Come funziona Linux
I servizi di rete

Dopo aver collegato il nostro computer ad Internet, vediamo come attivare i servizi di rete, ad esempio per scambiare file mediante ftp o per mettere in piedi un server web.

Tredicesima parte

di Giuseppe Zanetti

I servizi di rete sono gestiti da particolari programmi, detti server o demoni, che rimangono in attesa di connessioni sulla porta TCP o UDP riservata per il servizio (esempio la porta TCP/25 per il protocollo telnet o la porta TCP/110 per il POP3). L'assegnazione servizi-porte viene effettuata da un organismo internazionale, lo IANA, mediante il documento "Assigned Numbers" (RFC 1700).

Nel sistema operativo esiste un corrispondente di tale documento sotto forma di tabella in /etc/services. Si può vedere come quasi tutti i servizi di sistema usino porte inferiori alla 1024.

Tale scelta è stata fatta per evitare che un utente qualunque possa sostituire un server di sistema con uno proprio. Nulla vieta ad un utente comune di fare un programma che risieda su una porta con un indirizzo maggiore, a patto che non sia già usata da altri servizi.

La scelta del protocollo da usare (UDP o TCP) dipende dal tipo di servizio e dal fatto che esso necessiti di una "connessione" (ovvero di un collegamento in cui viene garantita la trasmissione affidabile di uno "stream" di dati) oppure sia sufficiente lo scambio di singoli pacchetti di dati (in questo caso è compito del programma gestire funzioni come il riordinamento dei pacchetti in arrivo o la ritrasmissione in caso di errori).

Avviare i servizi come demoni

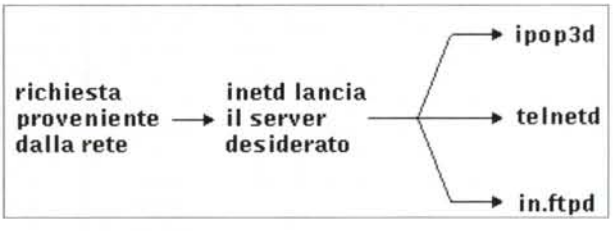
In Linux esistono due metodi per avviare servizi: il primo è quello di far girare il programma che sovrintende al servizio come demone a sé stante (standalone).

Esso di solito viene fatto partire mediante uno degli script di inizializzazione del sistema presenti sotto /etc/rc.d/. In questo modo funzionano ad esempio il server di posta SMTP sendmail (/etc/rc.d/rc3.d/S80sendmail) e il web server Apache (/etc/rc.d/rc3.d/S85httpd).

Per attivare, sospendere o far ripartire un servizio è sufficiente usare comandi del tipo:

```
/etc/rc.d/rc3.d/S85httpd start
/etc/rc.d/rc3.d/S85httpd stop
/etc/rc.d/rc3.d/S85httpd restart
```

Tale metodo ha il vantaggio di lasciare sempre in esecuzione una copia del programma pronta a rispondere velocemente alle richieste provenienti dalla rete locale o da Internet. Tuttavia in questo modo ogni programma in esecuzione porta via memoria e risorse e non è possibile gestire in modo semplice i permessi di accesso al servizio, ma ogni programma usa un proprio metodo.



Funzionamento di inetd.

Il superserver inetd

Un metodo alternativo è quello di lanciare i servizi utilizzando il "superserver" inetd. Si tratta di un programma che rimane in ascolto su tutte le porte su cui si sono abilitati dei servizi e che quando arriva una richiesta esegue una copia del programma che gestisce il servizio desiderato: ad esempio se arriva una richiesta di connessione per la porta TCP 110, viene eseguito il programma ipop3d che gestisce il protocollo POP3.

Il vantaggio di tale metodo è che non è più necessario tenere attivi in memoria tutti i server, ma solo un unico, piccolo, programma. Tale approccio consente inoltre di gestire i permessi di accesso ai servizi in modo centralizzato, in quanto è possibile fare i controlli necessari (ad esempio sull'indirizzo del client che sta tentando la connessione) prima di lanciare il programma server desiderato. Analogamente è possibile gestire altri tipi di controllo, come il numero massimo di connessioni contemporanee ad un servizio.

