Servizi di rete in Linux
Servizi di rete in UNIX

Nei sistemi *UNIX* è possibile avviare i servizi di rete in due modi distinti:

- come **servizio a sé stante** (*standalone*);
- come **servizio gestito attraverso il supervisore inetd** (oppure **xinetd**).

Nel primo caso il *server* è un programma autonomo che generalmente viene avviato mediante uno *script* in `/etc/rc.d`, si pone in ascolto (*bind*) su una determinata porta TCP o UDP e provvede in proprio alla gestione di eventuali problematiche di sicurezza.

Utilizzando il programma supervisore `inetd`, esso rimane in ascolto su tutte le porte per cui è stato configurato un servizio e quando giunge una richiesta ad una determinata porta avvia una copia del programma che fornisce il servizio ad essa associato (ad esempio se arriva una richiesta alla porta 110 viene eseguito il *server* POP3).

La maggioranza dei programmi che gestiscono servizi di rete possono essere utilizzati sia mediante `inetd` che come *server* a sé stante.

La gestione attraverso un programma supervisore offre il vantaggio di ridurre il carico nel sistema, in quanto i servizi non rimangono attivi in memoria ma vengono avviati solamente in caso di bisogno.

Tuttavia tale soluzione non risulta facilmente scalabile, in quanto l'avvio di un nuovo processo richiede una chiamata al sistema operativo (*fork*) che risulta piuttosto pesante. Pertanto, nel caso occorrano tempi di risposta brevi o si debba gestire un numero significativo di richieste, conviene avviare il servizio come un *server* autonomo, in modo da tenerlo costantemente attivo in memoria.

Utilizzando `inetd` è possibile centralizzare la gestione della sicurezza dei servizi mediante un meccanismo di *access list* basato sul *software TCP wrapper* (`tcpd`).

Essendo parte dei compiti propri di un *server* di rete già svolta a monte da `inetd`, il codice dei singoli programmi diventa molto più semplice e perciò meno soggetto a problemi di sicurezza. Nei casi più banali, la programmazione di un servizio TCP consiste nel realizzare un programma che riceve comandi dallo standard *input* ed invia i risultati nello standard *output*.

Permessi nei servizi di rete

Come ogni altro programma, anche quelli che gestiscono i servizio di rete funzionano con i privilegi di un determinato utente. Per motivi di sicurezza generalmente si evita di far funzionare i servizi di rete con l'utente *root*, ma si preferisce utilizzare un utente non privilegiato, ad esempio l'utente generico *nobody*. In alcuni casi viene riservato un utente del sistema col preciso scopo di gestire i permessi di un determinato servizio, ad esempio httpd per il *Web server* oppure *mail* per i programmi che gestiscono la posta elettronica.

I servizi collegati a porte TCP/UDP privilegiate (inferiori a 1024) possono essere lanciati solamente da un utente con i permessi di *root*. Al contrario, qualunque utente può attivare un *server* che risponda ad una porta superiore alla 1024, a condizione ovviamente che non sia utilizzata da altri programmi.

Per superare tale limitazione, generalmente i *server standalone* vengono avviati come *root*, in modo che sia possibile eseguire il *bind* anche su una porta riservata, e successivamente eseguono una apposita chiamata di sistema che fa assumere al programma i permessi di un utente non privilegiato.

Analogamente, nel caso si utilizzi `inetd`, esso viene lanciato con i permessi di *root* e pertanto non ha limitazioni nell'eguire il *bind* sulle porte per cui è stato configurato. Successivamente i programmi da esso avviati assumono i permessi degli utenti specificati nel *file* di configurazione.

Il file /etc/services

L'associazione fra i servizi e le relative porte TCP/UDP (ad esempio la porta TCP/25 per il servizio telnet o la porta TCP/110 per il servizio POP3) viene effettuata da un organismo internazionale, lo IANA, attraverso il documento *Assigned Numbers* (RFC 1700). Nel sistema operativo esiste un corrispondente di tale documento sotto forma della tabella `/etc/services`, la quale ha un formato molto semplice, costituito dal nome del servizio, dalla porta corrispondente e da eventuali alias e commenti.

