Ministero dell'Istruzione, dell'Università e della Ricerca
Servizio Automazione Informatica e Innovazione Tecnologica

Modulo 7

Sistemi operativi di rete

ForTIC

Piano Nazionale di Formazione degli Insegnanti sulle Tecnologie dell'Informazione e della Comunicazione

Percorso Formativo C

Materiali didattici a supporto delle attività formative

2002-2004

Promosso da:

- Ministero dell'Istruzione, dell'Università e della Ricerca, Servizio Automazione Informatica e Innovazione Tecnologica
- Ministero dell'Istruzione, dell'Università e della Ricerca, Ufficio Scolastico Regionale della Basilicata

Materiale a cura di:

- Università degli Studi di Bologna, Dipartimento di Scienze dell'Informazione
- Università degli Studi di Bologna, Dipartimento di Elettronica Informatica e Sistemistica

Editing:

CRIAD - Centro di Ricerche e studi per l'Informatica Applicata alla Didattica

Progetto grafico:

Campagna Pubblicitaria - Comunicazione creativa

Scopo e obiettivi del modulo

In questa sezione verrà data una breve descrizione del modulo.
Gli scopi del modulo consistono nel mettere in grado di:

- Descrivere le principali funzioni dei componenti *software* di un *server* di rete.
- Descrivere le principali funzioni degli attuali sistemi operativi di rete.
- Installare e configurare *server software* di rete locale, *driver*, *routing*.
- Installare e connettere più *server* di rete.
- Installare e configurare *software* antivirus.
- Installare e configurare *software client* di rete.
- Installare un *proxy server*.

Il modulo è strutturato nei seguenti argomenti:

- ***Server software***
  - Descrivere le principali funzioni dei componenti *software* di un *server* di rete.
  - Confrontare le principali funzioni e caratteristiche degli attuali sistemi operativi di rete (includendo i servizi di *directory*).
  - Installare e configurare un *server* di rete locale.
  - Installare e configurare *driver* per schede e periferiche di rete.
  - Installare e configurare il *routing* di una rete usando la documentazione appropriata.
  - Descrivere aspetti di gestione e procedure per gestire più *server* su una rete.
  - Installare e connettere più *server* (anche basati su piattaforme diverse).
- ***Client software***
  - Descrivere le principali funzioni dei componenti *software* di un *client* di rete.
  - Installare e configurare *client software* su più piattaforme *hardware*.
  - Installare e configurare *driver* per schede ed altre periferiche di rete (incluso le stampanti).
  - Configurare un *client* in modo che supporti più protocolli.
- **Installare e configurare servizi**
  - Installare e configurare applicazioni *client/server* su un *server* quali: *e-mail*, FTP, *Web*, sistemi di messaggistica, *chat*, eccetera.
  - Installare un *proxy server*.

Introduzione

Server software

Franco Callegati
Paolo Presepi
Riccardo Gori

Principali funzioni dei componenti software di un server di rete

Un *server* è composto principalmente da un sistema operativo, protocolli e moduli di rete e il supporto di programmi *server* e servizi.

Il sistema operativo

Le caratteristiche essenziali di un sistema operativo orientato alla gestione di ambienti distribuiti di rete sono la multiprogrammazione e la multiutenza.
Un sistema operativo multiprogrammato è in grado di eseguire più programmi contemporaneamente mediante algoritmi di gestione dei processi. Questa peculiarità è fondamentale per consentire ai *client* di utilizzare una moltitudine di programmi e servizi residenti sul *server*. La capacità di gestire più utenti consente una maggiore sicurezza del sistema unita possibilità di regolare permessi di accesso alle risorse.

Protocolli di rete

Il sistema operativo comprende inoltre moduli che implementano i protocolli di rete, possono essere integrati nel *kernel* oppure essere caricati o installati su richiesta. (Ad esempio moduli del *kernel* di *Linux* o l'installazione dei protocolli nella rete per i sistemi *Windows*).

Programmi *server* e servizi

Tutte le funzionalità *server* solitamente sono demandate a programmi specifici, denominati programmi *server*. Tali programmi vanno configurati e impostati secondo le esigenze della rete in cui operano, occorrono perciò buone conoscenze di base sia del sistema operativo in uso, sia delle reti e dei protocolli di rete utilizzati.
I programmi *server* installati implementano soltanto il lato *server* nell'architettura *client/server* necessaria per il funzionamento del servizio. Di solito, dipendentemente delle risorse del sistema (velocità di calcolo e spazio disco e banda disponibile), si integrano più programmi *server* sulla stessa macchina, dando luogo a sistemi concentrati di servizi di rete.

