter P INPUT DROP
$ iptables -t filter P OUTPUT DROP
$ iptables -t filter P FORWARD DROP
```

## 5.3 da Intranet a Pubblic

Impostiamo la funzionalità di **MASQUERADE** in uscita sull'interfaccia eth2 (pubblica) con il comando

```
$ iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE
```

Abilitamo la navigazione internet da parte dei client sulla rete intranet. Per questa regola che comprende le porte 80 (hhtp) e 443 (https), possiamo utilizzare l'attributo *multiport* che permette di specificare in una sola regola più porte. Dobbiamo anche abilitare il traffico di ritorno utilizzando l'attributo *state*.

```
$ iptables -A FORWARD -i eth1 -o eth2 -p tcp -m multiport --dports
          80,443 -j ACCEPT

$ iptables -A FORWARD -i eth2 -o eth1 -m state --state ESTABLISHED
          -j ACCEPT
```

**Ogni altro tipo di traffico verso internet verrà *DROPPATO* dalla regola di default**

[5] In realtà questo comando andrebbe dato alla fine della configurazione del firewall, per evitare che il firewall venga utilizzato *maliziosamente* in corso di configurazione

[6] Di default, se omessa la tabella nelle regole, viene considerata sempre la tabella *filter*. In questo caso per una più facile lettura viene sempre indicata la tabella di appartenenza

## 5.4 da Intranet a DMZ

Per permettere l'utilizzo di **ftp**[7], ricordiamoci di caricare sul kernel il modulo che consente il funzionamento in modalità stateful per il protocollo **ftp**, **ip_conntrack_ftp**, ed abilitare il traffico ESTABLISHED e RELATED dalla DMZ ⇒ intranet. Il protocollo dei DNS prevede anche pacchetti di tipo **udp** e quindi, in aggiunta alla regola con il multiport, dobbiamo inserire anche una regola che permetta il traffico di tipo *domain* utilizzando il protocollo **udp**.

```
$ modprobe ip_conntrack_ftp
$ iptables -A FORWARD -i eth0 -o eth1 -m state
           --state ESTABLISHED,RELATED -j ACCEPT

$ iptables -A FORWARD -i eth1 -o eth0 -p tcp -m multiport
           --dports 53,80,443,20,21,25,110 -j ACCEPT
$ iptables -A FORWARD -i eth1 -o eth0 -p udp --dport 53 -j ACCEPT
```

## 5.5 da Internet (pubblic) a DMZ

Per questa funzionalità non dobbiamo dimenticarci di eseguire il *nat* tra l'indirizzo pubblico del firewall e l'indirizzo interno delle macchine con i servizi da pubblicare. Parlando del server web, e quindi delle porte 80 e 443, la regola che andremo ad inserire andrà a sostituire l'indirizzo ip pubblico del destinatario con l'indirizzo ip privato del server web, se e solo se il pacchetto è diretto all'indirizzo pubblico del firewall e destinato alle porte 80 e 443. La regola andrà inserita nella catena *PREROUTING* della tabella **nat**, e quindi, prima di applicare le tabelle di routing, il pacchetto viene trasformato e poi con il nuovo destinatario, si applicano le politiche di routing. Bisogna fare ancora un po' di attenzione, perchè in questo modo non stiamo evitando che un pacchetto destinato all'indirizzo privato del server web, transiti dall'interfaccia pubblica del firewall. Nella realtà è un po' difficile che questo succeda[8], in quanto, il router che ci connette ad internet filtra tutti i pacchetti con indirizzi privati, ma, dal momento che la sicurezza aumenta all'aumentare del grado di paranoia del *network administrator*, aggiungere una regola che scarti tutti i pacchetti provvenienti dall'interfaccia pubblica del firewall e diretti verso un ip privato[9], della DMZ, ci farà sentire più tranquilli.