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

Esistono oltre ad `/etc/services`, altri *file* di supporto contenenti tabelle con assegnazioni standard, ad esempio la lista dei protocolli possibili sopra IP (TCP, UDP, ICMP, OSPF, IGP, ...) è contenuta in `/etc/protocols`.

Servizi eseguiti come server a sé stante

Quando c'è la necessità di avere un servizio costantemente attivo e con tempi di risposta brevi, è preferibile attivarlo come un demone, ovvero come un servizio a sé stante che si lega (*bind*) ad una porta e gestisce le richieste dirette ad essa. Generalmente quando arriva una nuova richiesta, il *server* non la gestisce direttamente, bensì manda in esecuzione una copia di se stesso come processo figlio (*fork*). Poiché tale operazione rallenta i tempi di risposta, per alcuni servizi è possibile specificare un certo numero di processi che rimangano sempre attivi (*prefork*). In questo modo si ha un miglioramento delle prestazioni, a scapito di un certo consumo di memoria. Nei sistemi moderni nel caso siano in esecuzione più processi facenti riferimento allo stesso programma, viene tenuta in memoria solamente una copia del codice (vi è comunque una duplicazione della memoria riservata alle variabili e ai dati).

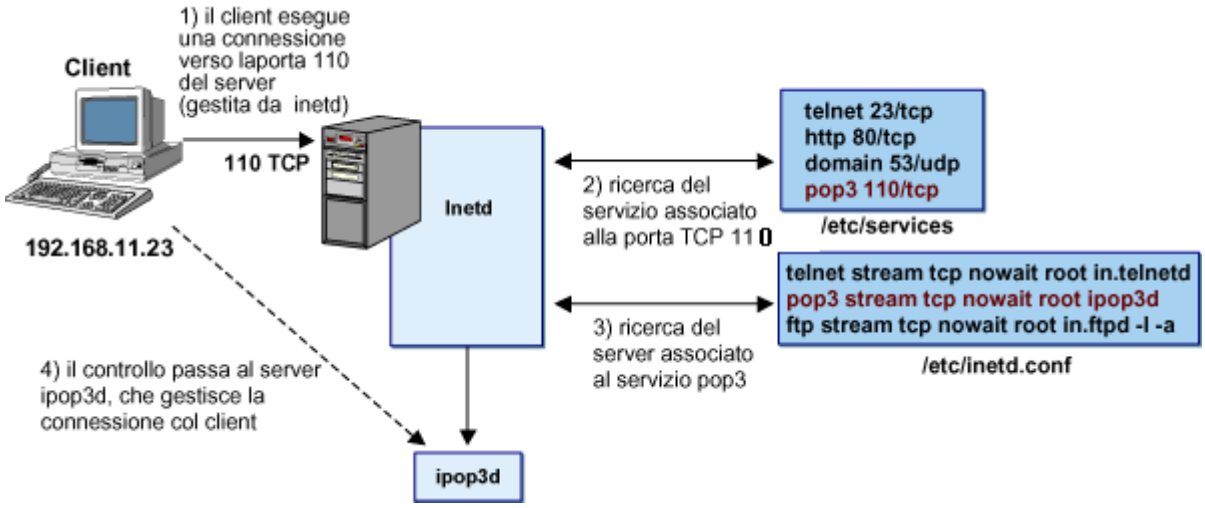
I demoni che sovrintendono ai servizi di rete vengono normalmente lanciati mediante uno *script* di avvio in `/etc/rc.d`. Ad esempio il *Web server* `httpd` viene avviato su *Red Hat Linux* mediante `/etc/rc.d/init.d/httpd` (o meglio, mediante un link simbolico posto nella *directory* `rc.d` corrispondente al *runlevel* di funzionamento, `/etc/rc.d/rcX.d/S85httpd`).

