Introduzione alle reti in ambiente GNU/Linux
e
al firewalling con iptables

Linux User Group Mantova
LinuxDay 2003, Pegognaga (MN)

Lorenzo Grespan
lorenzo@muug.it

28 novembre 2003

Indice

1 Introduzione

1.1 panoramica di questo documento

Lo scopo di questo documento è un'introduzione alle reti e al firewalling sotto UNIX in generale e GNU/Linux in particolare. Non è una guida completa a questi due argomenti, ma va inteso come punto di partenza per approfondirne la conoscenza.

Questo testo si basa interamente sulla mia esperienza, e come tale va inteso. Eventuali errori e omissioni sono da imputarsi unicamente all'autore.

1.2 requisiti

Requisiti per la comprensione di questo documento sono una certa familiarità con il sistema operativo GNU/Linux, indipendentemente dalla distribuzione; è necessario avere dimestichezza con un editor possibilmente da console, quale vi/vim, emacs, nano, eccetera. È utile inoltre avere chiari i concetti di filesystem, di interprete dei comandi (bash in particolare) e le basi del networking. Si cercherà di approfondire l'argomento "reti" in modo empirico, ovvero basandosi su prove pratiche e tralasciando la teoria ove non strettamente necessaria.

La conoscenza dell'"Inglese Informatico" non è utile per la comprensione specifica di questo testo ma è pressochè fondamentale per approfondirne gli argomenti trattati.

1.3 per chi è stato scritto questo documento

Ho scritto questo testo, oltre che per il mio workshop al LinuxDay 2003 a Pegognaga (MN), soprattutto per me stesso. Ho cercato di scrivere il documento che non ho mai trovato "per inziare ad imparare le reti" sotto GNU/Linux, spiegando i dubbi che ho risolto nel corso degli anni ed elencando i comandi essenziali cercando di non perdermi nei dettagli ove non fosse strettamente indispensabile. Questo testo è quindi rivolto ad appassionati di questo sistema operativo e a tutti quelli che desiderano approfondire la propria conoscenza in questo campo.

2 Configurazione di rete

2.1 ambiente di rete: introduzione, concetti, riconoscimento hardware

Durante l'installazione della maggior parte dei sistemi UNIX si chiede di configurare la rete e di predisporre il computer sul quale si sta installando (in questo caso specifico) GNU/Linux per un utilizzo in un ambiente di networking. Anche se si pensa di mantenere il proprio computer isolato dal resto del mondo, il networking è parte integrante di questo sistema operativo anche per il funzionamento in locale, ovvero "a se stante". Inoltre oggi è raro avere anche solo un personal computer non connesso ad una rete di qualche tipo, che sia una rete locale (LAN, Local Area Network) o una rete internazionale (quale Internet).

Si suppone che si abbia una versione funzionante di GNU/Linux e che essa sia configurata in modo da permettere di usare il sistema secondo le proprie esigenze. I comandi che si vedranno nel resto di questo documento sono tutti validi nell'ambiente della console, e proprio questa è la loro potenza; infatti mediante questi comandi sarà possibile configurare il networking di computer situati a distanza, senza dover essere presenti fisicamente davanti alla macchina e senza dover dipendere da un'interfaccia grafica.

Si darà per scontato che le schede di rete siano già riconosciute dal sistema operativo, e che gli eventuali rispettivi driver vengano caricati automaticamente. A titolo di esempio, mediante il comando

```
ifconfig eth0
```

si avrà un elenco di tutte le interfacce di rete presenti sul nostro sistema. Nel caso il comando dovesse ritornare un errore, sarà necessario caricare i driver per questa scheda per poterla usare. Quindi con il comando

```
lspci
```

si dovranno verificare le periferiche di cui dispone il nostro computer. L'output del comando sarà simile a questo:

```
00:01.1 Ethernet controller:  Silicon Integrated Systems [SiS] SiS900
10/100 Ethernet (rev 82)
```

quindi la nostra scheda sarà una Sis. Il corrispondente driver da caricare si troverà nella directory

```
/lib/modules/<versione del kernel in uso>/kernel/drivers/net/
```

e avrà nome (in questo caso)

```
sis900.o
```

Nota: su kernel di tipo 2.6, il nome del modulo non sarà più "sis900.o" bensì "sis900.ko".

Volendo essere più specifici, il file "sis900.o" non è propriamente un "driver" nell'accezione che si ha in ambiente Microsoft; esso è un modulo del kernel. La differenza è sottile, ma si nota quando ad esempio si dovranno usare moduli per abilitare certe funzioni avanzate di rete (cosa che non si può fare con un semplice "driver" di una periferica), indipendentemente dall'hardware utilizzato. Il "kernel" è il cuore del sistema operativo, ciò che sta tra l'utente e l'hardware; esso gestisce l'accesso alla memoria e al processore, la rete, l'output, l'input, e via dicendo. Ogni sistema operativo è composto da un kernel e da applicazioni; sotto GNU/Linux, ciò che si chiama "Linux" è solo il kernel, e le applicazioni appartengono all'insieme più generale denominato "GNU".

Tornando al discorso precedente, mediante il comando

```
lsmod
```

si vede se questo modulo è caricato o meno nel kernel; se non lo fosse, andrà caricato con

```
modprobe sis900
```

si noti l'assenza dell'estensione nel nome del file (".o"). A questo punto riprovando "ifconfig eth0" dovrebbero comparire informazioni relative alla scheda di rete appena rilevata.

Utile per diagnosticare problemi di questo tipo è il comando

```
dmesg
```

che permette di vedere l'output ed eventuali messaggi di errore. "dmesg" riporta tutto quello che viene stampato a schermo durante l'avvio del sistema; le ultime righe invece si riferiscono alle ultime operazioni eseguite che hanno restituito un messaggio.

Non si vedranno ora gli eventuali problemi di riconoscimento dell'hardware e caricamento dei moduli; a questo proposito è disponibile in rete ogni informazione necessaria.

2.2 comandi di sopravvivenza: ifconfig, route, ping

In questo paragrafo si inizierà a parlare di indirizzi IP e di risoluzione dei nomi. Questi concetti verranno spiegati in dettaglio nel prossimo capitolo, per ora ci si accontenti di sapere che un "indirizzo IP" è un identificativo univoco di un computer in una rete, e la "risoluzione dei nomi" è un processo che ci permette di risalire dal nome alfanumerico di una macchina al suo indirizzo IP.

Il primo comando che si discuterà è stato mostrato poco fa e si tratta di

```
ifconfig
```

che permette non solo di vedere a schermo le interfacce di rete attive sul computer, ma anche di avere per ognuna di esse alcuni parametri utili a capire il suo corretto funzionamento. Infatti, questo comando viene usato per configurarne l'indirizzo (ovvero ciò che identifica un computer su una rete) e per capire se c'è un flusso di dati in ingresso o in uscita.

Come accennato in precedenza, un'interfaccia vista mediante il comando "ifconfig" non è necessariamente un dispositivo fisico; essa può anche essere una rappresentazione virtuale di una connessione mediante un determinato protocollo. Ad esempio, una connessione via modem viene gestita mediante l'interfaccia

`ppp`
(generalmente ppp0, può essere presente anche come ppp1, ppp2, eccetera)

Essa non è una scheda inserita nel computer, bensi è una scheda "virtuale" che il kernel sta utilizzando per un determinato tipo di collegamento. Al contrario, una scheda di rete standard che utilizzi il protocollo Ethernet sarà chiamata

`eth`
(anche in questo caso si potrà avere eth0, eth1, eth2...)

Nota: iptables, che verrà spiegato in seguito, usa un "alias" per indicare tutte le interfacce di rete di un certo tipo. Esso sarà della forma

`eth+`

e quando usato farà riferimento a tutte le interfacce di tipo "eth".

Se si dovesse aggiungere una scheda di rete dopo aver già configurato la scheda esistente, può succedere che la nuova interfaccia prenda il nome di "eth0" e la vecchia diventi "eth1". Questo spostamento dipende dall'ordine con cui il sistema operativo va a "cercare" le schede sui bus interni, ed è spesso imprevedibile.

Usando il comando

```
ifconfig eth0
```

si vedrà un output di questo tipo (ovviamente i valori possono cambiare):

```
eth0      Link encap:Ethernet  HWaddr 00:90:F5:0A:59:01
          inet addr:192.168.2.15  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:85976 errors:0 dropped:0 overruns:0 frame:0
          TX packets:27392 errors:0 dropped:5 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:109383099 (104.3 MiB)  TX bytes:4027863 (3.8 MiB)
          Interrupt:10 Base address:0x3200
```

Ora non è importante comprendere il significato di ogni punto; per adesso si noti soltanto la terza riga (che indica, con "UP", lo stato attuale dell'interfaccia) e la seconda riga (che indica l'indirizzo della scheda all'interno della rete).
Il significato degli altri parametri è il seguente:

eth0 è il nome della scheda

Link encap:Ethernet indica il protocollo

HWaddr indica l'indirizzo fisico (MAC) della scheda, che e' univoco e dal quale e' possibile risalire al produttore e al numero di serie della scheda stessa

inet addr: indica che il tipo di indirizzo che segue è di tipo "inet", seguito dall'indirizzo IP (spiegato in seguito), che il "broadcast" ha un certo indirizzo e che la "netmask" assume un certo valore.

UP (eccetera) il flag "UP" ci indica che l'interfaccia è attivata. gli altri flag indicano che l'interfaccia risponderà a particolari richieste (di tipo "BROADCAST"), che è vista come funzionante ("RUNNING"), che ha un indirizzo di tipo "MULTICAST"

MTU: Maximum Transfer Unit, ovvero la quantità massima di dati che è possibile inviare in una singola unità di trasmissione

RX/TX packets: indica i pacchetti totali ricevuti, specificando gli errori ("errors"), i pacchetti scartati ("dropped"), i pacchetti scartati a causa della loro elevata velocità di arrivo ("overruns") e il numero di frames ricevuti.

collisions: il numero di collisioni rilevate in rete con altre trasmissioni

RX/TX bytes: bytes ricevuti e inviati

Interrupt: l'interrupt hardware dell'interfaccia

Base address: l'indirizzo di memoria al quale è caricata l'interfaccia

Nota: questo comando si trova solitamente nella directory /sbin/. Se lo eseguite da un utente non root probabilmente sarà mostrato un messaggio di errore del tipo "comando non trovato". Conviene quindi diventare superutente (con il comando

```
su -
```

(si noti il segno "-" che permette di importare le variabili di ambiente dell'utente root nell'utente corrente) e lavorare da root. La ragione per cui il comando non viene trovato è che gli utenti normali non hanno configurato nel loro PATH (l'elenco delle directory nelle quali vengono cercati i comandi da eseguire) la directory /sbin che storicamente in ambito UNIX contiene i comandi utili solo all'amministratore di sistema.

Il PATH è per l'appunto una variabile di sistema definita per ogni utente nei files .bashrc e .bash_profile (sotto GNU/Linux) presenti nella home directory.

Si ricordi che tutti i comandi che si useranno da qui in poi richiedono i permessi di root per agire sul sistema (quindi per modificare indirizzi, percorsi dei pacchetti, eccetera) mentre si accontenteranno dei permessi di un utente qualunque per restituire i parametri che ci interessano. In pratica eseguendo /sbin/ifconfig da utente si vedrà l'indirizzo IP ma non si potrà modificare nulla.

Un altro comando utile è

```
route -n
```

che mostrerà un output del tipo

```
Kernel IP routing table
Destination  Gateway      Genmask        Flags Metric Ref  Use Iface
192.168.2.0  0.0.0.0      255.255.255.0  U     0      0      0 eth0
0.0.0.0      192.168.2.1  0.0.0.0        UG    0      0      0 eth0
```

Nota: il flag -n del comando route chiede a quest'ultimo di mostrarci solo gli indirizzi IP e di non cercare di risolverli nei loro nomi. Questo rende l'output più veloce, perchè il sistema operativo non è costretto a "chiedere in rete" a chi appartiene un certo indirizzo.

L'output del comando route ci dice che i pacchetti aventi per destinazione la rete "192.168.2." non hanno un gateway (ovvero un computer che li instrada verso altre destinazioni), e usciranno dal compuer attraverso l'interfaccia (Iface) eth0. Inoltre, i pacchetti con tutte le altre destinazioni (Destination 0.0.0.0, detta anche "default") vengono mandati all'indirizzo 192.168.2.1 (Gateway) per poter essere instradati correttamente; questo Gateway si deve trovare nella stessa nostra rete, e farà da "ponte" tra la macchina locale e la destinazione da raggiungere.

Nota: in GNU/Linux non si è limitati all'unica scheda di rete installata fisicamente sulla macchina. Si possono infatti creare schede di rete "virtuali" con indirizzi IP differenti mediante il concetto di "alias". Ad ogni scheda fisica installata si possono associare molti alias, in questo modo:

```
ifconfig eth0:0 10.0.0.1 up
```

```
ifconfig eth0:prova 10.1.1.1 up
```

facendo poi riferimento alle nuove interfacce virtuali create mediante la sintassi "ethX:nome", dove "nome" è un nome interno che usiamo per distinguerle e X è il numero della scheda di rete fisica. Ovviamente, se si dovesse disattivare la scheda principale con il comando

```
ifconfig eth0 down
```

tutte le altre schede virtuali non risulteranno più connesse.