Inoltre, quando inetd esegue un server in risposta ad una connessione, esso si occupa anche di tutta la parte relativa al protocollo TCP. Perciò si può programmare un server come se si stesse scrivendo un programma che riceve comandi dallo standard input e manda i risultati nello standard output. Un programma siffatto è veloce da scrivere ma anche semplice da testare, in quanto è sufficiente eseguirlo da linea di comando e scrivere da tastiera l'input necessario.

Essendo parte del lavoro già fatta a monte da inetd, il codice dei singoli server diventa molto più semplice e perciò meno soggetto ad errori, con indubbi benefici anche per quanto riguarda la sicurezza.

Tutti i vantaggi appena visti si pagano però in termini di velocità, in quanto ogni volta che arriva una nuova richiesta è necessario mandare in esecuzione una nuova copia del server. In alcuni casi il ritardo introdotto può non essere tollerabile, ad esempio in server di posta con molti utenti. In questo caso è possibile vedere se il server che si utilizza può funzionare anche in modalità standalone o se è possibile sostituirlo con un software equivalente che lo permetta (ad esempio come server POP3 si può usare gnu-pop3d invece dell'ipop3d che viene fornito standard con Red Hat).

Configurazione di inetd

Per avviare un servizio mediante inetd è necessario per prima cosa verificare che esso sia definito in /etc/services. Tale file è costituito da una serie di righe, una per ogni servizio, contenenti il nome del servizio seguito dal numero della porta e dal protocollo utilizzato. Possono seguire eventuali alias o commenti. Le linee seguenti, associano alle relative porte alcuni protocolli di uso frequente:

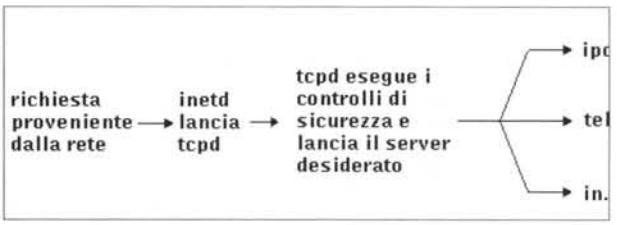
```
ftp-data  20/tcp
ftp       21/tcp
telnet    23/tcp
smtp      25/tcp   mail
domain    53/udp   nameserver dns  # Domain Name Server
www       80/tcp   http            # World Wide Web
pop3      110/tcp                  # POP version 3
talk      517/udp
```

Per ogni servizio che si vuole fornire è poi necessario definire una riga in /etc/inetd.conf che associ al servizio il programma che lo gestisce:

```
talk    dgram   udp  wait    nobody.tty  in.talkd
telnet  stream  tcp  nowait  root        in.telnetd
pop3    stream  tcp  nowait  root    /usr/bin/ipop3d
```

La prima informazione contenuta nella riga è l'etichetta con cui è stato definito il servizio in /etc/services. Segue l'indicazione sul tipo di socket e sul protocollo da usare. Nel caso il servizio si appoggi a TCP, si deve indicare "stream tcp". Nel caso si usi invece UDP, bisogna scrivere "dgram udp". Sono supportati anche altri protocolli meno utilizzati, come RPC.

L'indicazione wait/nowait permette di scegliere se attendere o meno il termine del server precedentemente lanciato prima di eseguirne ulteriori copie. Tale parametro si applica solo a connessioni di tipo UDP e dipende da come il server



Funzionamento di inetd con il supporto per il TCP wrapper.

gestisce le connessioni. Se si indica "nowait", è possibile scegliere il numero massimo di server lanciabili contemporaneamente, specificandoli nella forma "nowait.100". Nel caso tale parametro venga omesso, viene considerato un valore di default di 40.

Nella linea in /etc/inetd.conf vanno poi specificati, separati fra loro da un punto, lo user e il gruppo con i cui permessi deve girare il server. E' possibile specificare anche solo l'utente. Seguono infine il path del programma che funge da server e gli eventuali parametri con cui esso deve essere richiamato. Per i servizi, ad esempio "daytime" (provate a fare un telnet alla porta TCP 13), che vengono forniti internamente da inetd, si deve scrivere "internal".