Pertanto per fermare o fare ripartire un servizio si può richiamare lo *script* opportuno nel seguente modo:

- `/etc/rc.d/init.d/httpd` *`start`* (avvia il servizio);
- `/etc/rc.d/init.d/httpd stop` (ferma il servizio);
- `/etc/rc.d/init.d/httpd` *`restart`* (riavvia il servizio);

Rispetto all'utilizzo di `inetd`, non esiste una gestione comune degli accessi ma ogni programma deve prevede dei propri meccanismi. Per i servizi per i quali sia sufficiente le gestione di una *access list*, recentemente sta diffondendosi l'utilizzo della libreria `libtcpwrap` che consente il controllo degli accessi mediante i medesimi *file* di controllo utilizzati da `tcpd`.

Servizi lanciati mediante inetd



Il *superserver* `inetd` viene generalmente avviato mediante uno degli *script* in `/etrc/rc.d` (in *Red Hat Linux* da `/etc/rc.d/rcX.d/S50inet`) e si pone in ascolto sulle porte corrispondenti ai servizi contenuti nel *file* di configurazione `/etc/inetd.conf`. Tale *file* viene letto all'avvio di `inetd` ed è possibile forzarne la rilettura nel caso siano apportate delle modifiche, inviando al demone un segnale di tipo HUP:

```
kill -HUP pid_di_inetd
```

dove il valore del PID viene ottenuto mediante il comando ps.

In *Linux* è possibile, più semplicemente, utilizzare il comando *killall* specificando direttamente il nome del processo a cui spedire il segnale:

```
killall -HUP inetd
```

Quando proviene una richiesta destinata ad una delle porte su cui è in ascolto, `inetd` determina il servizio associato ad essa utilizzando i dati contenuti nella tabella `/etc/services` ed avvia una copia del programma specificato nella linea corrispondente in `/etc/inetd`.

Ad esempio se proviene una richiesta diretta alla porta TCP 110, `inetd` ricava dalla seguente linea di `/etc/services`

```
pop3 110/tcp # POP version 3
```

il nome del servizio associato alla porta (pop3) e lo utilizza come chiave per ricavare dalla linea corrispondente in `/etc/inetd.conf`

```
pop3 stream tcp nowait root ipop3d
```

il nome del programma da mandare in esecuzione per gestire la richiesta (`ipop3d`).

Formato di inetd.conf

Il *file* `/etc/inetd.conf` è formato da un insieme di linee col seguente formato generale (eventuali commenti possono essere introdotti mediante il carattere #):

```
service sock_type proto flags user[.group] server_path args
```

dove il primo campo corrisponde all'etichetta con cui il servizio è stato definito in `/etc/services`.

Seguono le indicazioni sul tipo di *socket* e sul protocollo da usare per gestire la richiesta (*stream* tcp per TCP, *dgram* udp per UDP, ...).

Il valori *wait* e *nowait* nel campo *flag* hanno significato solo per connessioni di tipo UDP ed indicano rispettivamente di attendere o non attendere il termine del *server* precedentemente lanciato prima di eseguirne ulteriori copie. Scegliendo *nowait*, è possibile anche definire il numero massimo di copie del *server* eseguibili contemporaneamente, ad esempio *nowait*.100. Nel caso tale parametro venga omesso, viene utilizzato un valore di *default*.

Seguono le informazioni relative all'utente con i cui permessi deve girare il *server* (ed eventualmente il gruppo, nella forma *`user.group`*) ed il percorso del programma da eseguire, assieme agli eventuali parametri con cui esso deve essere richiamato. Per i servizi che vengono forniti internamente da `inetd`, si sostituisce il percorso del programma con la parola *internal*.