"ping" è un comando che ci permette di verificare se un computer remoto è funzionante:

```
ping 192.168.2.2
PING 192.168.2.2 (192.168.2.2): 56 data bytes
64 bytes from 192.168.2.2: icmp_seq=0 ttl=63 time=1.0 ms
64 bytes from 192.168.2.2: icmp_seq=1 ttl=63 time=0.5 ms
64 bytes from 192.168.2.2: icmp_seq=2 ttl=63 time=0.4 ms

--- 192.168.2.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.6/1.0 ms
```

il comando continuerà all'infinito ad inviare piccoli pacchetti di dati e ad attendere risposta. Per interromperne l'esecuzione e vedere le statistiche di trasmissione (tempo minimo, medio e massimo di ricezione) è necessario premere CTRL-C.

2.3 in dettaglio: indirizzi IP

Un indirzzo IP, come il nome suggerisce, è un codice univoco che identifica un "qualcosa" connesso ad una rete informatica. "IP" indica un particolare protocollo ("Internet Protocol") standard utilizzato per trasportare messaggi. Mediante IP si possono identificare le parti fondamentali di un messaggio, ovvero mittente e destinatario.

Di fatto un indirizzo IP è rappresentato con un insieme di quattro gruppi di cifre decimali, divise dal simbolo di punto ("."). Ogni gruppo di cifre va da 0 a 255 per ragioni che verranno introdotte più avanti. Mediante l'indirizzo si identifica anche la rete di appartenenza: essa è la parte più a sinistra dell'indirizzo. Questa definizione generica è necessaria proprio per la versatilità degli indirizzi IP, che permettono di identificare in maniera univoca molte reti di computer o molti computer connessi alla stessa rete.

Ad esempio, un indirizzo IP del tipo

```
192.168.2.15
```

con una "netmask" di tipo

```
255.255.255.0
```

indica che la parte identificativa della rete di quell'indirizzo è composta dai primi tre gruppi di cifre. Quindi essa sarà:

```
192.168.2
```

e che la parte restante indica come è stato numerato il computer che ha quell'indirizzo. Gli altri computer di quella rete potranno avere indirizzi compresi tra

```
192.168.2.1
```

e

```
192.168.2.254
```

Nota: l'indirizzo .255 è riservato per indicare la rete intera di appartenenza, e l'indirizzo terminante con ".0" indica invece tutta la rete.

A titolo di esempio, l'indirizzo IP

```
10.0.0.1
```

con netmask

```
255.0.0.0
```

indica che la rete è identificata da

```
10.
```

mentre l'host è numerato come 0.0.1 di quella rete. Altri computer (o stampanti, o server, eccetera) potranno avere indirizzo 10.0.1.2, 10.1.0.1, e così via. In questo caso abbiamo una rete nella quale possono essere definiti 255 elevato alla terza potenza computer. Decisamente un numero elevato.

Tutte le trasmissioni effettuate da una scheda di rete vengono codificate in bit ("1" e "0") e proprio per questa ragione ogni campo di un indirizzo IP si limita al valore 255. Infatti, 255 è 2 elevato alla ottava potenza (si parte dal valore 0 e non dal valore 1). Ogni indirizzo IP è identificato da quattro gruppi di 8 bit. Si ricordi che 8 bit indicano un byte; quindi ogni indirizzo IP è identificato da 4 byte. Questo permette un buon risparmio di dati da inviare per ogni pacchetto, e consente un buon modo di identificare univocamente "qualcosa" in un insieme.

2.4 routing di base

Per "routing" si intende il percorso che un pacchetto dovrà seguire per giungere alla propria destinazione. Una rete infatti può essere costruita connettendo tra loro altre reti, messe in comunicazione mediante gateway che svolgeranno la funzione di "ponte" tra una rete e la sua vicina. Apparecchi di questo tipo possono essere sia dei computer che degli apparecchi dedicati. Essi prendono il nome di "router" e sono spesso delle scatole di metallo con un lato predisposto all'inserimento di cavi, e alimentati dalla normale tensione elettrica. Al loro interno esse sono dotate di un processore e un sistema operativo che si occupa di "prendere i pacchetti" da una porta in arrivo e di smistarli sulla porta alla quale è connesso fisicamente il destinatario.

Come abbiamo visto in precedenza, il comando

```
route -n
```

ci restituisce un output del tipo

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.2.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
0.0.0.0         192.168.2.1     0.0.0.0         UG    0      0        0 eth0
```

ora sappiamo che all'indirizzo 192.168.2.1 c'è un router che si occuperà di mandare i nostri pacchetti a destinazione. La potenza di questo sistema è che non è necessario per la singola macchina connessa alla rete essere a conoscenza del percorso migliore per far raggiungere ai propri dati la loro destinazione. Essa conosce solo a chi mandarli per instradarli nella direzione corretta.

In questo caso specifico "192.168.2.15" è un computer connesso fisicamente ad una rete locale (mediante cavi di rete) e "192.168.2.1" è un modem ADSL. L'utente del computer vuole visualizzare una pagina web; egli quindi dovrà stabilire una connessione verso il server web che ospita la pagina richiesta. Per stabilire una connessione dovrà inviare dei dati sotto forma di "pacchetti" attraverso Internet; il computer inizierà ad inviare questi dati per prima cosa al modem, che si occuperà a sua volta di passarli ad un altro router (questa volta su Internet), che li passerà ad un altro, e così via fino a raggiungere la destinazione.

Ogni router interpellato non conosce tutti i dettagli della destinazione; è sufficiente che conosca il percorso migliore per instradare i pacchetti. In questo modo si è costruita una rete globale dalle enormi potenzialità, capace anche di sopravvivere a guasti e disastri locali. Se un router dovesse ad esempio avere un malfunzionamento o se la linea dovesse subire danni di qualsiasi tipo i router "a monte" se ne accorgerebbero e inoltrerebbero i dati attraverso altri router alternativi fino a destinazione.

2.5 risoluzione dei nomi: DNS

Quando si visualizza un sito su Internet in genere si "apre una pagina" in un browser web e si digita l'indirizzo web della pagina richiesta. Tuttavia un indirizzo di questo tipo può avere lunghezza variabile, e contenere molti tipi di caratteri (lettere maiuscole e minuscole, numeri, alcuni segni di punteggiatura). Esso va tradotto dalla sua forma alfanumerica alla sua forma digitale, codificata mediante indirizzo IP. Si parla quindi di "risoluzione dei nomi", ovvero di traduzione di un indirizzo alfanumerico nella sua corrispondente codifica numerica.

Ad esempio,

```
http://www.lugman.org
```

viene tradotto in questo modo: il web browser, mediante quella stringa, comunica che sta usando il "protocollo di trasferimento di ipertesti" (HyperText Transfer Protocol) e che sta richiedendo la pagina principale del sito www.lugman.org. Questa richiesta viene passata al nostro computer, che prende l'hostname (in questo caso www.lugman.org) e "chiede" ad un servizio apposito (chiamato "DNS", Domain Name Resolution) a che indirizzo corrisponde.

Il DNS è un protocollo di trasferimento di informazioni che di fatto si traduce in un software attivo su alcuni server, che ritornerà l'indirizzo IP di www.lugman.org. In pratica, mediante il comando

```
host www.lugman.org
```

si ottiene l'output

```
www.lugman.org has address 62.94.8.185
```

ora il computer potrà mettersi in comunicazione direttamente con quell'indirizzo IP per ottenere la pagina web richiesta.

Se non c'è nessun gateway configurato correttamente, o se non si connessi ad internet, non si avrà modo neanche di fare la richiesta al DNS; dopo un certo timeout il browser ritornerà un errore del tipo "l'host non può essere trovato" (host unreachable).

Il DNS dev'essere conosciuto a priori; nella maggior parte dei casi può essere fornito automaticamente al momento della connessione ad Internet dal provider, o impostato manualmente. Ad esempio, un DNS è l'indirizzo IP

```
151.1.1.1
```

che corrisponde a "dns.it.net" ovvero al dns di IT.net, provider nazionale di connettività. Un altro DNS utile è

```
212.216.172.62
```

ovvero "ns1.tin.it" - il DNS di Telecom Italia Net, un altro fornitore di connettività.

A questo punto dovrebbe risultare chiaro perchè si è invocato il comando "route" con il flag "-n" nella linea di comando; senza questo flag, il comando route avrebbe chiesto al DNS il nome di ogni indirizzo IP elencato, rallentando così la visualizzazione dell'output.

I DNS sotto la maggior parte degli UNIX sono impostati nel file

```
/etc/resolv.conf
```

che ha la seguente forma:

```
search home.lan
nameserver 217.141.110.142
nameserver 151.1.1.1
```

La stringa "search home.lan" significa "nel caso in cui venga richiesto un nome di host, prima si deve aggiungere il suffisso "home.lan' e poi si deve chiedere quel nome ai DNS specificati con "nameserver'.

Nota: mediante i DNS è possibile sia risalire all'indirizzo IP conoscendone il nome, sia al nome conoscendone l'indirizzo IP. Ovviamente non c'è un nome per ogni indirizzo, pertanto non è sempre possibile.

2.6 tracerouting dei pacchetti

Si è visto come un pacchetto debba passare attraverso alcuni gateway per raggiungere la destinazione. Per sapere quali sono questi gateway ed eventualmente per capire dove i pacchetti si fermano si usa il comando

```
traceroute -n
```

dove il flag "-n' ha lo stesso scopo che ha per il comando "route": non richiedere la risoluzione dei nomi. Ad esempio,

```
traceroute -n www.lugman.org
```

restituirà un output simile a questo

```
traceroute to www.lugman.org (62.94.8.185), 30 hops max, 38 byte packets
 1  192.168.2.1  0.610 ms  0.486 ms  0.505 ms
 2  192.168.100.1  155.384 ms  180.925 ms  173.967 ms
 3  217.141.110.199  82.752 ms  63.607 ms  181.068 ms
 4  151.99.99.97  155.994 ms  206.654 ms  265.828 ms
 5  151.99.75.213  70.187 ms  69.190 ms  121.758 ms
 6  151.99.98.226  156.518 ms  159.412 ms  64.659 ms
```

```
 7  217.29.67.64  65.660 ms  76.663 ms  71.246 ms
 8  62.94.31.37  160.003 ms  83.735 ms  142.399 ms
 9  62.94.0.125  136.082 ms  168.809 ms  105.807 ms
10  62.94.8.185  120.354 ms  261.868 ms  279.433 ms
```

Qui si può vedere vedere come il primo HOP (ovvero host di passaggio) è proprio il gateway connesso ad internet all'indirizzo 192.168.2.1. Il successivo (192.168.100.1) è il collegamento tra il gateway (modem ADSL) e la centrale telecom. Dal terzo in poi siamo già su internet, e l'ultimo è proprio l'IP a cui risponde il server web "www.lugman.org", ovvero 62.94.8.185. Per ogni host vengono indicati i tempi di percorrenza totali, di andata e di ritorno in millisecondi.

La man page di traceroute spiega in dettaglio le opzioni di questo comando e le varie casistiche che si possono ottenere.

2.7 porte tcp/udp

Come si vedrà più avanti il protocollo IP si occupa soltanto di "mettere in comunicazione" più computer; ciò che trasporta dati veri e propri e si occupa del loro corretto recapito è il protocollo TCP (Transmission Control Protocol) mentre il protocollo UDP (User Datagram Protocol) si limita ad indirizzare la comunicazione verso un servizio specifico senza preoccuparsi dell'arrivo a destinazione delle informazioni. Il primo protocollo inoltre garantisce che il pacchetto venga ricevuto e che venga mantenuto un ordine coerente rispetto agli altri pacchetti inviati per quella comunicazione.

Entrambi i protocolli TCP e UDP utilizzano il concetto di "porta di comunicazione": ogni porta è un numero compreso tra 1 e 65535 (2 elevato alla 16ma potenza: la porta si identifica quindi con 2 bytes) e ad ogni porta può essere messo in ascolto un servizio preciso. Questi servizi (che vengono gestiti da software chiamati spesso "demoni") possono essere letteralmente di ogni tipo: streaming audio/video, web, POP3 (lettura e-mail da server), SMTP (invio e-mail), DNS (risoluzione dei nomi), SSH (console remota sicura), VNC (controllo remoto di una workstation), eccetera.

I principali servizi sono elencati sotto UNIX nel file /etc/services. Questo file è del tipo:

```
[...]
ftp             21/tcp
fsp             21/udp          fspd
ssh             22/tcp                          # SSH Remote Login Protocol
ssh             22/udp                          # SSH Remote Login Protocol
telnet          23/tcp
smtp            25/tcp          mail
[...]
```

ovvero la porta 21 individuata con protocollo TCP è assegnata di default al servizio "ftp" (per il trasferimento di files); alla porta 22 lo standard indica SSH, alla 23 telnet, alla 25 SMTP e via dicendo.

Attenzione - questa terna ordinata (nome,porta,protocollo) non è nient'altro che uno standard; nessuno vieta di mettere in ascolto il demone SSH sulla porta 80, o di aprire servizi nascosti su porte non standard. Per questo è

importante tenere in considerazione questo file unicamente come un elenco di riferimento al quale attingere per riconoscere porte e protocolli. Inoltre, come si vedrà in seguito, a questo file fanno riferimento molti programmi per associare nomi di servizi a porte. Ad esempio, per indicare ad un firewall di bloccare le connessioni al servizio SSH, si potrà specificare la porta (in questo caso 22) o il nome del servizio così com'è riportato in /etc/services.

2.8 netstat

C'è un comando utile per visualizzare lo stato attuale delle connessioni di rete; come per "route" e "traceroute", conviene usarlo nella forma:

```
netstat -n
```

in questa forma mostrerà sia le connessioni di tipo INET (ovvero le connessioni di rete) sia i "socket" attivi. I socket sono una sorta di "tubo" che mette in comunicazione due diversi processi. Essi prendono la forma di files speciali sul disco fisso, ad esempio:

```
ls -laF /tmp/.X11-unix/
total 8
drwxrwxrwt    2 root     root         4096 Nov 26 12:41 .
drwxrwxrwt    8 root     root         4096 Nov 26 19:33 ..
srwxrwxrwx    1 root     root            0 Nov 26 12:41 X0=
srwxrwxrwx    1 root     root            0 Nov 26 12:18 X64=
```

come si vede da questo listato, i files "X0" e "X64" vengono identificati come "socket" dalla lettera "s" all'inizio della loro riga corrispondente. Uno dei vantaggi di questa forma è che è possibile trasferire dati da programmi diversi che non hanno conoscenza l'uno dell'altro; il primo immetterà dati in questa socket, e il secondo li potrà leggere come se si trovasse all'altro capo di un tubo. In questo modo si sono messi in comunicazione due processi indipendentemente dal nome con cui vengono chiamati o dal PID (identificativo univoco del processo), che possono variare.

L'utilizzo di netstat più comune è comunque legato alle connessioni di rete (identificate con il tipo "INET"). Per visualizzarle useremo il comando

```
netstat -n --proto=inet
```

che restituirà un output simile a questo:

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address       Foreign Address  State
tcp        0      0 192.168.2.15:33080  192.168.2.2:22   ESTABLISHED
```

il cui significato è: c'è una connessione attiva dall'indirizzo locale 192.168.2.15 all'indirizzo 192.168.2.2. Essa ha origine dalla porta 33080 (mostrata dopo il simbolo di due punti in "Local Address") e come destinazione la porta 22 dell'host remoto. Inoltre, essa utilizza il protocollo tcp (come si vede dal primo campo) e ha stato "ESTABLISHED", ovvero è ancora in corso.