In ogni momento possono essere aggiunti, modificati o cancellati servizi da inetd.conf. Per rendere attive le modifiche, bisogna tuttavia aver cura di far rileggere ad inetd i file di configurazione.

Ciò avviene spedendogli un segnale di tipo "SIGHUP", che si può fare con un "kill -HUP pid", dove pid è l'identificativo di processo di inetd ottenuto mediante il comando ps, oppure, più semplicemente, con "killall -HUP inetd".

Tale metodo è abbastanza standard per tutti i programmi di Linux, anche se, come vedremo fra poco, ci sono alcune importanti eccezioni.

Inetd esteso: xinetd

Nelle nuove versioni di Linux, ad esempio in Red Hat dalla versione 6.3 in poi, al posto di inetd si utilizza xinetd. Rispetto al precedente, esso consente maggiori possibilità di logging e di gestione dei permessi, oltre che una manutenzione più modulare del file di configurazione. Uno dei limiti di inetd è infatti quello di tenere tutto in un unico file, cosa che rende assai macchinoso l'inserimento o la rimozione di servizi. Xinetd consente invece di avere i soli parametri globali descritti in /etc/xinetd.conf e di creare un nuovo servizio semplicemente aggiungendo un file nella directory /etc/xinetd.d.

Un semplice server di rete

Il fatto che inetd/xinetd gestiscano al proprio interno tutti i dettagli della connessione TCP/IP e altre funzioni come i permessi d'accesso, permette di scrivere un server di rete come se si stesse scrivendo un programma da usare sulla console della macchina. Un esempio è costituito da un semplice server che accetti richieste HTTP e mandi in risposta un file. Nella sua versione base il protocollo HTTP è abbastanza semplice: quando nel browser si scrive un URL del tipo http://www.pippo.it/doc/pippo.html, viene instaurata una connessione verso la porta TCP/80 della macchina www.pippo.it e viene mandata una richiesta di tipo GET, a cui il server risponde con un header che descrive il contenuto della pagina e del file vero e proprio.

```
GET /doc/pippo.html HTTP/1.0

HTTP/1.0 200 OK
Content-Type: text/html

Ciao <B>mondo</B>
```

Fare un piccolo server usando la shell diventa dunque semplicissimo. Ovviamente si tratta solo di un esempio didattico per capire come funzionano le cose (che inoltre apre un vistosissimo buco di sicurezza...).

```
#!/bin/sh

DIR=/home/beppe/myhttpd/

# legge la richiesta del browser

read COMMAND PAGE HTTP

# verifica se il documento richiesto esiste

if [ -e $DIR/$PAGE ]
then
    # il documento esiste: manda un header di tipo
OK

    echo "HTTP/1.0 200 OK"
else
    # il documento non esiste: manda al browser un
codice di errore
    # e una pagina di tipo text/html contenente un
messaggio

    echo "HTTP/1.1 404 Not Found"
    echo "Content-Type: text/html"
    echo
    echo "<B>Errore</B>: La pagina richiesta non
esiste."
    exit 0
fi

# scrive un MIME type a seconda dell'estensione
del file

case "$PAGE" in
    *htm|*html)   echo "Content-Type: text/html"
                  ;;

    *txt|*text)   echo "Content-Type: text/plain"
                  ;;

    *gif)         echo "Content-Type: image/gif"
                  ;;
esac

# termina l'header con una riga vuota

echo

# invia il documento richiesto

cat $DIR/$PAGE

exit 0
```

E' possibile provare subito il funzionamento del nostro server eseguendolo da linea di comando e simulando da tastiera il colloquio col browser:

```
# /usr/local/bin/myhttpd
GET /index.html HTTP/1.0

HTTP/1.0 200 OK
Content-Type: text/html

Ciao <B>mondo</B>
```

Per fare in modo che il nostro server risponda ad una porta TCP, in modo che sia accessibile con un URL del tipo http://www.mioserver.com:9999/, è necessario innanzitutto inserire un nome che identifichi il nuovo servizio in /etc/services

```
myhttpd     9999/TCP       # il mio server HTTP
```