Un esempio di *file* `inetd.conf` è il seguente:

```
time stream tcp nowait root internal
talk dgram udp wait nobody.tty in.talkd
telnet stream tcp nowait root in.telnetd
finger stream tcp nowait root in.ftpd -l -a
pop3 stream tcp nowait root ipop3d
```

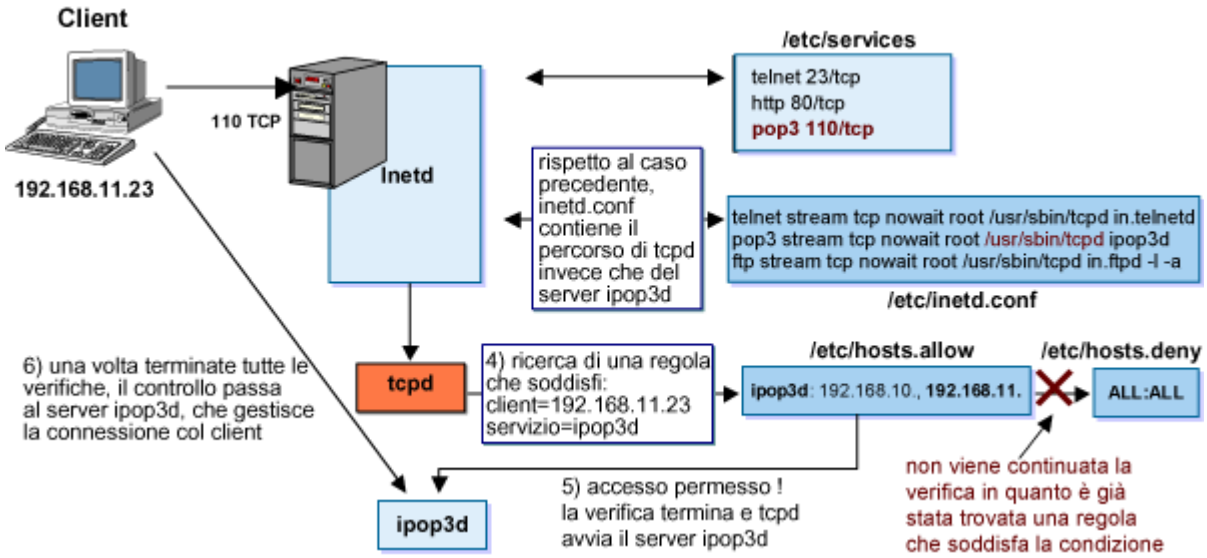
Aggiunta di un servizio ad inetd.conf

Riassumiamo le operazioni necessarie per aggiungere un nuovo servizio ad `inetd`:

- verificare se in `/etc/services` esiste già una associazione fra il nome del servizio e la corrispondente porta (esempio: smtp 25/tcp) ed eventualmente crearla.
- Agire sui *file* di configurazione di `inetd` per creare una associazione fra il servizio e il programma che lo gestisce (esempio: servizio *stream tcp nowait nobody* *`/usr/local/bin/nomeserver`*).
- Spedire ad `inetd` un segnale affinché rilegga il *file* di configurazione (*`kill -HUP pid_di_inetd`*).

Poiché tutti i servizi sono descritti nello stesso *file* di configurazione `/etc/inetd.conf`, risulta difficile aggiungere automaticamente un nuovo servizio, ad esempio da parte dello *script* di installazione di un programma.

TCP wrapper (tcpd)



È possibile demandare ad `inetd` alcuni controlli relativi alla sicurezza utilizzando il *TCP wrapper* `tcpd`. Esso viene richiamato da `inetd` al posto del programma che gestisce il servizio e a sua volta lo attiva, verificando prima che l'indirizzo del *client* che tenta l'accesso sia contenuto in una apposita *access list*.

La linea relativa ad un servizio in `inetd.conf` deve essere pertanto modificata nel seguente modo:

```
telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd
```

`Tcpd` offre inoltre funzioni di log degli accessi attraverso il meccanismo di *syslog*, che nei sistemi *Linux* viene configurato per mandare i messaggi relativi alla sicurezza in `/var/log/secure`.

```
Mar 6 15:08:58 freddy in.ftpd[16740]: refused connect from
191.125.55.80.abcdef.ru
```

La access list /etc/hosts.allow e /etc/hosts.deny

In questa sede viene spiegato solamente l'utilizzo elementare di `tcpd`, rimandando alla documentazione in linea (*man tcpd* e *man* 5 *hosts_access*) per ulteriori dettagli.