```
iptables -A FORWARD -i eth2 -o eth0 -m state --state NEW,ESTABLISHED,
          RELATED -j ACCEPT
iptables -A FORWARD -i eth0 -o eth2 -m state --state ESTABLISHED,
         RELATED -j ACCEPT
iptables -t nat -A PREROUTING -p tcp -i eth2 -d 151.99.103.1 --dport 80
         -j DNAT --to-destination 172.16.2.10
```

[7]Se si volesse far utilizzare l'ftp in modalità *passiva*, avremmo dovuto impostare una regola del tipo: *da eth1 a eth0 stato ESTABLISHED,RELATED ACCEPT*

[8]Può succedere che il nostro router venga compromesso e, da quella posizione, da noi considerata sicura, si potrebbero sferrare degli attacchi alle nostre macchine

[9]in realtà ci sarebbero anche altre classi d'indirizzi da filtrare che ho omesso volutamente per non appesantire il tutto

Ora blocchiamo tutto il traffico diretto un qualsiasi indirizzo privato e proveniente dall'interfaccia pubblica del firewall:

```
iptables -t nat -A PREROUTING -i eth2 -d 10.0.0.0/8 -j DROP
iptables -t nat -A PREROUTING -i eth2 -d 172.16.0.0/12 -j DROP
iptables -t nat -A PREROUTING -i eth2 -d 192.168.0.0/16 -j DROP
```

## 5.6 Salvare le regole

Tutto ciò che è stato fatto finora, verrà perso al successivo riavvio della macchina. Dobbiamo salvare le regole, per poi ripristinarle ai successivi riavvii della macchina. Molto importante ai fini della sicurezza, è attivare le regole **prima** che l'interfaccia di rete diventi attiva, per evitare qualsiasi tipo di connessioni **non autorizzate** durante la fase di *start-up* della macchina. IpTables ci viene in aiuto con i comandi:

```
$ iptables-save
$ iptables-restore
```

che ci consentono di utilizzare lo *standard-out* e lo *standard-in* per salvare e leggere le regole impostate. Nello specifico, **iptables-save** stampa a video le regole impostate in un formato comprensibile a **iptables-restore**, conseguentemente, **iptables-restore** legge le regole nel formato prodotto da **iptables-save**. É quindi sufficiente, redirigere lo *standard-out* verso un file da far poi leggere come *standard-in* da **iptables-restore**. Distribuzioni differenti di Linux, possono prevedere diversi sistemi per salvare e leggere le regole di *iptables*, ma fattore comune a tutte le distribuzioni, è l'utilizzo di **iptables-save** e **iptables-restore**, fornito dal pacchetto **iptables**. Imparato l'uso dei comandi forniti da **iptables**, si è in grado di salvare e ripristinare le regole in una qualsiasi distribuzione Linux. Salviamo le regole con il comando:

```
$ iptables-save > rule.txt
```

e ripristiniamo con:

```
$ iptables-restore < rule.txt
```

questi comandi andranno inseriti in uno script, che verrà eseguito **prima** della configurazione dell'interfaccia di rete. Rimando ai vari **System V**[10] di ogni distribuzione la ricerca di dove e come inserire i comandi per il *save* e *restore* di iptables. Nel paragrafo 8 ho inserito tutte le regole utilizzate fino ad ora, dove non si fa uso di *iptables-save* e *iptables-restore*. Questo però significa che saremo costretti a modificare lo script ogniqualvolta avessimo bisogno di cambiare le regole. Nello script ho anche inserito il modulo per consentire la modalità *statefull* per il protocollo ftp. Se si opta per l'uso dei comandi *iptables-save* e *iptables-restore*, ricordarsi di, utilizzando le giuste modalità, far caricare al kernel i moduli necessari al corretto funzionamento di *iptables*.

[10] UNIX System V, spesso abbreviato in System V, è una versione del sistema operativo proprietario Unix

# 6 Test e verifiche funzionali

## 6.1 ip_forward

Dalla macchina fisica, lanciare un ping verso l'ip 172.16.1.10. Se l'ip_forward è stato abilitato (sezione 5.1) si dovrebbe poter ricevere gli **echo reply**. Se così non fosse, ripetere il test dopo aver dato:

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

## 6.2 Verifica delle politiche di default(DROP)

Con un semplice ping dalla macchina fisica, verso il firewall o verso una delle macchine virtuali poste dietro il firewall, o anche dalla rete intranet verso la rete DMZ, verificare che regole impostate nella sezione 5.2 funzionino correttamente.

## 6.3 Il masquerating

dopo aver dato il comando per accettare qualsiasi pacchetto[11] in *forwarding*, mettere la macchina fisica in *sniffing*[12] sull'interfaccia pubblica (eth2) del firewall. Dalla macchina virtuale **intranet** lanciare un ping così formato:

```
$ ping 151.99.103.2
```

Dal *tcpdump* verificare che il risultato sia del tipo:[13]

```
172.16.1.10 > 151.99.103.2: ICMP echo request
151.99.103.2 > 172.16.1.10: ICMP echo reply
```

Si evince che sull'interfaccia della nostra macchina fisica sta arrivando una richiesta di ping con indirizzo ip sorgente privato. In un caso reale, questo tipo di pacchetto sarebbe stato bloccato dal primo router incontrato per arrivare su internet. Per fortuna non è il nostro caso, e quindi possiamo verificare il corretto funzionamento dell'istruzione vista in 5.3. Dopo l'abilitazione del **MASQUERADE** verificare che l'output sia il seguente:

```
151.99.103.1 > 151.99.103.2: ICMP echo request
151.99.103.2 > 151.99.103.1: ICMP echo reply
```

L'indirizzo, prima privato, è ora *nattato* con quello pubblico del firewall, e la richiesta può essere tranquillamente inoltrata verso la rete internet.

[11] un'alternativa più sicura, è quella di abilitare il solo icmp di tipo *echo request* con il comando: *iptables -A FORWARD -i eth1 -o eth2 -p icmp –icmp-type echo-request -j ACCEPT*

[12] tcpdump -ni eth2 icmp

[13] ho omesso gli altri dati come il seq. number,...

| **tcpdump -ni eth2 port 80** | **tcpdump -ni eth0 port 80** |
|---|---|
| 151.99.103.2.53910 ⇒ 151.99.103.1.80 | 151.99.103.2.53910 ⇒ 172.16.2.10.80 |
| 151.99.103.1.80 ⇒ 151.99.103.2.53910 | 172.16.2.10.80 ⇒ 151.99.103.2.53910 |
| 151.99.103.2.53910 ⇒ 151.99.103.1.80 | 151.99.103.2.53910 ⇒ 172.16.2.10.80 |
| 151.99.103.2.53910 ⇒ 151.99.103.1.80 | 151.99.103.2.53910 ⇒ 172.16.2.10.80 |
| 151.99.103.1.80 ⇒ 151.99.103.2.53910 | 172.16.2.10.80 ⇒ 151.99.103.2.53910 |
| ⋮ | ⋮ |

Tabella 7: Confronto tra traffico in ingresso ed uscita

## 6.4 Il NAT

Per testare la corretta funzionalità del NAT, è sufficiente *sniffare*[14] il traffico sulle 2 interfacce del firewall, quella pubblica (eth2) e quella sulla DMZ (eth0), e verificare che la trasformazione del pacchetto funzioni correttamente. Il risultato atteso, in caso di una connessione http generata dalla macchina fisica e diretta verso l'ip del firewall porta 80 è come da tabella[15] 7. É facilemente verificabile che sull'interfaccia pubblica (eth2) transitano solo pacchetti validi per quel segmento di rete, mentre nella parte DMZ (privata), il pacchetto è stato trasformato per poter raggiungere il servizio richiesto sulla porta 80 del server con IP privato. Questo tipo di configurazione ci dà la possibilità di non *sprecare* indirizzi pubblici, ma di utilizzare un solo IP per poi fare un *NAT* verso gli ip privati dei servizi che vogliamo mettere a disposizione, purché i servizi utilizzino differenti porte. Se ad esempio avessimo 2 server http nella nostra DMZ, tutti e due starebbero in ascolto sulla porta 80 e questo ci impedirebbe di discriminare, attraverso l'uso della porta, su quale dei due server poter indirizzare le richieste.

## 6.5 test con nmap

Grazie al tool **nmap** siamo in grado di verificare velocemete quali porte si rendono disponibili sulle nostre macchine e per ogni rete. Rimando al manuale di nmap[16] l'utilizzo dello stesso. Nel frattempo ci limiteremo solo a verificare che nessuna macchina con ip privato, sia raggiungibile dall'esterno, e che le porte aperte sull'interfaccia pubblica del firewall, siano le sole *nattate* ai servizi interni.

# 7 Conclusioni e sviluppi futuri

Il firewall è sicuramente migliorabile, e non completamente esaustivo sul piano della sicurezza di rete. É però una base sufficiente a capire il funzionamento di un tool tanto potente quanto complicato come **IpTables**. Non sarebbe male, ad esempio, attivare il log su tutte le nuove connessioni di tipo SSH, o bloccare tutti i port scanner di tipo un po' anomalo come quelli che utilizzano i flag tcp (SYN,ACK,FIN,RST), ....

[14] tcpdump -ni eth2 port 80, tcpdump -ni eth0 port 80

[15] ometto volutamente tutte le informazioni non neccessarie alla comprensione della regola

[16] nmap –help

# 8 Lo script delle regole