Si noti che questo comando mostra le connessioni effettuate dall'utente. Per un elenco di tutte le connessioni attive al momento e di tutte le porte in ascolto, si usa il comando

```
netstat -na --proto=inet
```

il flag "-a" in questo caso indica "All ports". L'output sarà del tipo:

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address       Foreign Address  State
tcp        0      0 0.0.0.0:901         0.0.0.0:*        LISTEN
tcp        0      0 0.0.0.0:21          0.0.0.0:*        LISTEN
tcp        0      0 0.0.0.0:22          0.0.0.0:*        LISTEN
[...]
tcp        1      1 192.168.2.2:4666    192.168.2.10:2051  LAST_ACK
tcp        0      0 192.168.2.2:4666    192.168.2.11:60213 TIME_WAIT
tcp        0      0 192.168.2.2:33080   192.168.2.2:22     ESTABLISHED
tcp        0      0 192.168.2.2:32771   157.27.240.10:993  ESTABLISHED
udp        0      0 192.168.2.2:137     0.0.0.0:*
udp        0      0 0.0.0.0:137         0.0.0.0:*
[...]
```

Si può vedere come la porta "901" sia in ascolto ("LISTEN") per tutti gli indirizzi (identificati con "0.0.0.0"), come due connessioni siano appena finite (identificate da "LAST_ACK" e da "TIME_WAIT"), e di come ci siano due connessioni in corso ("ESTABLISHED").

Inoltre il protocollo UDP non mostra lo stato della connessione. Come si è già visto, questo protocollo non si occupa di garantire l'arrivo a destinazione dei dati; quindi non potrà nemmeno sapere se la connessione si è interrotta o se è andata a buon fine. Per il protocollo UDP si vede solo la porta 137 in ascolto sia su tutti gli indirizzi possibili, sia sull'indirizzo specifico "192.168.2.15".

Ancor più utile è la forma

```
netstat -nap --proto=inet
```

che indica tutte le connessioni ("a") senza risolvere i nomi di dominio ("n") mostrando anche il PID del programma che gestisce quella connessione ("p"). L'output sarà della forma

```
tcp   0  0 192.168.2.15:33080  192.168.2.2:22  ESTABLISHED 835/ssh
```

la differenza con l'output precedente è la presenza del PID (Process IDentifier, il numero del processo in esecuzione in questo momento) unito al proprio nome. In questo caso il processo è "ssh" e ha PID 835.

Nota: la linea di comando può essere anche del tipo

```
netstat -n -a -p --proto=inet
```

specificando le singole opzioni precedute dal segno meno ("-"). La forma usata in precedenza permette di condensare queste opzioni in un'unica stringa per migliorarne la leggibilità.

2.9 il concetto di IP pubblico e IP privato. cenni di subnetting.

Come si è visto un indirizzo IP è composto da quattro ottetti (ovvero quattro bytes). In formato decimale ogni ottetto può avere valori compresi tra 0 e 255, inclusi gli estremi. Ogni IP pubblico viene assegnato da autorità competenti, mentre per fare esperimenti o per reti locali sono state riservate delle classi di indirizzi.

Esse vengono definite in questo modo (RFC 1918):

```
classe A: 10.0.0.0        -   10.255.255.255  (indicata anche come 10/8)
classe B: 172.16.0.0      -   172.31.255.255  (172.16/12)
classe C: 192.168.0.0     -   192.168.255.255 (192.168/16)
```

dove la notazione /n indica il numero di bit che identificano la rete. Ad esempio, un indirizzo IP del tipo

```
192.168.0.15
```

appartiene alla terza classe (classe C).

Mediante la coppia indirizzo IP / netmask si è in grado di capire la rete di appartenenza. Infatti, se per l'indirizzo precedente fosse indicata una netmask di

```
255.255.255.0
```

se ne deduce che la rete di appartenenza avrà indirizzi da 192.168.0.1 a 192.168.0.254. Per convenzione, il primo indirizzo (.0) indica la rete e l'ultimo (.255) indica il broadcast, ovvero l'indirizzo al quale "rispondono" tutti gli host. Per questa ragione gli IP disponibili partono da 0.1 (anzichè da 0.0) e terminano a 0.254 (anzichè a 0.255).

Non è necessario che una rete abbia netmask del tipo 255.255.255.0 o 255.255.0.0 o 255.0.0.0. Si possono anche avere netmask come 255.255.255.240, oppure notazioni del tipo 192.168.1/28. La notazione "/28" indica il numero di bit che nell'indirizzo indicano la rete, mentre gli altri bit sono riservati per indicare l'host. Data una netmask e un indirizzo di rete si può calcolare a mente o mediante tool specifici (come ipcalc) i corrispondenti indirizzi di broadcast e IP disponibili.

Un esempio (sulla destra si vedono le rappresentazioni binarie):

```
ipcalc 192.168.1.0/28

Address:   192.168.1.0           11000000.10101000.00000001.0000 0000
Netmask:   255.255.255.240 = 28  11111111.11111111.11111111.1111 0000
Wildcard:  0.0.0.15              00000000.00000000.00000000.0000 1111
=>
Network:   192.168.1.0/28        11000000.10101000.00000001.0000 0000 (Class C)
Broadcast: 192.168.1.15          11000000.10101000.00000001.0000 1111
HostMin:   192.168.1.1           11000000.10101000.00000001.0000 0001
HostMax:   192.168.1.14          11000000.10101000.00000001.0000 1110
Hosts/Net: 14                    (Private Internet RFC 1918)
```

Da questo output si vede come gli indirizzi disponibili siano in tutto 14, da 192.168.1.1 a 192.168.1.14.

Nota: con l'indirizzo

```
127.0.0.1
```

si indica sempre l'host locale, indipendentemente dal fatto che possa non esserci una scheda di rete installata. Mediante quell'IP molti processi riescono a comunicare in "loopback".

2.10 schema ISO/OSI

Con questa sigla si intende un sistema a livelli (layer) sviluppato dalla ISO (International Standard Organization), detto modello OSI (Open Systems Interconnect). Il modello OSI descrive semplicemente i livelli di astrazione ottimali mediante i quali si possono definire i diversi compiti che deve svolgere ogni componente di rete.

Per "componente" in questo caso si intende tutto ciò che contribuisce alla comunicazione tra le diverse entità di una rete: dalle singole schede ai driver del sistema operativo, dai router ai firewall ai software per gli utenti.

I sette livelli sono così indicati:

1. Application Layer (Top Layer) A questo livello corrispondono tutti i programmi che veicolano informazioni da e verso l'operatore umano. Appartengono a questa categoria i web browser, i server web (intesi come pacchetto software), eccetera.

2. Presentation Layer Questo livello gestisce la security e gli scambi di informazioni tra le applicazioni e il sistema operativo del computer. Un esempio in ambiente Microsoft possono essere le shares (condivisioni) di rete mediante protocollo SMB, oppure in ambiente Unix i concetti di "mount NFS" (ovvero partizioni montate da server remoti).

3. Session Layer Questo livello identifica mittente e destinatario delle informazioni e il corretto trasporto dei dati. Un esempio è il protocollo TCP (legato al concetto di "porta" alla quale risponde un servizio).

4. Transport Layer Al quarto livello si trova il protocollo IP, che gestisce i percorsi dei singoli pacchetti dall'host sorgente fino all'host destinazione. IP si preoccupa inoltre di identificare in maniera univoca i componenti di una rete (mediante l'indirizzo IP).

5. Network Layer Il Network Layer gestisce le informazioni che vengono passate dai livelli più bassi verso i livelli più alti: ad esempio, si occupa dell'indirizzo fisico (hardware address, detto anche MAC address) dei dispositivi connessi e cura la frammentazione delle informazioni in piccole "celle" facilmente trasportabili dai livelli sottostanti. Ogni dato superiore ad una certa dimensione (dell'ordine di KB, migliaia di bytes) viene frammentato in piccoli pezzi che vengono inviati separatamente attraverso il mezzo fisico di trasmissione, e riassemblati alla destinazione. Il Network Layer si occupa appunto di dimensionare questi "pezzi" a seconda del mezzo che li dovrà trasportare.

6. Data Link Layer A questo livello corrisponde, ad esempio, il firmware della scheda di rete che trasporta i dati. Esso converte le informazioni che giungono dal Network Layer (livello superiore) e le passa al livello sottostante sotto forma di bit; inoltre si occupa di "ricostruire" le informazioni che giungono dalla rete, di scartare tutti quei pacchetti che arrivano corrotti e di chiedere la loro ritrasmissione.

7. Physical Layer (Bottom Layer) L'ultimo livello si occupa della trasmissione delle informazioni vere e proprie. Può convertirle in onde elettromagnetiche, flussi di luce, differenze di potenziale e via dicendo. A questo livello i bit sono soltanto bit, indistinguibili gli uni dagli altri se non conoscendo il protocollo con il quale le informazioni sono state codificate.

Questo schema è ampiamente discusso, criticato e osannato da tempo. Anche se probabilmente non è il modo migliore per astrarre le trasmissioni di rete esso funziona più che discretamente e ha permesso di giungere allo stato attuale della tecnologia mantenendo coerenza e semplicità nei suoi singoli componenti.

Il punto di forza di un'astrazione come quella del modello OSI è la sua interdipendenza tra livelli; l'application layer non deve sapere su che mezzo verranno trasmessi i dati, ma è sufficiente che sappia come interpretare ciò che gli viene passato dal Presentation Layer. In pratica, il browser non deve sapere se si sta guardando una pagina web via modem analogico, adsl, satellite o fibra ottica. È inutile e porterebbe ad una complessità di sviluppo troppo elevata. Esso deve unicamente saper interpretare le stringhe HTML che gli vengono fornite dai livelli sottostanti, e mostrarle correttamente all'utente.

2.11 connessione a tre vie (three-way handshaking) su protocollo TCP

È necessario a questo punto specificare brevemente come avviene una connessione mediante protocollo TCP. Innanzitutto questo protocollo si occupa di trasmettere dati a destinazione e di verificare che siano arrivati. Esso lavora mediante un meccanismo di controllo a tre vie (detto three-way handshaking) in questo modo:

1. il mittente invia una richiesta di inizio connessione al destinatario

2. il destinatario risponde con un "ok" (in inglese Acknowledged) o con un rifiuto

3. il mittente conferma al destinatario che ha ricevuto la risposta e inizia a trasmettere

il funzionamento preciso è di questo tipo:

1. il mittente invia un pacchetto TCP con un bit dell'header (intestazione) impostato a 1. Questo bit è detto "SYN' bit, e indica la richiesta di inizio connessione. Inoltre, questo pacchetto contiene un numero casuale per identificare a che connessione i due computer stanno facendo riferimento (detti "sequence" e "acknowledgment numbers", numeri di sequenza).

2. il destinatario risponde con un pacchetto TCP che ha i bit SYN e ACK settati a 1 in caso sia disposto a ricevere la connessione; in caso contrario risponde con un pacchetto con solo il bit RST ("Reset") settato a 1. Il numero casuale viene incrementato di un intervallo noto.

3. il mittente, in caso di risposta positiva, inizia la trasmissione impostando l'ACK bit a 1.

Questo processo è importante (come si vedrà in seguito) per separare trasmissioni di dati già iniziate e trasmissioni che stanno iniziando, e per capire da che parte del firewall queste trasmissioni hanno avuto inizio.

2.12 comandi specifici delle principali distribuzioni GNU/Linux: Debian, RedHat, Mandrake, Suse, Gentoo.

In questa sezione introdurremo i files e i comandi per gestire la rete di alcune distribuzioni tra le più note. La maggior parte delle distribuzioni "end-user" (Mandrake, Redhat, SuSe) hanno basato la loro politica di configurazione su tool grafici usabili mediante le interfacce grafiche installate di default, quali gnome o kde; tuttavia in questa sede si vedranno, dove possibile, le utility usabili da terminale proprio per l'esigenza di poter far manutenzione su un server soprattutto da postazioni remote.

1. Debian gestisce la rete nella directory

   `/etc/network`

   dove i file utili sono

   `/etc/network/interfaces`
   configurazione delle interfacce di rete

   `/etc/network/options`
   opzioni di rete: forwarding, syncookies

   in questa directory è anche possibile gestire gli script da avviare quando la connessione è stabilita, come firewall, sincronizzazione dell'orologio via internet, eccetera.

2. Per configurare la rete in RedHat c'è il comando

   `/usr/sbin/netconfig`

   mentre i files relativi si trovano in

   `/etc/sysconfig/networking`

   `/etc/sysconfig/network-scripts`

tuttavia redhat raccomanda di non modificare manualmente questi file e di lasciar fare all'utility redhat-config-network, disponibile via interfaccia grafica.

3. Mandrake, basata su RedHat, ha anch'essa gli script e i files relativi in

   `/etc/sysconfig/networking`

   `/etc/sysconfig/network-scripts`

   mentre per la configurazione di rete si affida soprattutto alle utility grafiche (centro di controllo Mandrake)

4. Anche SuSe si appoggia a

   `/etc/sysconfig`

   oltre al suo ottimo tool di gestione YaST, disponibile sia in modalità grafica che sotto console.

   Infine, Gentoo si appoggia a

   `net-setup`

   `adsl-setup`

   per la configurazione assistita della rete. Nel caso in cui il riconoscimento automatico fallisca, i file di configurazione si trovano in

   `/etc/conf.d/`

3 Firewalling di base

3.1 concetti

Il concetto di firewall si può riassumere brevemente ed in maniera poco corretta come "decidere chi può andare dove e chi non può andare". Poichè in ambito di reti il "chi" è identificato da un indirizzo IP, il più delle volte si intende come firewall un controllo degli accessi basato sull'IP di provenienza. Vi sono comunque tre macro-categorie di accesso: a livello di applicazione (ad esempio, far accedere ad un servizio un certo utente e bloccare altri utenti); a livello di servizio (ad esempio, lasciare al servizio stesso il compito di decidere da chi ricevere connessioni); a livello di IP (ovvero decidere in base a parametri quali la provenienza della connessione chi può accedere ad un certo servizio).

3.2 esempio: controllo degli accessi a livello di applicazione

Il controllo degli accessi a livello di applicazione esula dagli scopi di questo documento; basterà specificare che viene configurato in maniera strettamente dipendente dal servizio, basandosi su schemi di autenticazione differenti. Ad esempio si potrà usare un server di autenticazione che gestisce le risorse alle quali gli utenti della rete possono accedere e al quale i vari server che forniscono servizi fanno richiesta per autenticare una persona. In ambiente UNIX sarebbe un server di tipo "LDAP" in ambiente Microsoft sarebbe un "Controller di Dominio". Un altro esempio di controllo degli accessi a livello di applicazione potrebbe essere un mail server che fornisce servizio SMTP (posta in uscita) solo agli utenti che si sono già identificati con successo (ad esempio, si può inviare posta ad utenti esterni ad un certo dominio solo dopo aver effettuato una login POP3 valida).

3.3 esempio: controllo degli accessi a livello di servizio

In questo caso si parla di gestione degli accessi host-based; un esempio in ambiente UNIX è il file