Prima di eseguire il demone che sovrintende ad un servizio, il *wrapper* confronta l'indirizzo del *client* con il contenuto dei *file* `/etc/hosts.allow` ed `/etc/hosts.deny` per verificare se esso dispone del permesso di accesso. In essi ad ogni servizio attivo nel sistema è associata una lista di *host* o reti ai quali l'accesso è permesso (*`hosts.allow`*) o vietato (*`hosts.deny`*).

La sintassi di una generica regola di `/etc/hosts.allow` o `/etc/hosts.deny` è la seguente:

elenco di *server*: elenco di *client*

in cui a sinistra si indicano i servizi e a destra gli indirizzi delle macchine a cui si vuole permetterne l'accesso. Per identificare un servizio viene utilizzato il nome del programma che funge da *server*, comedefinito in `/etc/inetd.conf`. Riferendosi agli esempi precedenti, si useranno pertanto le etichette in.telnetd, ipop3d, in.ftpd, ...

Al posto del nome di un *server* è possibile utilizzare l'etichetta *ALL* per indicare tutti i servizi. Ad esempio per permettere l'utilizzo di tutti i servizi al *client* 192.168.14.4 si può scrivere:

ALL: 192.168.14.4

Quando perviene una richiesta di accesso ad un servizio da parte di un *client*, il *software* di controllo degli accessi consulta nell'ordine i *file* `/etc/hosts.allow` e `/etc/hosts.deny`, utilizzando le seguenti regole decisionali:

- l'accesso viene permesso se la coppia (*server*, indirizzo del *client*) corrisponde ad una *entry* in `/etc/hosts.allow`. La scansione dei *file* ha termine appena una *entry* soddisfa la condizione;
- altrimenti l'accesso viene negato se la coppia (*server*, indirizzo del *client*) soddisfa una *entry* in `/etc/hosts.deny`;
- se nessuna delle condizioni precedenti è verificata, l'accesso viene permesso.

Poiché la scansione ha termine appena viene trovata la prima linea che soddisfa una condizione, l'ordine in cui appaiono le regole è importante. Inoltre bisogna fare attenzione al fatto che un *file* di controllo non esistente viene considerato come se fosse vuoto. Questo significa che se il *file* `/etc/hosts.deny` è assente o vuoto, ogni tentativo di accesso che non trovi corrispondenza in `/etc/hosts.allow` viene accettato. Pertanto è buona norma definire in `/etc/hosts.deny` la seguente regola, che per *default* blocca tutti i servizi non esplicitamente permessi mediante `/etc/hosts.allow`:

ALL:*ALL*

Nella compilazione delle liste di accesso è possibile controllare gli accessi in base all'indirizzo IP del *client* oppure mediante il suo nome simbolico:

x.y.z.k: individua un singolo *client* di cui sia specificato l'indirizzo IP

esempio: 192.168.3.45 *hostname*, esempio: pc001, pc010.dominio.com: permette di gestire l'accesso ad un singolo *client* utilizzando il suo nome simbolico. Poiché `tcpd` conosce solamente l'indirizzo del *client*, il confronto viene effettuato fra questo ed il risultato di una chiamata alla funzione di sistema *gethostbyaddr*(), che risolve il nome simbolico in un indirizzo numerico. A seconda dell'ordine specificato in `/etc/host.conf`, vengono consultati il *file* `/etc/hosts`, il DNS o una tabella NIS.

Poiché l'utilizzo di nomi di *host* o di dominio nelle regole di configurazione del *wrapper* genera una richiesta al DNS di risoluzione inversa dell'indirizzo IP in nome simbolico (192.168.12.15->pc015.miaditta.com), è bene prestare attenzione a risolvere i possibili indirizzi dei *client* il più localmente possibile, ad esempio mediante `/etc/hosts` oppure mediante una zona di riverse (12.168.192.in-addr.arpa) in un DNS sulla rete interna configurato come prioritario in `/etc/resolv.conf`. Questo accorgimento permette di evitare traffico inutile verso il DNS e possibili rallentamenti nei tempi di risposta del servizio.