Confronto tra le funzioni dei principali sistemi operativi di rete

Esistono attualmente disponibili sul mercato o liberamente reperibili su Internet una grande quantità di sistemi operativi con implementazioni e funzionalità orientate alle reti.

Tra i sistemi maggiormente diffusi di tipo *open source* citiamo *GNU/Linux*: disponibile in una moltitudine di distribuzioni è implementazione libera e *open source* del sistema *UNIX*. La famiglia di sistemi BSD come *OpenBSD*, *NetBSD*, *FreeBSD* sono implementazioni *open source* derivate da BSD *UNIX*, la versione di *UNIX* sviluppata alla *Berkeley Univesity*.

Tra i sistemi operativi commerciali troviamo implementazioni del sistema *UNIX* come *Digital UNIX* (*Digital*), Tru64 (*Compaq*), HP-UX (HP), AIX (IBM) e Solaris (*Sun*). *Windows* NT/2000 e XP sono implementazioni del sistema *Windows* prodotto da *Microsoft* per le reti.

Tuttavia, per la generalità degli argomenti trattati e per la preponderante diffusione in ambienti scolastici presentiamo le configurazioni e le descrizioni dei sistemi operativi: *GNU/Linux RedHat* e *Microsoft Windows* 2000. Per le diverse versioni e varianti di questi sistemi operativi e per sistemi simili come struttura e funzionalità rimandiamo alle descrizioni ed ai manuali relativi.

Vedremo le loro principali strutture e funzionalità.

Caratteristiche di Windows2000 - Client

Microsoft Windows 2000 *Professional* migliora le caratteristiche delle precedenti versioni di *Windows* nelle seguenti aree aree:

- **Semplicità d'uso**. Oltre ai miglioramenti relativi all'interfaccia utente, la semplicità d'uso di *Microsoft Windows* 2000 *Professional* si traduce nelle funzionalità di Supporto per Utenti Remoti e nelle funzionalità di Supporto della Stampa. Per quel che riguarda il Supporto per Utenti Remoti ricordiamo le seguenti

funzionalità:
    - *Network Connestion Wizard*. Permette di configurare tutte le tipologie relative ad un *client* di accesso remoto (connessione *dial-up*, connessione ad Internet, connessione VPN, connessione via cavo).
    - Supporto per *Virtual Private Network* (VPN). Permette di connettersi alla rete aziendale tramite un ISP in maniera sicura.
    - Cartelle *Offline*. Permette di memorizzare localmente *files* residenti su un *server* in maniera tale che siano accessibili anche in modalità *offline*.
    - Per quel che concerne invece il Supporto della Stampa ricordiamo le seguenti funzionalità:
        - *Internet Printing Protocol* (IPP). Permette di inviare documenti ad un *print server Microsoft Windows* 2000 tramite Internet.
        - *Add Printer Wizard*. Semplifica il processo di installazione di un *client*.
- **Gestione Semplificata**. Permette all'utente di gestire in maniera semplificata i dati, le applicazioni e tutte le impostazioni di sistema. Ricordiamo le seguenti funzionalità:
    - *ADD/Remove Programs Wizard*.
    - Servizio di *Windows Installer*.
- **Supporto *Hardware***. *Microsoft Windows* 2000 *Professional* supporta più di 7000 *device hardware*. Ricordiamo a tal proposito:
    - *Wizard Add/Remove Hardware*.
    - Supporto *Plug and Play*.
    - Opzioni di risparmio energetico: *standby*, ibernazione.
    - Sistema *multiprocessor* simmetrico (SMP). Fino a due processori.
- **Gestione dei *Files***. *Microsoft Windows* 2000 *Professional* supporta le seguenti tecnologie per semplificare e migliorare la gestione dei *files*:
    - *File System* NTFS. Supporta la crittografia, la sicurezza locale ed il controllo delle quote.
    - *File System* FAT32. Garantisce la piena compatibilità con *Microsoft Windows* 95 OSR2 e versioni successive.
    - *Utility* di Deframmentazione dei Dischi.
    - *Utility* di *Backup*.
- **Sicurezza**. *Microsoft Windows* 2000 *Professional* è il sistema operativo *desktop* più sicuro, sia per ambienti *stand-alone* che per ambienti di rete. Ricordiamo le seguenti funzionalità:
    - *Kerberos* 5. Protocollo standard di autenticazione che permette una autenticazione ed un accesso alle risorse di rete più efficaci.
    - *Encrypting File System* (EFS). Permette di cifrare i *file* memorizzati su disco.
    - *Internet Protocol Security* (IPSec). Permette di specificare politiche di cifratura per flussi di dati sulla rete.