```
/etc/hosts.allow
```

e il file

```
/etc/hosts.deny
```

Questi file hanno una forma del tipo (per hosts.allow):

```
ALL: LOCAL
smtp: esempio.com
```

ovvero "fornisci accesso di tipo SMTP solo all'hostname che si risolve come "esempio.com', e fornisci tutti gli altri servizi solo alla macchina stessa". Tutto è ben spiegato nella manpage di hosts_access e di hosts_options.

Un ulteriore esempio potrebbe essere un servizio di http proxy che viene fornito solo a certi utenti, magari integrando il controllo sugli IP con un meccanismo di accesso ad autenticazione centrale, per poter usare il proxy solo dopo aver fornito una login e una password.

3.4 esempio: controllo degli accessi a livello di IP

Il controllo degli accessi di questo livello è tra i più diffusi sulla rete perchè permette di controllare mediante un singolo firewall la protezione di un'intera LAN, e consente un monitoraggio molto dettagliato di quanto avviene tra il mondo esterno e il mondo interno. Un esempio di questo "firewall" è "iptables", un programma che nel kernel 2.4 è il meccanismo di firewalling per antonomasia. Nel kernel 2.2 aveva un altro nome (ipchains) e potenzialità più limitate. Da quando è stato introdotto iptables si è arrivati ad avere in ambiente Open Source un firewall in competizione con alcuni tra i più conosciuti firewall commerciali.

Prima di iniziare con l'analisi delle possibilità di questo strumento, è necessario specificare che iptables non è un firewall a livello di applicazione; esso si occupa unicamente di gestire le connessioni in base all'indirizzo IP.

Se un firewall consente gli accessi dall'esterno ad un webserver interno alla rete, ed il webserver risulta vulnerabile ad un attacco, la colpa non sarà del firewall o del suo amministratore ma di chi gestiva il server violato. Per questo motivo è molto importante non solo cercare di prevedere tutte le casistiche, ma anche mantenere il livello di sicurezza alto in ogni punto vulnerabile ad un attacco.

A titolo di informazione, si ricordi che un firewall come iptables viene definito anche "stateful", mentre una semplice gestione degli accessi (ad esempio con `/etc/hosts.allow`) viene chiamata "stateless". Con il termine "stateful" si intende il mantenimento della traccia delle connessioni (connection tracking); ogni pacchetto non solo viene valutato in base alla provenienza, ma anche in base all'appartenenza ad una connessione già stabilita. Un firewall stateful quindi occupa meno risorse, non dovendosi preoccupare di tutte le connessioni già autorizzate. Al contrario, un firewall stateless ha delle regole precise che lo portano a controllare ogni singolo pacchetto di dati in arrivo.

3.5 iptables: concetti di base

Un firewall basato su iptables altro non è che un elenco nel quale viene indicata la strada che un pacchetto può o non può intraprendere. Esso viene letto dalla prima riga all'ultima; se all'inizio avremo una regola del tipo

```
iptables -A INPUT -p TCP -s 192.168.2.15 -d 192.168.2.2 --dport 22
-j ACCEPT
```

il firewall si comporterà nel seguente modo: appena arriva un pacchetto proveniente da 192.168.2.15 (source, ovvero "-s'), destinato all'IP 192.168.2.2 (destination, "-d') di tipo TCP (protocollo, "-p') indirizzato alla porta 22 (destination port, "–dport 22') esso verrà accettato ('ACCEPT') e il firewall si disinteresserà del destino di questo pacchetto. Può darsi che alla porta 22 di 192.168.2.2 non risponda nessun servizio, ma questo non è rilevante agli scopi del firewall stesso.

Questo primo esempio su iptables mostra come di fatto un firewall non sia altro che uno script in cui sono elencate, riga per riga, le varie destinazioni che un pacchetto può raggiungere. Una destinazione può essere sia un "target finale" (come "ACCEPT": il pacchetto viene accettato), sia un'altra riga del firewall nella quale controllare

Ci sono due modi di impostare un firewall: il primo modo indica *cosa è proibito fare*, bloccando ciò che non rispetta questa regola e lasciando passare tutto il resto. Il secondo modo indica *cosa è permesso fare* e blocca tutto quello che non è previsto. Entrambi i modi hanno pro e contro; un firewall restrittivo è più sicuro ma di più difficile progettazione; un firewall più permissivo consente di studiare meno la situazione, ma ha il rischio di "lasciar passare" pacchetti di cui si preferirebbe fare a meno.

3.6 iptables: percorso dei pacchetti ed esempio di port forwarding

Il percorso dei pacchetti dall'interfaccia di rete di ingresso all'interfaccia di rete di uscita segue un tragitto predefinito anche all'interno della memoria del computer stesso, dove viene analizzato e modificato all'occorrenza. Si è parlato di due interfacce di rete perchè un firewall in genere viene immaginato come una "barriera" fisica tra ciò che entra e ciò che esce.

Il percorso dei pacchetti quindi è diviso in tre categorie:

1. pacchetti diretti verso il firewall stesso
2. pacchetti uscenti dal firewall stesso
3. pacchetti provenienti o diretti da/a macchine all'interno della rete che il firewall stesso ha il compito di proteggere.

Ognuna di queste categorie ha un nome all'interno di iptables:

1. INPUT (diretti al firewall)
2. OUTPUT (uscenti dal firewall)
3. FORWARD (che attraversano il firewall)

La potenza di iptables consiste nell'utilizzare concetti astratti come "tabelle" per raggruppare gruppi di categorie differenti.

Negli esempi che seguono verranno mostrati i concetti appena introdotti. Più avanti si troverà un firewall completo. Esso sarà utile non solo per capire i concetti di "table" (tabella) e per vedere il funzionamento di INPUT, OUTPUT e FORWARD, ma verrà anche tenuto come riferimento nel resto del documento.

Le tre tabelle principali di iptables sono

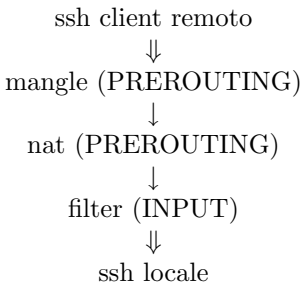
- *filter*: questa tabella di default raccoglie le tre categorie principali, ovvero
  - INPUT
  - OUTPUT
  - FORWARD
- *nat*: questa tabella gestisce tutti i pacchetti che creano nuove connessioni da e verso l'esterno. Ad esempio per effettuare un port forwarding (ovvero per indirizzare ad una porta di un server una connessione verso la macchina locale) si utilizzerà la table "nat"; essa consiste di tre categorie:
  - PREROUTING per gestire i pacchetti appena arrivano
  - OUTPUT per gestire pacchetti i generati localmente prima di instradarli
  - POSTROUTING per gestire i pacchetti in uscita
- *mangle*: questa particolare tabella è utilizzata per alterare i singoli pacchetti, ad esempio per modificare il campo TOS (Type Of Service) di un pacchetto IP, per effettuare un MARK su una particolare classe di pacchetti, eccetera. Essa consiste (a partire dal kernel 2.4.18) di:

- PREROUTING per gestire i pacchetti che stanno arrivando
- INPUT per gestire i pacchetti in entrata nel firewall stesso
- FORWARD per gestire i pacchetti in transito nel firewall
- OUTPUT per gestire i pacchetti in uscita dal firewall stesso ed infine
- POSTROUTING per gestire i pacchetti uscenti

Non è possibile in questa sede spiegare il funzionamento teorico di questa particolare table; maggiori informazioni sono reperibili alla pagina http://www.netfilter.org.

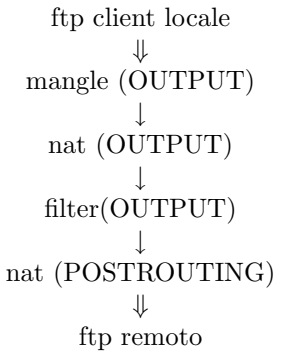
Alcuni percorsi classici all'interno di un firewall possono essere:

1. connessione ssh al firewall locale: i pacchetti destinati alla porta ssh (tcp/22) locale avranno il seguente percorso



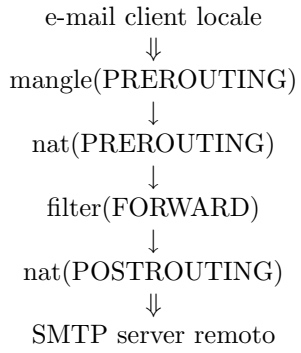
ovvero: un pacchetto entrante prenderà la categoria "PREROUTING" della table "mangle", poi attraverserà la "PREROUTING' della table "nat', ed infine arriverà alla "INPUT' della table "filter'; dopo questo percorso arriverà al servizio ssh sul computer locale.

2. connessione dal firewall locale a un ftp remoto:



All'inizio del percorso si troverà il client ftp, mentre all'uscita risponderà il server.

3. connessione ad un servizio (es: posta elettronica) presente all'interno della LAN protetta dal firewall:



In questo caso come si può notare i pacchetti attraversano fisicamente la macchina, entrando da un'interfaccia per poi passare a PREROUTING di "mangle' e "nat', attraversando la "FORWARD' di "filter' ed infine passando attraverso la "POSTROUTING' di "nat'.

In ogni esempio il pacchetto attraversa prima la "mangle' e poi la "nat' per un motivo preciso: mediante il "mark' si possono "marcare" pacchetti sul firewall per monitorarli (log) oppure per aumentarne la priorità (TOS, Type Of Service). Una volta marcati possono essere gestiti attraverso la PREROUTING di "nat', ovvero essere "girati' verso l'host di destinazione.

Tutte queste differenze nel percorso sono necessarie perchè ad ogni categoria di ciascuna table possono essere applicate solo certi tipi di manipolazioni. Ad esempio, per un port forwarding si userà una riga (che in fin dei conti è un semplice comando da shell) del tipo