Dobbiamo poi inserire una riga in /etc/inetd.conf, che associ il servizio al nostro script:

```
myhttpd     stream tcp  nowait beppe  /usr/local/
bin/myhttpd
```

Volendo usufruire del controllo degli accessi mediante il TCP wrapper, dobbiamo modificare la riga precedente come:

```
myhttpd     stream tcp  nowait beppe  /usr/sbin/
tcpd/usr/local/bin/myhttpd
```

e inserire gli opportuni permessi in /etc/hosts.allow:

```
myhttpd: 127.0.0.1, 193.43.98.
```

Usando xinetd è necessario invece creare un file /etc/xinetd.d/myhttpd contenente le seguenti linee:

```
service myhttpd
{
    socket_type         = stream
    wait                = no
    user                = beppe
    server              = /usr/local/bin/myhttpd
    disable             = no
}
```

In questo caso il supporto per il TCP wrapper è attivo senza necessità di ulteriori aggiunte e perciò bisogna inserire gli opportuni permessi in /etc/hosts.allow. Non resta che segnalare a inetd/xinetd di rileggere i file di configurazione, con il solito killall. A questo punto è possibile inserire nella directory /home/beppe/myhttpd/ un file index.html, assegnargli il permesso di lettura per l'utente beppe e provare il tutto, prima facendo un telnet direttamente alla porta 9999 e simulando a mano la connessione e successivamente inserendo nel browser un URL del tipo http://mionome:9999/index.html.

Questo nuovo approccio offre parecchi vantaggi, specialmente per chi deve distribuire un server sotto forma di pacchetto software RPM o tar, in quanto non è più costretto a scrivere uno script che vada a modificare /etc/inetd.conf (o, peggio, a far modificare a mano il file a chi sta installando il software).

Per lanciare il server POP3 usando xinetd avremo perciò semplicemente un file di nome /etc/xinetd.d/ipop3 contenente delle righe simili alle seguenti:

```
# default:      off
# description: The POP3 service allows remote
               users to access their mail \
               using an POP3 client such as
               Netscape Communicator, mutt, \
               or fetchmail.
service pop3
{
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/sbin/ipop3d
        log_on_success          += USERID
        log_on_failure          += USERID
        disable                 = no
}
```

Principali direttive di xinetd

nome della direttiva	descrizione	valori accettati
server	percorso completo del programma da eseguire quando giunge una richiesta di connessione al servizio	percorso del server
server_args	argomenti con cui richiamare il programma che funge da server	argomenti da passare nella command line al server
only_from	reti o macchine da cui accettare connessioni	nomi simbolici o indirizzi IP; è possibile l'uso di wildcard
no_access	reti o macchine a cui rifiutare l'accesso	nomi simbolici o indirizzi IP; è possibile l'uso di wildcard; la stringa 0.0.0.0 indica tutti gli indirizzi
socket_type	tipo del socket	"stream" o "dgram"
protocol	protocollo utilizzato	"tcp" o "udp"
wait	equivalente al corrispettivo in inetd	"yes" o "no"
user	utente con i cui permessi far girare il processo	user ID o username
log_type	tipo di log: indica se si vuole avere il log su un file o mediante il metodo standard del syslogd	percorso del file di log oppure "SYSLOG"
log_on_success	indica cosa "loggare" in caso di accesso riuscito al servizio	PID, HOST, EXIT, RECORD, ATTEMPT, USERID, ...
log_on_failure	indica cosa "loggare" in caso di tentativo fallito di accesso al servizio	PID, HOST, EXIT, RECORD, ATTEMPT, USERID, ...
id	nome del servizio da usare nel log	di default viene usato il nome del programma che gestisce il servizio
instances	numero massimo di copie del server da eseguire contemporaneamente	valore numerico
start_max_load	carico della macchina a cui smettere di lanciare ulteriori copie del server	valore numerico
nice	priorità con cui eseguire il processo del server	da -10 a +10
disabled	indica se il servizio è disabilitato	"yes" o "no"