```
#!/bin/bash

I="/sbin/iptables"
PUB="eth2"
DMZ="eth0"
INT="eth1"

echo "1" > /proc/sys/net/ipv4/ip_forward
modprobe ip_conntrack_ftp

### Blocco tutto ###
$I -P OUTPUT DROP
$I -P INPUT DROP
$I -P FORWARD DROP

############################
### Intranet -> Pubblic ###
### solo http e https   ###
### MASQUERADE          ###
############################
$I -t nat -A POSTROUTING -o $PUB -j MASQUERADE
$I -A FORWARD -i $INT -o $PUB -p tcp -m multiport --dports 80,443
   -j ACCEPT
$I -A FORWARD -i $PUB -o $INT -m state --state ESTABLISHED,RELATED
   -j ACCEPT

############################
### Intranet -> DMZ     ###
### ########################
$I -A FORWARD -i $DMZ -o $INT -m state --state ESTABLISHED,RELATED
   -j ACCEPT
$I -A FORWARD -i $INT -o $DMZ -p tcp -m multiport
   --dports 53,80,443,20,21,25,110 -j ACCEPT
$I -A FORWARD -i $INT -o $DMZ -p udp --dport 53 -j ACCEPT

############################
### Pubblic -> DMZ      ###
### 80,443,25,53        ###
### NAT                 ###
############################
$I -A FORWARD -i $PUB -o $DMZ -m state --state NEW,ESTABLISHED,RELATED
   -j ACCEPT
$I -A FORWARD -i $DMZ -o $PUB -m state --state ESTABLISHED,RELATED
   -j ACCEPT
```

```
$I -t nat -A PREROUTING -p tcp -i $PUB -d 151.99.103.1 --dport 80
   -j DNAT --to-destination 172.16.2.10
$I -t nat -A PREROUTING -p tcp -i $PUB -d 151.99.103.1 --dport 443
   -j DNAT --to-destination 172.16.2.10
$I -t nat -A PREROUTING -p tcp -i $PUB -d 151.99.103.1 --dport 25
   -j DNAT --to-destination 172.16.2.10
$I -t nat -A PREROUTING -p tcp -i $PUB -d 151.99.103.1 --dport 53
   -j DNAT --to-destination 172.16.2.10
$I -t nat -A PREROUTING -p udp -i $PUB -d 151.99.103.1 --dport 53
   -j DNAT --to-destination 172.16.2.10
### blocchiamo tutte le classi private in ingresso ###
$I -t nat -A PREROUTING -i $PUB -d 10.0.0.0/8 -j DROP
$I -t nat -A PREROUTING -i $PUB -d 172.16.0.0/12 -j DROP
$I -t nat -A PREROUTING -i $PUB -d 192.168.0.0/16 -j DROP

############################
### DMZ -> Pubblic      ###
### 25,53               ###
############################
$I -A FORWARD -i $DMZ -o $PUB -p tcp -m multiport --dports 25,53
   -j ACCEPT
```