```
iptables -t nat -A PREROUTING -p TCP -i eth1 --dport 5900 -j DNAT
--to-destination 192.168.2.15
```

ovvero: alla categoria "PREROUTING' della tabella "nat' ('-t nat') aggiungi ('-A') la seguente regola: se il pacchetto che ha raggiunto questo firewall ha come destinazione la porta 5900 ('–dport 5900') e proviene dall'interfaccia di rete eth1 ('-i eth1'), allora esegui un DNAT ('-j DNAT': destination nat, ovvero "gira' le connessioni alla porta 5900) alla destinazione 192.168.2.15 ('–to-destination'). Una regola come questa non avrebbe potuto essere applicata, ad esempio, alla "FORWARD' di "filter', nè alla "POSTROUTING' di "nat'; infatti il port forwarding va effettuato prima che il pacchetto raggiunga completamente il nostro firewall.

Questa rule permette ai pacchetti destinati alla porta 5900 di venir "girati' ad un altro computer che fornirà un particolare servizio (in questo caso VNC); tuttavia essa permette a chiunque di connettersi alla 5900. Ora si dovrà definire "chi può entrare" mediante una rule del tipo

```
iptables -A FORWARD -i eth1 -s <IP sorgente> -d <IP destinazione> -j
ACCEPT
```

ovvero: aggiungi ('-A') alla categoria FORWARD (è sottintesa la table "filter') la seguente regola: permetti ('-j ACCEPT') ai pacchetti provenienti dall'interfaccia di rete eth1 ('-i eth1') che hanno indirizzo di provenienza ¡IP sorgente¿ e che

sono destinati all'indirizzo locale ¡IP destinazione¿ di entrare e di passare alla tabella successiva.

In pratica con la prima regola (-t nat -A PREROUTING...) apriamo un corridoio verso una destinazione interna. Con la seconda regola effettivamente apriamo la porta di quel corridoio a chi si vorrà dare accesso.

È possibile raffinare la prima regola mostrata inserendo anche un controllo sulla sorgente del pacchetto del tipo -s ¡IP sorgente¿. Tutto questo verrà spiegato più avanti. Per ora è sufficiente sapere che con quelle due regole permettiamo a chi proviene da "IP sorgente" di connettersi alla porta 5900 di 192.168.2.15.

Nota: è importante notare che l'IP di destinazione sia locale e non pubblico. Infatti potremmo essere collegati via modem o con un IP dinamico e non sapere a che IP risponde l'interfaccia "esterna"; quindi è sufficiente specificare la destinazione interna e lasciare che sia il firewall a capire che i pacchetti destinati alla nostra porta "x' vanno rigirati alla destinazione interna, sulla stessa porta "x'. Il firewall comunque non "indovina", ma si basa sulla prima regola ('–to-destination') per capire a chi indirizzare i pacchetti che ci interessano.

3.7 iptables: firewalling di base. NAT/masquerading

L'esempio visto in precedenza non appartiene a quello che viene inteso come "firewall semplice". Nelle prossime sezioni verrà mostrato il funzionamento di base di un firewall, tralasciando momentaneamente gli argomenti dettagliati sulle varie tables per poi tornarci in seguito.

Un firewall di base deve poter svolgere due compiti:

1. permettere a chi è "dentro" di poter andare "fuori"
2. permettere a chi è "fuori" di non poter andare "dentro".

Per gestire un lavoro del genere non è necessario avere nè grosse capacità a livello di hardware (un vecchio computer va benissimo) nè grosse conoscenze tecniche; basta una semplice riga con iptables:

```
iptables -t nat -A POSTROUTING -o <interfaccia di uscita> -j MASQUERADE
```

con questo comando si chiede ad iptables di gestire i pacchetti in uscita dal nostro computer ('-t nat') che sono già stati instradati ('-A POSTROUTING') attraverso l'interfaccia ¡interfaccia di uscita¿ ('-o') e di farli comparire come se provenissero dal nostro indirizzo IP ('-j MASQUERADE'). È importante abilitare il forwarding dei pacchetti negli script di avvio della distribuzione Linux che si sta usando o a mano mediante il comando

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

che imposta al valore numerico "1" il file ip_forward nella directory /proc/sys/net/ipv4. Se questo file contenesse il valore "0" il kernel si rifiuterebbe di inoltrare pacchetti e non si potrebbero stabilire connessioni al di fuori di quelle provenienti direttamente dal firewall. Ogni distribuzione GNU/Linux va ad impostare questo valore all'avvio se viene richiesto da un file preciso. A titolo di esempio, Debian controlla il parametro

`ip_forward`
(dev'essere impostato a "yes")
nel file

`/etc/network/options`

Questa tecnica che permette di "mascherare" un'intera LAN alle spalle del firewall con un unico indirizzo IP prende il nome di "masquerading" o "NAT" (Network Address Translation). Supponiamo di avere il seguente scenario

$$\text{LAN} \iff \text{gateway} \iff \text{internet}$$

l'utente al computer "X" della LAN deve comunicare in maniera bidirezionale con un server su Internet (ovvero deve poter inviare dati e ricevere risposte). L'host "X" non ha un indirizzo IP pubblico al quale il server su Internet può rispondere, ma ha solo un IP privato all'interno della rete. Di conseguenza il gateway dovrà effettuare un'operazione di NATting sulla connessione uscente, sostituendo all'IP privato dell'utente il proprio IP pubblico. Inoltre dovrà mantenere una traccia di questa connessione in modo da mandare a "X" la risposta proveniente dal server, se essa giungerà entro un tempo limite.

3.8 iptables: gestione delle policy.

Con iptables è possibile impostare un comportamento di default che viene tenuto dalle categorie (dette anche "chains') di una particolare tabella. Il comando

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

dice al firewall di far cadere ('DROP') tutti i pacchetti che arrivano in fondo al set di regole per quella categoria. Si ricordi che le regole vengono lette riga per riga, e se la riga descrive il pacchetto in arrivo verrà intrapresa un'azione di conseguenza. Se nessuna riga dovesse corrispondere al pacchetto in arrivo, il firewall userà la policy di default per decidere che fine far fare al pacchetto.

Supponiamo che si sia impostato un port forwarding nella chain FORWARD per certi IP. Se si imposta la policy di FORWARD a DROP, i pacchetti che non corrispondono agli IP permessi nelle regole raggiungeranno il "fondo" della catena e verranno lasciati cadere.

Nota: è importante impostare delle buone regole di uscita ('OUTPUT') per poter comunicare con l'esterno.

3.9 iptables: sopravvivenza (cosa fare in caso di emergenza). Flushing e cancellazione delle tabelle. Show rules.

In caso di emergenza si possono pulire le rules con il comando

```
iptables --flush
```

abbreviato anche con l'opzione `-F`. Esso riporta tutte le rules ad empty, ma mantiene le policy allo stato attuale e non cancella le tables aggiunte. Esse vanno cancellate manualmente specificando il nome,con il comando

```
iptables -X <nome tabella>
```

ATTENZIONE: nel caso in cui si stia facendo manutenzione su un firewall da console remota, prima di fare un flush di tutte le rules è importante riportare le policy ad ACCEPT. In caso contrario si resterebbe "tagliati fuori", e l'unica soluzione sarebbe quella di collegare monitor e tastiera ed accedere in locale.

A questo proposito c'è un esempio più avanti nello shell script; esso permette di far ripartire un firewall azzerando tutte le regole e reimpostandole da capo senza pericolo di disconnessioni.

3.10 iptables: aggiunta e cancellazione di rules precise.

Una volta definite nell'analisi le regole che si intendono far seguire ai pacchetti in arrivo, in uscita e in transito si può iniziare a scriverle nel firewall una alla volta. Per aggiungere una regole il comando è

```
iptables -A <nome della chain> <parametri> -j <destinazione>
```

ad esempio:

```
iptables -t nat -A POSTROUTING -o eth1 -p TCP -s !  192.168.0.5 -d
0/0 --dport 25 -j DROP
```

che viene interpretato come: aggiungi ('-A') alla tabella "nat' ('-t nat') la seguente regola ai pacchetti che stanno uscendo dall'interfaccia eth1 ('-o eth1'): se il protocollo è TCP ('-p TCP') e NON provengono dall'ip 192.168.0.5 ('-s ! 192.168.0.5'),qualsiasi destinazione abbiano ('-d 0/0') e se sono indirizzati ad un mail server SMTP ('–dport 25'), allora falli cadere ('-j DROP').

La precedente regola è utile per bloccare connessioni verso server di posta elettronica esterni alla rete da parte eei computer in LAN, eccetto quelle provenienti dal mail server interno. In questo modo quindi si riescono a bloccare molti virus che si propagano attraverso la posta elettronica. Installando un antivirus sul mail server infatti esso bloccherà i virus in uscita, e bloccando l'accesso ai mail server esterni i computer infetti non potranno propagare il virus.

Per cancellare una rule è sufficiente mettere l'opzione -D al posto di -A, in questo modo:

```
iptables -t nat -D POSTROUTING -o eth1 -p TCP -s !  192.168.0.5 -d
0/0 --dport 25 -j DROP
```

facendo attenzione che i parametri siano gli stessi di quelli inseriti durante l'aggiunta della rule.

Per vedere esattamente una rule com'è definita, si usa il comando

```
iptables -L -n -v
```

il quale elenca ('-L') tutte le connessioni in modo dettagliato ('-v') senza cercare di risolvere gli indirizzi IP in nomi ('-n').

Si noti inoltre che il precedente comando non visualizza le rules delle singole tabelle. Per questo è necessario specificarla in questo modo:

```
iptables -L -n -v -t <nome tabella>
```

per ogni tabella presente nel firewall.

Nota: è possibile cancellare anche righe specifiche; ad esempio, se con il comando "iptables -L -n -v -t nat' otteniamo un output nel quale la prima riga è :

```
pkts bytes target  prot opt in   out    source        destination
21  1008 DROP      tcp  --  *    eth1   !192.168.0.6  0.0.0.0/0   tcp dpt:25
```

essa potrà essere rimossa dal firewall con

```
iptables -t nat -D POSTROUTING 1
```

ovvero "cancella la prima riga ('1') della POSTROUTING chain della tabella nat".

3.11 iptables: esempio di firewall di base. non rispondere ai ping.

Anche se sconsigliato dagli RFC, a volte può essere utile decidere di far "sparire" da un ping la propria macchina. Con iptables è semplice:

```
iptables -A INPUT -p icmp -j DROP
```

dove con "-p' si intende il protocollo ICMP (che gestisce i ping, i traceroute, eccetera). Attenzione però che non sarà possibile neanche ricevere risposte ai ping inviati dalla macchina stessa, e molti messaggi di errore (destinazione non raggiungibile, servizio non disponibile sulla porta richiesta, ...) non verranno ricevuti.

3.12 iptables: esempio di firewall di base. bloccare le connessioni a una porta.

Ci sono due modi per bloccare le connessioni ad una porta: con il target DROP e con REJECT. Ad esempio:

```
iptables -t nat -A POSTROUTING -o eth1 -p TCP -s 0/0 -d 0/0 --dport
1214 -j DROP
```

ovvero: blocca le connessioni in uscita ('-o') verso la porta 1214/TCP (in questo caso è una delle porte di connessione del sistema di file sharing "kazaa").

L'uso di DROP e REJECT è differente: con REJECT viene mandato un pacchetto di tipo ICMP all'host richiedente una connessione, informandolo che non è possibile stabilire una comunicazione con la porta richiesta. Con DROP

si “lasciano cadere” i pacchetti disinteressandosi della loro sorte, e l’host richiedente resterà in attesa per un certo periodo di tempo senza poter sapere se c’è o meno “qualcosa” a rispondere al servizio.

3.13 iptables: esempio di firewall di base. Policy restrittiva, cosa lasciar passare per uscire.

In questa sezione verrà mostrato uno script per BASH per attivare o disattivare un firewall di medie dimensioni e complessità. In questo esempio si suppone che tutti i computer in NAT dietro questa macchina debbano poter uscire tranquillamente su Internet, e che dall’esterno non sia necessario accettare connessioni salvo alcune (ad esempio ssh) sulle quali applicare altri meccanismi di controllo (password, certificati a chiave pubblica, e via dicendo). Si vuole inoltre che un computer interno sia raggiungibile mediante port forwarding da un indirizzo IP statico conosciuto a priori.

Il carattere “#” indica un commento e tutto ciò che segue questo simbolo viene ignorato dalla shell. Il carattere “\” indica un “a-capo” e significa che le righe successive andrebbero di seguito alla riga attuale.