Le informazioni in esso contenute sono del tutto simili a quelle che si usavano in /etc/inetd.conf. Anzi, se si installa xinetd come upgrade di inetd, è possibile generare i file di configurazione a partire da /etc/inetd.conf usando una delle utility fornite a corredo (itox o xconv). Come si vede, tutte le opzioni sono nella forma "direttiva=valore" (oppure nella forma "direttiva+=valore" nel caso si voglia "appendere" un ulteriore valore ad una direttiva già definita). Internamente xinetd gestisce alcune funzioni particolari, come il logging (log_on_*), i cui messaggi vengono inviati tramite syslog al file /etc/secure.

Per disabilitare un servizio si può eliminare il file corrispondente oppure porre a "yes" il valore della direttiva "disable". Ad ogni modifica è necessario far rileggere al superserver i file di configurazione. A questo proposito è necessario porre una certa attenzione al fatto che il segnale da mandare a xinetd non è il "solito" SIGHUP, bensì "SIGUSR2" (killall -USR2 xinetd).

Tale eccezione rispetto allo standard "de facto" non viene presentata dall'autore come un bug, ma come una caratteristica voluta, che dovrebbe servire per mettere in difficoltà un eventuale pirata che, non conoscendo il nuovo programma, mandandogli il segnale sbagliato invece di fargli rileggere i file di configurazione lo ucciderebbe.

Togliere i servizi non necessari

La maggior parte dei problemi di sicurezza di un sistema Linux derivano dalla configurazione errata di un servizio accessibile dall'esterno o dalla presenza di servizi intrinsecamente pericolosi, come tftp. Un'altra causa importante di problemi sono i bug dei programmi.

A questo proposito esistono dei veri e propri bollettini sulla sicurezza (i più usati

Come funziona un servizio:
il protocollo POP3

Per leggere la propria e-mail non è necessario avere un client (es: Outlook), ma è sufficiente usare il programma telnet per collegarsi alla porta 110 e conoscere quanto necessario del protocollo POP3. Esso è molto semplice e consiste innanzitutto in una semplice fase di autenticazione basata su username e password. Una volta autenticati è possibile usare alcuni semplici programmi, ad esempio "LIST" per vedere la lista dei messaggi nella nostra mailbox oppure "RETR" o "TOP" per scaricare un determinato messaggio o solo il suo header (intestazione). Il comando "DELE" permette infine di cancellare un messaggio dalla mailbox. Si tratta esattamente delle stesse operazioni che compiono Outlook o Eudora quando andiamo a fare il check della nostra mailbox.

```
# telnet pop.tiscalinet.it 110
Trying 195.130.225.172...
Connected to pop.tiscalinet.it.
Escape character is '^]'.
+OK      POP3      server      ready      (5.5.025)
<1A998240BB7612A02598F7@mail.tiscalinet.it>
USER pippo
+OK Password required
PASS xvgjj665g
+OK 5 messages
LIST
+OK
1 3241
2 1023
3 1059
4 1023
5 843
.
TOP 5 0
+OK 443 bytes
Date: Thu, 19 Aug 1999 13:02:32 +0200
From: System Administrator <root@freddy.profuso.com>
Message-Id: <200108191102.NAA03466@freddy>
Apparently-To: pippo@tiscalinet.it
.

RETR 5
+OK 455 bytes
Date: Thu, 19 Aug 1999 13:02:32 +0200
From: System Administrator <root@freddy.profuso.com>
Message-Id: <200108191102.NAA03466@freddy>
Apparently-To: pippo@tiscalinet.it

test

.
DELE 5
+OK message marked for deletion
QUIT
+OK POP3 server closing connection
Connection closed by foreign host.
```

Come si vede dall'esempio, l'accesso "manuale" al POP3 non è molto difficile e può essere utile impararne i comandi base, ad esempio per cancellare quel messaggio indesiderato da 10 Mb che da alcuni giorni intasa la nostra mailbox e blocca il nostro client di posta.