L'uso di *access list* basate su nomi simbolici ove non si abbia il pieno controllo della sicurezza nel *server* DNS, è fortemente sconsigliato, in quanto un impostore potrebbe modificare la tabella di *reverse* per spacciarsi per un *client* avente diritto di accesso.

Nella scrittura delle regole possono inoltre essere utilizzate le seguenti abbreviazioni:

dominio (esempio: .dominio.com): una stringa iniziante con un punto viene confrontata con la parte finale del nome simbolico del *client* risultante da una *query* alla tabella DNS di *reverse* per l'indirizzo IP del cliente. Questo permette di abilitare l'accesso a tutti gli *host* appartenenti ad un dato dominio (esempio: pc002.dominio.com, server.intranet.dominio.con, ...). Valgono le considerazioni fatte precedentemente sull'uso di *access list* basate su nomi simbolici.

x.y.z. (esempio: 192.168.3.): in questo caso il confronto viene effettuato sui primi otteti dell'indirizzo IP del *client*. Questo permette di dare accesso ad intere sottoreti IP di classe A, B o C.

x.y.z.k/*netmask* (esesmpio: 131.155.72.0/255.255.254.0): permette di specificare degli indirizzi generici di sottorete (*classless*) utilizzando la notazione basata sulla *netmask*. La scrittura nell'esempio identifica i *client* con indirizzo compreso nell'intervallo da 131.155.72.0 a 131.155.73.255.

@dominio_nis (esempio: @administrators): identifica i *client* appartenenti ad un determinato *netgroup* NIS.

utente@host (esempio: beppe@server02.dominio.com): l'utente che richiede l'accesso al servizio viene verificato mediante il protocollo RFC 931 (oppure uno dei suoi derivati: IDENT, TAP, RFC 1413), il quale consente di richiedere ad un *host* (su cui deve essere attivo il servizio `identd`) lo *username* dell'utente che sta effettuando una connessione TCP/UDP utilizzando una determinata porta.

È possibile infine l'utilizzo nella scrittura delle regole di alcune parole chiave specifiche:

ALL: può essere utilizzata sia per indicare qualsiasi nome di servizio (esempio: *ALL*:127.0.0.1), sia per indicare qualsiasi indirizzo di *client* (esempio: ipop3d: *ALL*)

LOCAL: indica qualsiasi nome di *host* che non contenga il carattere punto (esempio: pc001, *server*02, ...). Valgono le considerazioni fatte precedentemente sull'uso di *access list* basate su nomi simbolici.

KNOWN/UNKNOWN: indica qualunque nome simbolico di *host* (oppure utente, se si utilizza `identd`) che sia (non sia) risolvibile inversamente. Tali parole chiave dovrebbero essere utilizzate con una certa cautela, in quanto un nome potrebbe non essere risolvibile anche a causa di problemi transitori in un *server* DNS.

PARANOID: indica un *client* in cui il nome simbolico indicato non corrisponda a quello restituito dal DNS. Se `tcpd` è stato compilato con l'opzione -DPARANOID, l'accesso viene rifiutato ancor prima di verificare la *access list*.

È possibile infine usare il seguente operatore:

EXCEPT: può essere utilizzato sia all'interno della lista dei servizi che in quella dei *client*. Permette di realizzare delle condizioni del tipo A eccetto B. Più operatori *EXCEPT* possono essere nidificati per creare condizioni più complesse, ad esempio a *EXCEPT* b *EXCEPT* c, equivalente a (a *EXCEPT* (b *EXCEPT* c)).