```
--- begin script ---

#!/bin/sh
# questo e‘uno script BASH per un firewall minimo.
#
# (c) lorenzo@muug.it
# this script is available under the GPL public license.
#
# prima definiamo alcune variabili d’ambiente
#
IPTABLES="/sbin/iptables"

# l’interfaccia connessa ad internet
# se dovessimo cambiare tipo di connessione ad internet, ovvero mediante
# modem analogico o ISDN, sara‘ sufficiente cambiare questa variabile
# e rilanciare lo script
inet_iface="eth1"

# l’interfaccia connessa alla rete locale
lan_iface="eth0"

# l’indirizzo dell’interfaccia connessa alla rete locale
lan_addr="192.168.0.1"

# l’indirizzo di un computer con un server VNC al quale vogliamo poterci
# connettere dall’esterno
vnc_lan="192.168.0.9"

# l’indirizzo della rete esterna alla quale e‘ permesso di connettersi
# via VNC
```

```
vnc_ok="<inserire l'indirizzo>"

### begin functions

# questa funzione lancia i comandi per abilitare il firewall
enable_firewall() {

# abilita il NAT nel kernel; sempre meglio essere sicuri
echo 1 > /proc/sys/net/ipv4/ip_forward

###
# iniziamo a lavorare con iptables
#
# definiamo le policy di default in maniera restrittiva
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# in aggiunta, definiamo anche altre due chains utili:
# questa chain contiene le connessioni ``consentite'' per fare dei controlli
# comuni sui pacchetti in arrivo
$IPTABLES -N allowed

# questa chain raggruppa i pacchetti ICMP
$IPTABLES -N icmp_packets

# le prossime righe senza commento iniziale permettono di effettuare
# debugging per vedere dove i pacchetti arrivano
# -- IMPORTANTE: per vedere i log, usare il comando dmesg

#$IPTABLES -t nat -A PREROUTING -j LOG --log-level DEBUG --log-prefix \
"debugging nat PREROUTING: "
#$IPTABLES -A FORWARD -j LOG --log-level DEBUG --log-prefix ``debugging \
filter FORWARD: "
#$IPTABLES -A INPUT -i ! lo -j LOG --log-level DEBUG --log-prefix \
"debugging filter INPUT: "
#$IPTABLES -A OUTPUT -j LOG --log-level DEBUG --log-prefix ``debugging \
filter OUTPUT: "
#$IPTABLES -t nat -A POSTROUTING -j LOG --log-level DEBUG --log-prefix \
"debugging nat POSTROUTING: "

# ora il NAT: lascia uscire i pacchetti dalla LAN
$IPTABLES -t nat -A POSTROUTING -o $inet_iface -j MASQUERADE

# lasciamo uscire i pacchetti di tipo ICMP dalla lan (ovvero, permettiamo
# a ping e traceroute di uscire attraverso di noi)
$IPTABLES -A FORWARD -i $lan_iface -j ACCEPT

# la prossima riga e' molto importante: permette ai pacchetti che hanno
# gia' stabilito una connessione dalla LAN verso l'esterno di uscire
```

```
# senza ulteriori controlli
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

# la prossima riga DEVE essere l'ultima; essa permette di vedere quali
# pacchetti hanno raggiunto questo punto del firewall per poi essere
# lasciati cadere (si ricordi che l'ultima riga in assoluto e' la
# policy, che in questo caso e' DROP).
# l'opzione --limit 3/minute e --limit-burst permettono di non
# riempire i  log troppo velocemente
# si ricordi di usare il comando ''dmesg' per controllare i log, o
# di configurare correttamente syslog/syslog-ng
#$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 \
-j LOG --log-level DEBUG --log-prefix "IPT FORWARD bad packet died: "

###
#
# ora ci occupiamo della chain da noi definita come "allowed": essa
# raccoglie tutti i controlli che faremo sui pacchetti in uscita.
# Abbiamo introdotto questa chain appunto per evitare di riscrivere
# le stesse righe per ogni classe di pacchetti che vogliamo controllare.

# permetti il passaggio ai pacchetti che hanno il SYN bit settato a 1;
# questi pacchetti denotano una connessione che sta per essere
# iniziata; gli altri pacchetti che non hanno questo bit settato o
# appartengono a connessioni gia' effettuate (e che quindi abilitiamo
# in seguito), o appartengono a  connessioni "fantasma", o appartengono
# a tentativi di "spoofing". "spoofing''  e' quella tecnica per cui si
# simula un'identita' in rete con lo scopo di assumere il controllo
# di comunicazioni gia' in corso. in ogni caso non ci interessa
# che ci passino attraversino :)
#
# permettiamo solo SYN
$IPTABLES -A allowed -p TCP --syn -j ACCEPT

# permetti di passare ai pacchetti appartenenti a connessioni
# gia' stabilite (ESTABLISHED) o relativi a connessioni esistenti
# gia' autorizzate (RELATED)
$IPTABLES -A allowed -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT

# l'immancabile logging
#$IPTABLES -A allowed -p ALL -m limit --limit 3/minute --limit-burst \
3 -j LOG l-log-level DEBUG --log-prefix "allowed packet died: "

# tutto quello che non e' gia' passato viene considerato non
# autorizzato e viene lasciato cadere qui
$IPTABLES -A allowed -p TCP -j DROP

###
#
# questa e' la chain dei pacchetti ICMP; su questa possiamo lavorare
```

```
# in dettaglio, sapendo che i pacchetti di tipo ICMP verranno mandati
# qui dalle prossime sezioni del firewall
# tipi di ICMP suggeriti in genere: echo replies, dest unreach,
# redirect, time exceeded
# aggiunti in seguito: echo-request, icmp port unreachable

$IPTABLES -A icmp_packets -p ICMP --icmp-type echo-reply -j ACCEPT
$IPTABLES -A icmp_packets -p ICMP --icmp-type destination-unreachable\
-j ACCEPT
$IPTABLES -A icmp_packets -p ICMP --icmp-type redirect -j ACCEPT
$IPTABLES -A icmp_packets -p ICMP --icmp-type time-exceeded -j ACCEPT
$IPTABLES -A icmp_packets -p ICMP --icmp-type echo-request -j ACCEPT

# logging
#$IPTABLES -A icmp_packets -p ICMP -m limit --limit 3/minute \
--limit-burst 3 -j LOG --log-level DEBUG --log-prefix "IPT \
icmp packet died: "

# lasciamo cadere i pacchetti che non ci interessano.
$IPTABLES -A icmp_packets -p ICMP -j DROP

###
#
# la chain PREROUTING: controlla i pacchetti in arrivo

# le seguenti tre righe bloccano i pacchetti provenienti con ip
# sorgenti privati da internet; essi possono essere errori o
# deliberati tentativi di aggirare il nostro firewall; in ogni
# caso non ci interessano
# gli IP elencati sono le tre classi di IP riservati. La notazione
# /8, /12, /16 indica di tenere solo in considerazione i primi 8,
# 12 e 16 bit dell'ip
$IPTABLES -t nat -A PREROUTING -i $inet_iface -s 192.168.0.0/16 -j DROP
$IPTABLES -t nat -A PREROUTING -i $inet_iface -s 10.0.0.0/8 -j DROP
$IPTABLES -t nat -A PREROUTING -i $inet_iface -s 172.16.0.0/12 -j DROP

# un rudimentale port forwarding: VNC
# questa rule ``apre il corridoio'' verso questo computer; per abilitare
# gli indirizzi IP che possono accedere a VNC, vedi la FORWARD chain
# piu` avanti
# NOTA: e` possibile impostare anche un'altra porta di destinazione,
# con l'opzione --to-destination $vnc_lan:porta
$IPTABLES -t nat -A PREROUTING -p TCP -i $inet_iface --dport 5900 \
-j DNAT --to-destination $vnc_lan

###
#
# la chain FORWARD: controlla i pacchetti che ci passano attraverso

# la seguente riga e` controversa, e non e` parte di un firewall
```

```
# standard. In breve, alcuni stack TCP/IP mal implementati (vedi
# alcune versioni di Microsoft Windows) pare che inizino delle
# connessioni con pacchetti senza settare il SYN bit a 1. Questo
# e‘ ovviamente inaccettabile, tuttavia e‘ bene avere una riga di
# log pronta a segnalarci eventuali problemi nel caso qualche
# client Microsoft non riuscisse a usare la rete correttamente.
# raramente usata

#$IPTABLES -A FORWARD -p tcp ! --syn -m state --state NEW -j LOG \
--log-prefix "FORWARD New not syn: "

# blocca comunque i pacchetti che ci passano attraverso che non
# hanno il SYN bit impostato e che tuttavia sono inizio di nuove
# connessioni
$IPTABLES -A FORWARD -p tcp ! --syn -m state --state NEW -j DROP

# questa riga si occupa di permettere le connessioni di tipo VNC
# impostate prima; la porta in questo caso non cambia
# NOTA: e‘ importante specificare anche il protocollo (TCP e/o UDP
# su due linee separate) nel caso in cui si voglia impostare anche
# la porta di destinazione; ad esempio,
#  $IPTABLES -A FORWARD -i $inet_iface -s $vnc_ok -j ACCEPT
# la precedente linea ‘‘apre’’ a quell’IP specifico tutte le
# connessioni che attraversano il firewall
$IPTABLES -A FORWARD -p TCP -i $inet_iface -s $vnc_ok --dport 5900 \
-j ACCEPT
$IPTABLES -A FORWARD -p UDP -i $inet_iface -s $vnc_ok --dport 5900 \
-j ACCEPT

# l’immancabile logging
#$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 \
-j LOG --log-level DEBUG --log-prefix "IPT FORWARD packet died: "

###
#
# ora la chain INPUT: connessioni che vogliamo accettare
# nota: questa chain si riferisce solo alle connessioni che
# vogliamo accettare AL firewall

# lasciamo lavorare la scheda di rete virtuale di loopback
$IPTABLES -A INPUT -p ALL -i lo -j ACCEPT

# vogliamo essere pingabili?
$IPTABLES -A INPUT -p ICMP -j icmp_packets

# questa rule permette ai pacchetti usciti da NAT di raggiungere
# il nostro firewall; utile nel caso avessimo piu‘ indirizzi IP
# pubblici, o se volessimo giocare con gli indirizzi IP.
# Male non fa.
$IPTABLES -A INPUT -p ALL -m state --state ESTABLISHED,RELATED \
```

```
-j ACCEPT

# accettiamo i pacchetti provenienti dalla LAN; i pacchetti con
# protocollo TCP li controlliamo mediante la allowed chain, gli
# altri per i quali tali controlli non sussistono li accettiamo
# e basta
$IPTABLES -A INPUT -p TCP -i $lan_iface -d $lan_addr -j allowed
$IPTABLES -A INPUT -p ALL -i $lan_iface -d $lan_addr -j ACCEPT

# ssh disponibile all'esterno
$IPTABLES -A INPUT -p TCP --dport 22 -j allowed

# il nostro affezionato logging
#$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 \
-j LOG --log-level DEBUG --log-prefix "IPT INPUT packet died: "

# la DROP e' gia' implicita nelle policy

###
#
# infine la OUTPUT:

# lasciamo lavorare la loopback
$IPTABLES -A OUTPUT -p ALL -o lo -j ACCEPT

# il firewall puo' mandare pacchetti ovunque
$IPTABLES -A OUTPUT -p ALL -o lo -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -o $lan_iface -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -o $inet_iface -j ACCEPT

# logging
#$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3\
-j LOG --log-level DEBUG --log-prefix "IPT OUTPUT packet died: "

# fine della funzione di avvio del firewall
}

# questa funzione lancia i comandi per ripulire tutto
disable_firewall() {

# flushing delle regole
$IPTABLES -F

# puliamo la nat
$IPTABLES -t nat -F

# ripristiniamo le policy
$IPTABLES -P INPUT ACCEPT
$IPTABLES -P OUTPUT ACCEPT
$IPTABLES -P FORWARD ACCEPT
```

```
# cancelliamo le chain aggiunte
$IPTABLES -X allowed
$IPTABLES -X icmp_packets

# fine della funzione di arresto del firewall
}

# questa funzione ci permette di vedere come stanno le
# cose: packet counter, rules, eccetera
show_stats () {

$IPTABLES -L -n $1
$IPTABLES -t nat -L -n $1
$IPTABLES -t allowed -L -n $1
$IPTABLES -t icmp_packets -L -n $1

}

### end functions

### begin script core

# vediamo con che parametro e‘ stato lanciato il comando
case ‘‘$1’’ in
    start)
          enable_firewall
                ;;
    stop)
          disable_firewall
                ;;
    restart)
          disable_firewall
          enable_firewall
                ;;
# se oltre a ‘‘restart’’ si e‘ anche specificata
# l’opzione ‘‘-v’, mostra ogni cosa in dettaglio
    stat)
          show_stats $2
                ;;
    *)
          echo "Usage: $0 {start|stop|restart|stat [-v]}"; echo
esac