Una cosa da notare è che tutti i dati, compresa la password, viaggiano in chiaro e perciò sarebbe teoricamente possibile intercettarli durante il tragitto attraverso uno dei router che ci collegano col server di posta. Per questo motivo esistono delle versioni dei protocolli Internet più usati che fanno la codifica dei dati in transito (ad esempio "POP3 over SSL").

sono quelli del CERT), che indicano le versioni dei programmi affette da problemi e i necessari rimedi (generalmente l'upgrade ad una nuova versione). Senza perdere troppo tempo, si possono ottenere buoni risultati anche aggiornando la propria installazione di Linux con le versioni "patchate" dei vari programmi che si trovano nel sito del produttore della distribuzione, ad esempio nella directory update nel sito FTP di Red Hat.

Per ridurre al minimo il numero di possibili buchi, la prima precauzione da prendere è ovviamente quella di togliere dal sistema tutti i servizi non necessari.

Nel caso di server fatti partire come demoni è sufficiente rinominare il corrispondente script in /etc/rc.d/rcX.d in modo che inizi con la lettera K (ad esempio S80sendmail diventa K80sendmail). Nel caso di servizi lanciati da inetd è invece necessario eliminare le righe corrispondenti a quelli non desiderati da /etc/inetd.conf (o commentarle inserendo un # nella prima colonna):

```
#tftp    dgram    udp      wait     root     in.tftpd
```

Per rendere attiva la modifica senza dover fare un reboot bisogna mandare un segnale HUP all'inetd mediante il comando "killall -HUP inetd".

Se si usa xinetd, si può eliminare un servizio semplicemente aggiungendo nel file corrispondente l'opzione "disable=yes".

```
service sshd
{
        socket_type             = stream
        protocol                = tcp
    instances               = 10
    nice                    = 10
    wait                    = no
    user                    = root
    server                  = /usr/local/sbin/sshd
    server_args             = -i
    log_on_success          += USERID
    log_on_failure          += USERID
    disable                 = yes
}
```

Una alternativa all'eliminazione del servizio è quella di limitarne l'utilizzo a solamente certi utenti o indirizzi IP mediante delle "access list". Nel caso di server lanciati come demoni, ogni programma ha un proprio metodo di gestire gli accessi e bisogna fare riferimento al relativo manuale per vedere come implementare eventuali restrizioni. Se invece si utilizza xinetd, è possibile aggiungere delle semplici "access list" ai diversi servizi aggiungendo al file di configurazione le direttive "only_from" o "no_access", rispettivamente per indicare i nomi o gli indirizzi IP delle macchine per cui la connessione è permessa o vietata:

```
# default: off
# description: The POP3 service allows remote
               users to access their mail \
#              using an POP3 client such as
               Netscape Communicator, mutt, \
#              or fetchmail.
service pop3
{
```

```
        socket_type         = stream
        wait                = no
        user                = root
        server              = /usr/sbin/ipop3d
        log_on_success      += USERID
        log_on_failure      += USERID
        only_from           = 193.43.*, *.profuso.com
        only_from           += 10.0.0.0
        only_from           += 192.168.0.0
        no_access           = 193.43.98.4
        disable             = yes
}
```

Il log dei tentativi di accesso riusciti o falliti viene scritto in /var/log/secure:

```
Nov 11 20:42:30 miami xinetd[493]: START: telnet
pid=12069 from=62.98.88.141
Nov 11 20:42:35 miami xinetd[493]: EXIT: ftp
pid=12066 duration=6(sec)
Nov 11 20:42:48 miami xinetd[493]: START: ftp
pid=12071 from=62.98.88.141
Nov 11 20:42:52 miami xinetd[493]: EXIT: ftp
pid=12071 duration=4(sec)
Nov 13 12:42:34 miami xinetd[493]: FAIL: ftp libw-
rap from=128.134.101.1
```

In alternativa è possibile utilizzare il metodo standard del TCP wrapper, che vedremo nel prossimo paragrafo, che funziona anche usando inetd.

Controllo degli accessi mediante TCP Wrapper

Per rendere accessibile un servizio lanciato con inetd solamente a certe macchine, si può utilizzare il TCP wrapper "tcpd". Esso viene lanciato prima del server da proteggere, e a sua volta lo lancia solamente se la connessione avviene da uno degli indirizzi autorizzati.