In caso di tentativo di accesso ad un servizio da parte di un *client*, è possibile eseguire dei comandi. Tale funzione può essere sfruttata ad esempio per generare un allarme, creare una trappola oppure generare un *logging* più accurato, come nell'esempio che segue:

ALL:*ALL* : `(/usr/local/etc/safe_finger -l @%h | /bin/`*`mail`*` -s "PROBE %d from %c"`
`root``) &`

Esempi

ALL:*ALL*: permette (se inserito in `/etc/hosts.allow`) oppure nega (se inserito in `/etc/hosts.deny`) l'utilizzo di qualunque servizio da parte di qualunque *client*. Viene utilizzata

in coda ad altre regole per ottenere un comportamento di *default*. È buona norma inserire solamente questa linea in `/etc/hosts.deny` ed abilitare poi mediante `/etc/hosts.allow` solo i singoli servizi/*client* che necessitano.

ipop3d: *ALL EXCEPT* .dominio: abilita il servizio POP3 a tutte le macchine ad eccezione di quelle appartenenti al dominio specificato.

in.telnetd: .dominio.com *EXCEPT* modem01.dominio.com: abilita l'accesso a telnet a tutte le macchine di un dominio ad eccezione di una macchina specifica

Strumenti di verifica

Per verificare la configurazione delle *access list* possono essere utilizzati i comandi

- `tcpdchk`: verifica la configurazione di `/etc/hosts.allow` e `/etc/hosts.deny`

```
# tcpdchk -v
warning: /etc/inetd.conf, line 83: /discaccio: world writable
warning: /etc/inetd.conf, line 84: /discaccio: world writable
Using network configuration file: /etc/inetd.conf
>>> Rule /etc/hosts.allow line 9:
daemons: in.telnetd
clients: 127.0.0.1 192.168. .test.it 209.232.130.187 pc001 63.212.
warning: /etc/hosts.allow, line 9: can't verify hostname: gethostbyname
(localhost.localdomain) failed
warning: /etc/hosts.allow, line 9: host address 209.232.130.187->name lookup
failed
warning: /etc/hosts.allow, line 9: pc001: hostname alias
warning: /etc/hosts.allow, line 9: (official name: pc001.dominio.it)
access: granted
>>> Rule /etc/hosts.deny line 10:
daemons: ALL
clients: ALL
access: denied
```

- `tcpdmatch`: verifica il comportamento della configurazione corrente simulando delle richieste provenienti da un determinato *client* per un determinato servizio:

```
# tcpdmatch in.telnetd 192.168.10.2
client: address 192.168.10.2
server: process in.telnetd
matched: /etc/hosts.allow line 9
access: granted
# tcpdmatch in.telnetd 193.43.98.12
client: address 193.43.98.12
server: process in.telnetd
matched: /etc/hosts.deny line 10
access: denied
```

Xinetd

`Xinetd` è un *software* alternativo rispetto ad `inetd`, che svolge funzioni simili e che viene utilizzato nelle versioni recenti di alcuni sistemi *UNIX* (ad esempio in *Red Hat Linux* a partire dalla versione 7.x).

Rispetto ad `inetd`, `xinetd` offre i seguenti vantaggi:

- **modularità**: oltre al *file* comune di configurazione (`/etc/xinetd.conf`), ogni servizio può utilizzare un proprio *file* di configurazione del tipo `/etc/xinetd.d/servizio`. Questo facilita la pacchettizzazione e l'installazione del *software*, in quanto per aggiungere un nuovo servizio di rete è sufficiente copiare un *file* in una *directory* e mandare ad `xinetd` un messaggio di rilettura della configurazione (*`kill -USR2 pid_di_xinetd`*).
- **gestione migliorata della sicurezza**: le funzioni di *wrapper tcp* non vengono più gestite mediante un programma esterno, ma internamente da `xinetd`, attraverso la libreria `libwrap`.

Configurazione di xinetd

Nel *file* globale di configurazione `/etc/xinetd.conf` vengono di solito definiti solamente i parametri di *default* per tutti i servizi (numero massimo di istanze contemporanee, *logging*, ...). Un esempio è il seguente:

```
defaults
{
instances = 60
log_type = SYSLOG authpriv
log_on_success = HOST PID
log_on_failure = HOST
}
includedir /etc/xinetd.d
```

L'ultima riga indica di includere nella configurazione tutti i *file* contenuti nella *directory* `/etc/xinetd.d`.