### end script core

--- end script ---
```

Questo script non è ottimizzato; esso va configurato a seconda delle proprie esigenze volta per volta. Tuttavia esso racchiude problematiche abbastanza

comuni in realtà medio-piccole quali possono essere aziende o privati con una piccola rete domestica.

Una nota finale: come accennato in precedenza, al posto del numero di porta è possibile specificare il nome del servizio (ad esempio ssh, smtp, eccetera). Tuttavia questa specifica richiede al firewall di controllare la corrispondente porta nel file /etc/services, aggiungendo un lieve ritardo. Inoltre il file /etc/services potrebbe essere danneggiato, mancante o semplicemente errato, causando così problemi inaspettati e di difficile identificazione.

4 Firewalling avanzato

4.1 progettazione di un firewall avanzato; miti, leggende e consigli.

La progettazione di un firewall avanzato è l'operazione più delicata; per portarla a termine con successo è necessario non solo conoscere le potenzialità e i limiti del firewall, ma anche essere in grado di prevedere tutti i casi particolari che si possono verificare. È molto difficile progettare un firewall "definitivo" sul quale poi non sia più necessario agire per correggere difetti di progettazione o per rispondere meglio alle esigenze richieste; tuttavia una buona progettazione aiuta anche a capire i limiti e i difetti di una rete esistente e permette di focalizzare meglio i punti deboli di una struttura ancor prima di agire su di essa.

È fondamentale comunque avere ben chiaro un concetto: il firewall non è la soluzione di tutti i mali. Un firewall spesso è peggio di un sistema completamente insicuro, perchè dà quel senso di "falsa sicurezza" che porta l'amministratore della rete a ignorare le avanguardie di problemi (come un attacco in corso) fino a quando non è troppo tardi.

Si dice spesso che una LAN aziendale è come un confetto: duro esternamente, ma molto morbido all'interno. Infatti un attacco dall'inteno della rete verso l'interno non viene bloccato dal firewall "di confine" (ovvero situato tra la rete privata e la rete pubblica) Un problema si security interno può portare a conseguenze disastrose, che ci sia un firewall o meno. Per questo spesso in organizzazioni di grandi dimensioni si utilizzano firewall intermedi tra i vari dipartimenti. In questo modo le diverse tipologie di utenti (management, tecnici, commerciali, amministrativi..) non rischiano di danneggiare tutti gli altri reparti per un errore.

Supponiamo infatti che un dipendente poco attento apra un allegato via e-mail infetto con un trojan o un altro virus; questo programma si "apre" una connessione dall'interno della rete verso un computer sotto il controllo di un attaccante. L'attaccante mediante questo "tunnel" prende possesso del computer del dipendente (a sua insaputa) e da quel momento può fare letteralmente il bello e il cattivo tempo all'interno della LAN. Invece, se ci sono firewall dipartimentali (o intermedi), egli non potrà spingersi troppo oltre nella sua ricerca. Si noti che l'allegato pericoloso è passato attraverso il firewall senza attivare nessun allarme: la connessione era legittima (stabilita dall'utente che ha prelevato la sua posta elettronica) ed il firewall l'ha permessa.

Per ovviare a tali inconvenienti ci sono varie tecniche, tutte dipendenti dalle risorse che si vogliono investire in sicurezza e manutenzione:

- firewall intermedi

- IDS (Intrusion Detection System)
- antivirus aziendali
- policy di sicurezza sui server
- backup
- oculata gestione degli utenti

tuttavia l'anello più debole è e resterà sempre l'uomo. Quindi, come ultimo punto (ma non meno importante), non va assolutamente sottovalutata

- l'istruzione delle persone

per la quale, purtroppo, non esiste firewall che tenga.

4.2 esempio: port forwarding su richiesta (con script bash)

Un interessante esempio di funzionalità di un sistema UNIX è l'uso di script della shell per aprire connessioni temporanee sul firewall; in pratica, mediante questo script si aggiungeranno rules al firewall tali da permettere di accedere all'interno della rete per necessità di manutenzione. Ad esempio, supponendo un firewall come quello illustrato prima senza però tutta la parte di gestione del port forwarding, uno script BASH potrebbe avere questa forma:

```
--- begin script ---
#!/bin/sh
#
# script di aggiunta/rimozione di connessioni temporanee

# un po' di variabili d'ambiente

IPTABLES="/sbin/iptables"
INTERFACE="eth1"

server1="192.168.0.99"

# con questo comando prendiamo il nostro IP di provenienza come
# registrato da SSH
# NOTA - e' necessario connettersi con SSH perche' questo
# script funzioni

SRC=$(echo $SSH_CLIENT | awk ' { print $1 } ')

# abilitiamo vnc verso questo server
# in primo luogo abilitiamo il port forwarding
$IPTABLES -t nat -A PREROUTING -p TCP -i $INTERFACE --dport 5900 \
-j DNAT --to-destination $server1 && \
echo "Tunnel to $server1 created."
```

```
# in secondo luogo "apriamo" la FORWARD unicamente per il nostro IP
$IPTABLES -A FORWARD -i $INTERFACE -s "SRC" -d "DST" -j ACCEPT && \
echo "Connection from $SRC allowed."

# quando abbiamo finito, bastera‘ premere "INVIO" sul terminale
# che avremo lasciato aperto per chiudere tutto
echo
echo "Please press ENTER when done..."
read
$IPTABLES -t nat -D PREROUTING -p TCP -i $INTERFACE --dport 5900 \
-j DNAT --to-destination $server1 && \
echo "Tunnel closed."
$IPTABLES -D FORWARD -i $INTERFACE -s ‘‘$SRC’’ -d ‘‘$DST’’ -j ACCEPT &&
echo "Connection closed."
echo "...Done."

--- end script ---
```

In questo modo un amministratore di sistema potrà connettersi via ssh al proprio firewall mediante un semplice client (ad esempio l'ottimo PUTTY per ambiente microsoft), abilitarsi una connessione eseguendo lo script e richiuderla alla fine. Lo script si occuperà di "aprire" il firewall solo all'IP di provenienza dell'amministratore. Nel caso la connessione si interrompa, basterà rientrare nel firewall e far ripartire lo script di iptables con l'opzione "restart".

NOTA: durante il periodo di apertura del tunnel, ogni connessione verso la rete remota proveniente dall'IP da cui i collega l'amministratore viene accettata. Benchè sia minima, c'è sempre la possibilità che qualche utente malintenzionato intercetti questo tipo di connessione e nel tempo di apertura del tunnel entri anche lui nella rete. Questo scenario è possibile, ad esempio, se l'amministratore dovesse trovarsi in una rete locale e il suo IP pubblico sia un NAT. Sarebbe comunque un caso molto raro.

4.3 logging

Il formato dei log restituito da iptables è il seguente:

```
external SMTP: IN= OUT=eth1 SRC=192.168.0.141 DST=213.147.102.36
LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=13574 DF PROTO=TCP SPT=1378
DPT=25 WINDOW=16384 RES=0x00 SYN URGP=0}
```

questo esempio viene generato da un comando di questo tipo:

```
iptables -t nat -A POSTROUTING -o eth1 -p TCP -s !  192.168.3.5 -d
0/0 --dport 25 -j LOG --log-level DEBUG --log-prefix ‘‘external SMTP:
‘‘
```

(che registra tutte le connessioni in uscita dalla rete 192.168.3.0 verso un qualsiasi SMTP server su porta 25 che non provengono dall'IP 192.168.3.5). In dettaglio, il log mostrato poco fa indica:

external SMTP questo è il prefisso impostato mediante –log-prefix

IN= l'interfaccia di ingresso

OUT= l'interfaccia di uscita

SRC= l'ip sorgente del pacchetto

DST= l'ip destinazione

LEN= la lunghezza del pacchetto

TOS= Type Of Service: Tipo di pacchetto

PREC= Type Of Service: Precedenza

TTL= tempo di vita del pacchetto (Time To Live)

ID= identificativo del datagram, condiviso da tutti i pacchetti di questo frammento

DF flag "Don't Fragment" settato a 1

PROTO= il protocollo usato

SPT= la porta sorgente

DPT= la porta destinazione

WINDOW= la finestra di ricezione TCP

RES= i bit riservati nell'header del pacchetto

SYN flag "SYN" settato a 1

URGP= Urgent Pointer, pacchetti da mandare assolutamente; non sempre implementato negli stack TCP.

I campi possono comunque variare in numero; quello mostrato è un pacchetto TCP, e per ogni tipologia di pacchetto ci sono vari campi che possono o meno comparire. La pagina di manuale di iptables e la documentazione online è esauriente su questo argomento.

Nota: l'header del pacchetto IP è definito nell'rfc 791, mentre quello del pacchetto TCP è definito nell'rfc 793.

4.4 problemi con il logging

Il logging di iptables non è certo un meccanismo perfetto; allo stato attuale viene migliorato costantemente, e si sta introducendo un formato nuovo di logging chiamato ULOG. È disponibile come target al posto di -j LOG (usando -j ULOG), e i pacchetti destinati a questo logging vengono catturati da demoni come "ulogd" e messi su file o database. È utile nel caso si disponga di grandi quantità di log o per creare statistiche basate su grosse moli di dati.

In ogni caso il target LOG può dare problemi (sotto grossi carichi) di consistenza, o in caso di presenza di caratteri non stampabili (ad esempio se si mandano a log anche alcune parti del pacchetto catturato, e queste parti contengono un EOF ovvero un "a-capo", esse possono mostrare errori durante la visualizzazione).

Inoltre, cambiando spesso il numero di field ("campi") del log, esso diventa difficilmente controllabile con tools come awk, sed e grep. Infine, nel caso peggiore (con tutte le opzioni abilitate) una stringa di log può diventare troppo lunga per essere leggibile.

4.5 esempio: DMZ

Una DMZ (DeMilitarized Zone) è una sorta di "zona franca" alla quale sono consentite certe connessioni (ad esempio: server web) che tuttavia si vogliono mantenere separate dalla LAN. Supponiamo uno scenario del tipo:

$$\text{internet} \iff \text{firewall} \iff \text{LAN}$$

in parallelo alla LAN si può avere una DMZ, connessa al firewall mediante una terza scheda di rete dedicata a questo scopo:

$$\begin{array}{ccccc} \text{internet} & \iff & \text{firewall} & \iff & \text{LAN} \\ & & \Updownarrow & & \\ & & \text{DMZ} & & \end{array}$$

così nel caso in cui qualche server della DMZ dovesse essere compromesso, l'attaccante dovrebbe ancora oltrepassare il firewall per poter accedere alla LAN. Una LAN con DMZ garantisce una maggiore sicurezza, anche se non assoluta. Mediante iptables si può controllare questa scheda di rete così come se fosse una normale connessione autorizzata proveniente dall'esterno. Ad esempio:

```
--- begin script fragment ---

# tra le variabili d'ambiente si inseriranno:
#
# l'interfaccia di rete della DMZ
dmz_iface="eth2"

# l'indirizzo dell'interfaccia di rete del firewall connessa alla DMZ
dmz_addr="10.0.0.1"

# l'eventuale indirizzo pubblico della DMZ, se raggiungibile
# dall'esterno
dmz_host_ext="<indirizzo pubblico>"

# e l'IP privato del server in DMZ
dmz_host_int="10.0.0.2"

# tra le regole di iptables si andra' ad inserire:

# questa rule lascia passare attraverso di noi i pacchetti che
# provengono dal'interfaccia della DMZ
$IPTABLES -A FORWARD -i $dmz_iface -j ACCEPT

# un eventuale port forwarding dall'esterno all' interno della DMZ
$IPTABLES -t nat -A PREROUTING -d $dmz_host_ext -j DNAT \
```

```
--to-destination $dmz_host_int

# blocchiamo i tentativi di connessione dalla DMZ alla LAN
$IPTABLES -t nat -A PREROUTING -i $dmz_iface -d 192.168.0.0/16 \
-j DROP

# che servizi in DMZ vogliamo rendere disponibili pubblicamente?
# abbiamo un mail server SMTP e POP3 in DMZ
$IPTABLES -A FORWARD -p tcp -i $inet_iface -s 0/0 -d $dmz_host_int \
--dport 25 - j allowed
$IPTABLES -A FORWARD -p tcp -i $inet_iface -s 0/0 -d $dmz_host_int \
--dport 110 -j allowed
# abbiamo anche un web server in DMZ
$IPTABLES -A FORWARD -p tcp -i $inet_iface -s 0/0 -d $dmz_host_int \
--dport 80 - j allowed

# nel caso in cui volessimo aprire tutta la DMZ all'esterno
# (una honeypot)
#$IPTABLES -A FORWARD -i $inet_iface -s 0/0 -d $dmz_host_int -j ACCEPT

# nella sezione INPUT, vogliamo che il firewall possa comunicare
# anche con la DMZ agevolmente
$IPTABLES -A OUTPUT -p ALL -o $dmz_iface -j ACCEPT

--- end script fragment ---
```

Questo frammento di script si inserisce nel precedente script per firewall; esso consente di determinare una rudimentale DMZ, e di gestirla a piacimento. Non si dimentichi però che la sicurezza non sta solo nella DMZ, ma anche nei servizi che vengono fatti girare in essa. Una DMZ è utile ma se viene "abbandonata a se stessa" può diventare, se compromessa, fonte di attacco verso altri sistemi con pesanti conseguenze legali e di immagine.

Nota: per "honeypot" si intende una rete "fasulla" verso la quale attirare eventuali attaccanti, distogliendoli così da bersagli più sensibili. Questo concetto è spiegato brevemente più avanti.

4.6 esempio: transparent proxying con iptables e squid

Può essere interessante usare iptables come compendio di un transparent proxy, ovvero di un proxy che non viene "visto" dagli utenti ma che permette di effettuare un controllo dettagliato sull'uso del web. È sufficiente una riga:

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT
--to-port 3128
```

ovvero: ridirigi tutte le connessioni verso la porta 80 in ingresso dall'interfaccia eth0 alla porta 3128 (sulla quale sta girando un proxy come "squid"). In questo modo non si dovranno modificare le opzioni di tutti i web browser installati all'interno della LAN.

4.7 accenni di firewall fault-tolerance: problemi nel connection keeping

Come ogni possibile punto di rottura in una rete complessa che non può affrontare tempi di downtime, anche un firewall andrebbe previsto ridondato ovvero raddoppiato in ogni sua parte per garantire che in caso di guasto esso non interrompa il proprio servizio. Il meccanismo più semplice di fault tolerance è quello di replicare il firewall stesso su un'altra macchina, connetterla al firewall principale mediante un cavo seriale e/o mediante un'interfaccia di rete dedicata a questo scopo e fare in modo che in caso di interruzione del servizio del primo firewall il secondo si attivi nel più breve tempo possibile e prenda il posto del primo. Software per questo scopo (come linux-ha) ha un unico svantaggio: il connection tracking.