Per far ciò si deve modificare la linea in /etc/inetd.conf che lancia un servizio nel seguente modo, che corrisponde a lanciare il comando tcpd passandogli come parametro il nome del server desiderato:

```
pop3 stream tcp nowait root /usr/sbin/tcpd  ipop3d
```

Se si usa xinetd, il supporto per il TCP wrapper è compilato direttamente all'interno del programma (libwrap) e non è necessario utilizzare tcpd. Lo stesso vale per alcuni programmi, come openssh e sendmail, che possono usare direttamente le access list proprie del TCP wrapper anche se vengono lanciati come programmi a sé stanti.Il controllo degli accessi mediante TCP wrapper avviene confrontando l'indirizzo IP o il nome della macchina che chiede la connessione ad un servizio con il contenuto dei due file /etc/hosts.deny e /etc/hosts.allow. Il primo indica quali servizi devono essere negati e a quali macchine, il secondo indica invece cosa è permesso. Una policy prudente consiglia di usare /etc/hosts.deny per bloccare tutti i servizi a tutte le macchine:

```
ALL: ALL
```

e poi di usare /etc/hosts.allow per concedere solo i servizi desiderati a certe macchine o reti. Questo modo di operare (tutto quello che non è esplicitamente permesso è vietato) è, a mio vedere, quello sostanzialmente più corretto.

Nell'esempio che segue viene concesso l'accesso a tutti i servizi alla macchina locale. L'accesso a telnet viene concesso solo ad alcune macchine, mentre FTP viene reso disponibile alla sola rete locale.

La lettura della posta viene permessa a tutti usando il protocollo imap e a tutti fuorché le macchine della rete 212.55.44.x se si usa POP3.

```
ALL: 127.0.0.1, localhost
in.telnetd: 193.43., *.profuso.com,
freddy.pippo.com
in.ftpd: LOCAL
ipop3d: ALL EXCEPT 212.55.44.*
imapd: ALL
```

Anche nel caso del TCP wrapper, il log dei tentativi di accesso viene mandato in /var/log/secure:

```
Jan  5 11:25:59 freddy in.telnetd[10959]: refused
connect from mdm36.xyz.com
Jan  5 11:30:37 freddy in.ftpd[11515]: refused
connect from 62.98.85.124
Jan  5 11:30:38 freddy ipop3d[11619]: refused con-
nect from 62.98.85.124
Jan  5 11:31:29 freddy in.ftpd[11981]: connect
from 200.0.0.1
Jan  5 11:31:50 freddy in.telnetd[12069]: connect
from 193.43.98.12
```

Le possibilità offerte dal controllo degli accessi mediante TCP wrapper non si limitano al semplice esempio indicato. Per ulteriori dettagli si consulti la pagina del manuale in linea (man 5 access).

I metodi appena visti non sostituiscono altri metodi di protezione, come l'uso di un firewall o il tenere aggiornati i server alle versioni più recenti, tuttavia essi offrono un servizio minimo di protezione che dovrebbe essere utilizzato in ogni installazione di Linux.

Conclusioni

Con questa puntata termina il minicorso di Linux che ci ha accompagnati per più di un anno. Dal prossimo mese tenteremo di dare un po' più spazio anche ad altri argomenti, tuttavia non è detto che ogni tanto non vi sia l'occasione per qualche altra divagazione "tecnica".

L'approccio seguito non ha sicuramente la pretesa di essere esaustivo o di poter rispondere a tutte le domande. Per questo esiste l'ottima documentazione fornita dal "Linux Documentation Project" (http://www.linuxdoc.org/) e dal suo equivalente nostrano "Italian Linux Documentation Project" (http://www.pluto.linux.it/ildp), fra cui spiccano le più di 2000 pagine del libro di Giacomini "Appunti di software libero".

Da parte mia ho fatto del mio meglio per tentare di comunicare un po' della mia esperienza di 12 anni di lavoro con UNIX e Linux e spero che le varie lezioni siano state lo spunto per approfondimenti personali da parte dei lettori sugli argomenti trattati. MC