Nel seguente esempio è mostrato il *file* di configurazione specifico per il servizio POP3:

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

I valori contenuti nel *file* sono simili a quelli che si utilizzavano in `inetd.conf`. Essi sono espressi nella forma direttiva=valore, oppure nella forma direttiva+=valore nel caso si voglia appendere un ulteriore valore ad una direttiva già definita.

Le principali direttive utilizzabili per la configurazione di un servizio sono le seguenti:

Nome della direttiva	Descrizione
server	percorso completo del programma che gestisce le richieste per un determinato servizio
*server*_args	argomenti con cui richiamare il programma che funge da *server*
	access list sulle reti o macchine da cui accettare le connessioni verso questo

only_from	servizio (è possibile l'utilizzo sia di nomi simbolici che indirizzi IP, nonché l'uso di espressioni regolari)
no_access	*access list* sulle reti o macchine da cui rifiutare le connessioni verso questo servizio
socket_type	tipo del *socket* (*stream* o *dgram*)
protocol	protocollo utilizzato (tcp o udp)
wait	equivalente al corrispettivo in `inetd` (valori possibili: *yes* e no)
user	*username* oppure UID dell'utente con i cui permessi si desidera far funzionare il servizio
log_type	permette di scegliere se avere il log su un *file* (in questo caso se ne deve indicare il percorso), oppure se si desidera utilizzare il metodo standard *syslogd* (in questo caso si deve scrivere *SYSLOG*)
log_on_success	nel caso di accesso permesso al servizio, indica gli eventi in concomitanza dei quali deve essere emesso un messaggio di log (PID, HOST, EXIT, RECORD, ATTEMPT, USERID, ...)
log_on_failure	eventi per i quali deve essere emesso un messaggio di log in caso di accesso fallito al servizio
id	etichetta da utilizzare per identificare questo servizio nel *file* di log (come *default* viene usato il nome del programma che gestisce il servizio)
instances	numero massimo di connessioni contemporanee al servizio
start_max_load	carico della macchina oltre cui smettere di eseguire ulteriori copie del *server*
nice	priorità (da -10 a +10) con cui eseguire il servizio
disabled	se posto a *yes* disabilitare il servizio

Per disabilitare un servizio, al posto di eliminare il *file* corrispondente, si può pertanto porre a *yes* il valore della direttiva *disable*. Ad ogni modifica è necessario far rileggere a `xinetd` i *file* di configurazione mediante *kill*.

RPC (Remote Procedure Call)

RPC è un protocollo che permette l'esecuzione di procedure remote, utilizzabili nella realizzazione di programmi *client server*. Il funzionamento è basato, nel lato *server*, sul demone `portmap` (`rpc.portmap`), il quale viene avviato mediante uno *script* in `/etc/rc.d` e fornisce ai *client* le informazioni necessarie a contattare il servizio che fornisce una data procedura (ad esempio `rpc.nfsd`). `Portmap` può essere pensato come l'equivalente di `inetd` per il protocollo RPC. L'elenco dei servizi RPC e dei relativi numeri identificativi è contenuto nel *file* `/etc/rpc`:

```
portmapper 100000 portmap sunrpc rpcbind
rstatd 100001 rstat rup perfmeter rstat_svc
rusersd 100002 rusers
nfs 100003 nfsprog
ypserv 100004 ypprog
mountd 100005 mount showmount
ypbind 100007
```

I servizi RPC attivi su un *server* possono essere interrogati mediante il programma `rpcinfo`:

```
# rpcinfo -p nomeserver
program vers proto port
100000 2 tcp 111 portmapper
```

```
100000 2 udp 111 portmapper
100011 1 udp 975 rquotad
100011 2 udp 975 rquotad
100005 1 udp 985 mountd
100005 1 tcp 987 mountd
100005 2 udp 990 mountd
100005 2 tcp 992 mountd
100003 2 udp 2049 nfs
```

Esso inoltre può essere utilizzato per elencare tutti i *server* in grado di fornire un determinato servizio:

```
# rpcinfo -b mountd 1
192.168.10.1 server01
192.168.10.5 sun05
```