Con questo termine si intende la traccia di tutte le connessioni riconosciute come "attive" dal firewall primario e per le quali esistono appositi record nella memoria stessa del sistema. Cambiando fisicamente macchina quindi si perderanno tutte le tracce delle connessioni in corso, e sarà impossibile per il firewall di backup ripristinare la situazione di partenza al 100%.

Ad esempio, supponiamo che un utente X stia navigando su internet, e che un server Y interno alla LAN stia comunicando dei dati ad un database in DMZ. Nella peggiore delle ipotesi, in caso di caduta del firewall l'utente vedrà un messaggio di errore del tipo "Impossibile caricare la pagina richiesta"; nel tempo di un reload della pagina il nuovo firewall sarà già operativo. Invece, nel caso del trasferimento di dati, si rischia la corruzione di parte di questi o un rallentamento durante la migrazione. È importante quindi cercare di minimizzare questi rischi quanto più possibile.

A questo scopo, possono essere utili software di monitoring di server basati su protocollo SNMP (Simple Network Management Protocol). Mediante questi tool, che mostrano ad intervalli di tempo costante il corretto funzionamento di ogni apparecchio e che permettono di notificare l'amministratore di rete immediatamente in caso di problemi, si riuscirà a prevedere un failure hardware ed intervenire in tempo. Ovviamente è importante su firewall di questo tipo anche basarsi su hardware ridondato che garantisca continuità di servizio: doppio alimentatore, eventuali dischi fissi in RAID, eccetera.

Concludendo, un firewall in high-availability ("ha") o in load-balancing può essere uno strumento utile per mantenere l'efficienza della rete ma non dev'essere l'unico punto di appoggio in caso di problemi.

4.8 Attacchi: port scanning, REJECT e DROP

Il primo passo di un attacco informatico spesso è un port scanning, ovvero un tentativo di stabilire se una certa porta su un host è aperta, e se c'è qualcosa in ascolto. Per effettuare la scansione si dovranno stabilire tante connessioni verso l'host quante sono le porte che si desidera controllare. Una volta controllato un numero arbitrario di porte, si controllerà se ad esse corrisponde un servizio in ascolto, e se questo servizio si rivelerà vulnerabile allora si potrà procedere all'attacco vero e proprio.

Un firewall come iptables non si occupa di prevenire l'attacco vero e proprio o di proteggere i singoli servizi; per questo compito ci sono software e tecniche di sviluppo ampiamente documentate in rete. Tuttavia iptables ha il ruolo di

controllare chi può stabilire connessioni, e può anche essere usato per nascondere la presenza del firewall stesso o per rallentare un port scan, e infine per limitare i danni in caso di attacco di tipo "flood". Un flood (dall'inglese "inondazione") è un attacco che manda enormi quantità di dati ad un server per fermarlo o per causare downtime (mancanza di servizi).

Mediante l'uso di REJECT e DROP come destinazione di un pacchetto si ottengono due effetti:

1. REJECT invia all'host di provenienza un messaggio (mediante il protocollo ICMP) di tipo "servizio non disponibile"). In questo modo l'host sa che non c'è nulla in ascolto e non aspetterà un timeout prima di mostrare all'utente un messaggio di errore. Tuttavia, un attaccante saprà che non c'è nulla a quella porta con la stessa facilità, e potrà passare immediatamente a controllare la porta successiva.

2. DROP, invece, lascerà semplicemente cadere il pacchetto. Un eventuale attaccante dovrà aspettare un certo tempo prima di passare alla porta successiva, e in caso di scansione di tutte le 65535 porte questo tempo di attesa si allungherà parecchio. Inoltre un IDS (Intrusion Detection System) si accorgerà che "qualcuno" sta provando tutte le porte ad intervalli regolari (i tempi di timeout), e potrà segnalarlo all'amministratore della rete. Ovviamente i tool di scansione delle porte come "nmap" hanno opzioni per rallentare questo processo, e le ultime versioni degli IDS vengono migliorate per riconoscere anche questi tentativi.

Concludendo, DROP per un firewall è più utile per il suo compito originario. Inoltre, DROP permette di lasciar cadere il pacchetto e il firewall non deve preoccuparsi di mandare messaggi di errore all'host di destinazione.

In ultima analisi conviene impostare il firewall con DROP per tutto quello che proviene dall'esterno e con REJECT per quello che proviene dalla rete locale. Si suppone qui che l'interesse di un utente della LAN sia quello di avere subito una risposta sulla disponibilità o meno del servizio, e che comunque una scansione di tutte le porte dall'interno della LAN richiederebbe un tempo molto più breve rispetto a quello richiesto se lo scan provenisse da internet.

4.9 Attacchi: intrusione dall'interno, trojan, reti wireless

Come si è accennato in precedenza, un grosso problema di security odierno è il falso senso di sicurezza che un firewall può dare. Un altro problema emergente è l'uso indiscriminato di apparecchiature wireless all'interno di LAN aziendali senza adeguate protezioni.

Avere una rete wireless non protetta collegata ad una LAN cablata è come lasciar penzolare un cavo di rete di 50 metri dalla finestra del proprio ufficio, sperando che di notte nessuno passi di lì, se ne accorga e usi un computer portatile per fare un giro all'interno della rete. Anche i meccanismi di protezione standard (WEP, controllo basato su MAC) sono inefficienti: le chiavi WEP non sono sicure perchè vengono ripetute ad intervalli di tempo non troppo distanti tra loro (a seconda della mole di traffico), mentre l'accesso basato su MAC è facilmente aggirabile (gli indirizzi fisici o MAC address sono modificabili a piacimento all'interno del pacchetto). Inoltre, il traffico di questo tipo gira "in chiaro" all'interno della rete, e se non si utilizza WEP viene trasmesso in chiaro

anche nell'etere. Basta intercettare una password e iniziano a sorgere i problemi (non dimentichiamoci che spesso le stesse password vengono usate per più di uno scopo, ad esempio posta elettronica, login dell'utente e password di SSH).

In caso sia assolutamente necessaria una rete wireless si deve avere l'accortezza di usare meccanismi di crittografia dei dati (mediante una VPN, ad esempio) che garantiscano la sicurezza dei dati e l'identificazione sicura del mittente. Integrando una sicurezza di questo tipo con un firewall si potrà stare abbastanza sicuri: il punto debole della rete ora sarà unicamente l'uomo, ma non si prenderanno in considerazione tutte queste problematiche in questa sede.

In ultima analisi, la pericolosità di una rete wireless è intrinseca nel fatto che chiunque può installare sul proprio computer una scheda di rete di questo tipo, ad insaputa del responsabile. Tale scheda può garantire accesso all'interno ed essere un grosso pericolo per la sicurezza.

4.10 Attacchi: uso di crittografia (SSL) per bypassare i controlli del firewall

Un altro punto debole delle politiche di sicurezza basate solo sui controlli "al perimetro" (quali firewall e IDS) è la vulnerabilità di queste ultime ad attacchi condotti mediante protocolli di crittografia. Supponiamo infatti che ci sia un web server aziendale, all'interno della LAN, che risponde alle connessioni provenienti dall'esterno mediante un meccanismo di port forwarding. Supponiamo anche che il firewall sia configurato come IDS, e che questo IDS riesca ad interpretare se i dati mandati al webserver dall'esterno siano validi o se siano indice di un attacco in corso. Come può fare un attaccante a bypassare questi meccanismi di sicurezza? Potrebbe, ad esempio, utilizzare SSL.

SSL (Secure Socket Layer) è il protocollo che permette l'invio di dati sensibili attraverso il web, e viene usato ovunque ci sia la necessità di mantenere la riservatezza delle informazioni inviate quali numeri e scadenze di carte di credito, dati personali e così via. L'IDS non potrebbe decrittare la comunicazione tra l'attaccante e il webserver che avviene mediante SSL, e non si accorgerebbe magari che i dati così inviati sono sintomo di un attacco in corso. L'attaccante probabilmente riuscirà a prendere il controllo del webserver, e per allora probabilmente sarà troppo tardi.

Non c'è una soluzione generica a questo problema: l'unica cosa da fare è mantenere sempre aggiornato il webserver con le ultime patch di sicurezza, usare DMZ molto restrittive ed eventualmente far "ascoltare" l'IDS anche all'interno della LAN.

L'uso di una DMZ in questo caso aiuterebbe perchè un attaccante, anche se avesse libero accesso al server compromesso, dovrebbe ancora superare il firewall per accedere alla rete interna; il firewall potrebbe essere configurato tuttavia per lasciar "passare" solo certi tipi di connessione dalla DMZ alla LAN (ad esempio query SQL ad un database interno), ed è probabilmente molto raro che riesca in questo intento (sempre che si accorga di essere all'interno di una DMZ).

4.11 Difese: le honeypot

In ultima analisi vanno segnalate le cosiddette "honeypot", ovvero reti "fasulle" e meno protette, che si possono mettere in ascolto accanto al firewall. Reti di questo tipo in realtà sono dei programmi (ad esempio "honeyd" in ambiente

UNIX) che simulano server e LAN vulnerabili, ma che non contengono alcun tipo di dato sensibile. Un attaccante probabilmente cercherà il punto debole di una vittima, e lo troverà nella honeypot - restando in un certo senso "imprigionato" in un ambiente fasullo, e se non si renderà conto di essere finito in una vera e propria trappola verrà rintracciato entro breve tempo.

A Appendice

A.1 fonti di informazione: comp.os.linux.security.faq

Un'ottima fonte di informazione sia generica che specifica è la comp.os.linux.security.faq, della quale riporto solo l'indice per brevità:

```
Table of Contents
1)  FAQ Administrativia

    1.1)  Introduction to comp.os.linux.security
    1.2)  Does comp.os.linux.security have a charter?
    1.3)  What is covered in this FAQ?
    1.4)  Who is responsible for this faq, and how can I contribute?
    1.5)  Acknowledgments and contributions
    1.6)  Feedback
    1.7)  Disclaimer and Copyright

2)  General Linux Security

    2.1)  How can I protect myself from hackers, quickly and easily?
    2.2)  What is security?
    2.3)  How secure should my Linux box be?
    2.4)  Which is the most secure Linux distribution?
    2.5)  Once secured, how can I *keep* my box secure?

3)  Firewalling and Masquerading

    3.1)  How do I  set up a firewall under Linux?
    3.2)  Where can I get sample IPTables rules?
    3.3)  What is the best way to start-up my firewall at boot time?
    3.4)  How do I automatically start my firewall after leasing an IP via DHCP?
    3.5)  How do I allow incoming UDP, such as CUseeMe or Battlenet through my firewall?
    3.6)  How can I firewall out banner advertising?
    3.7)  How do I reject incoming/outgoing pings?
    3.8)  How can I track traffic volume going through my firewall?
    3.9)  Why should I filter out outgoing X Windows sessions?
    3.a)  How is IPTables different than IPChains?

4)  Services

    4.1)  What services should I run?
    4.2)  What services should I not be running?
    4.3)  How do I know what services I am running?
```

```
99.2)  Linux Security FAQs, HOWTOs, and other references
99.3)  Miscellaneous Linux security resources
99.4)  Linux specific security websites
99.5)  Linux related security websites
99.6)  Other useful security resources
99.7)  Security Newsgroups
99.99)  Miscellaneous Contributors
```

Come si può vedere è una lista lunga e complessa, ed oltremodo interessante; in questa sede non posso sviluppare tutte le considerazioni tenute in quelle FAQ, tuttavia i principi di sicurezza che l'esperienza insegna a considerare restano sempre validi.

In ambiente UNIX non esiste "La Guida Definitiva", esistono piuttosto tanti "suggerimenti" che portano ad altri documenti i quali porteranno ad altre fonti di documentazione, e così via. L'importante non è conoscere tutto su iptables o sui firewall - è molto più importante sapere "dove mettere le mani" per iniziare, e continuare per conto proprio. Questo può essere un vantaggio o uno svantaggio a seconda della filosofia adottata. A titolo personale posso solo dire che questa filosofia porta nella peggiore delle ipotesi ad accrescere la propria conoscenza teorica, conoscenza poi applicabile a molti più campi di quanti ci si possa aspettare in un primo momento.

A.2 considerazioni finali: RTFM

La manpage di iptables è un'ottima guida a tutte le opzioni possibili. Consiglio vivamente di leggersela tutta almeno una volta, per farsi venire nuove idee in base alle opzioni presenti e per chiarire ogni dubbio.

Se proprio siete affamati di sapere ci sono RFC in abbondanza, e se ancora non vi basta c'è sempre il sorgente del kernel. Se, infine, non siete soddisfatti, credo che sia ora di iniziare a contribuire attivamente allo sviluppo di Linux scrivendo codice e testando quello esistente :)

A.3 links:

http://www.google.com/linux

A.4 ringraziamenti

W.A.Mozart e i Pink Floyd per l'accompagnamento musicale che mi ha seguito durante la stesura di questo testo. Antonio e Alessandro "Cirpo" per i loro sforzi di proof-reading e per la loro pazienza. Rolando e Giovanni per l'aiuto con LaTeX, e Claudio "TheFly" per avermi fatto vedere un suo lavoro in LaTeX al quale mi sono ispirato per questa struttura. Ifine, tutte le persone che mi hanno chiesto qualcosa sui firewall - grazie alle loro domande e alle informazioni che ho trovato per rispondere ho imparato più di quanto credessi. Ringrazio anche il LugMAN e tutti i membri che con i loro sforzi mantengono questo LUG sempre attivo e che stanno perseguendo il loro obbiettivo di diffusione di GNU/Linux nonostante tutte le avversità.

lorenzo grespan/verona, novembre 2003