Chiudiamo questa breve panoramica elencando quelli che sono i prerequisiti *hardware* per l'installazione di *Microsoft Windows* 2000 *Professional*:

- CPU: *Pentium based*
- Memoria : 32 MB (raccomandati 64 MB)
- Spazio Disco : uno o più dischi con un minimo di 650 MB (raccomandati 2 GB)
- Sulla partizione che conterrà i *files* di sistema
- Scheda di rete
- Scheda Video VGA Standard o superiore
- Tastiera e *Mouse*

Caratteristiche di Windows2000 - Server

Microsoft Windows 2000 *Server* è una piattaforma che contiene tutte le funzionalità di *Microsoft Windows* 2000 *Professional* più le funzionalità che ne ottimizzano le *performance* per le funzionalità di *file server*, *print server* ed *application server*.

Alla base ci sono tutta una serie di servizi basati su *Active Directory*, che permette di centralizzare la gestione di utenti, gruppi, sicurezza e risorse di rete.

Microsoft Windows 2000 *Advanced Server* supporta fino a 8 processori SMP e fino a 8 GB di memoria RAM.

Microsoft Windows 2000 *Server* è una buona soluzione per l'implementazione di soluzioni *enterprise* in realtà medio-piccole: *file server*, *print server*, *Web server*, *Terminal Services server*, *server* di accesso remoto.

Di seguito i requisiti *hardware* minimi per l'installazione di *Microsoft Windows* 2000 *Server*:

- Processore: 32-bit *Pentium* 133 MHz.
- Memoria: 64 MB per reti di meno di 5 *computer*; 128 MB è il minimo raccomandato per tutte le altre situazioni.
- Spazio Disco : uno o più dischi con un minimo di 680 MB (raccomandati 2 GB) Sulla partizione che conterrà i *file* di sistema.
- CD-ROM o DVD-ROM *drive* (El Torito-compatible).
- Scheda di rete.
- Scheda Video VGA Standard o superiore.
- Tastiera e *Mouse*.

Caratteristiche di Windows2000 - Windows 2000 Advanced Server and Datacenter Server

Microsoft Windows 2000 *Advanced Server* è la piattaforma *software* raccomandata per *application server* dipartimentali che necessitano anche di elevata disponibilità e bilanciamento dei carichi di lavoro (*cluster*). *Microsoft Windows* 2000 *Server* supporta fino a 8 processori SMP e fino a 8 GB di memoria RAM.

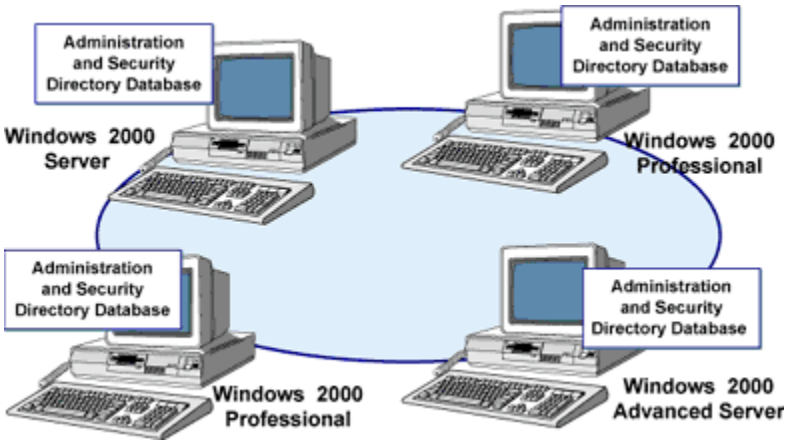
Di seguito i requisiti *hardware* minimi e raccomandati per l'installazione di *Microsoft Windows* 2000 *Advanced Server*:

- Processore: *Intel Pentium* 166 (raccomandato uno o più *Pentium* III 300).
- Memoria: 64 MB (raccomandato 128 fino ad un massimo di 8 GB).
- CD-ROM o DVD-ROM *drive* (El Torito-compatible).
- 2 GB di spazio su dischi Ultra IDE (raccomandato Ultra Wide SCSI).
- Scheda di rete (raccomandato *Fast Ethernet*).
- Scheda Video VGA Standard o superiore.
- Tastiera e *Mouse*.
- Per l'implementazione del *cluster* è richiesto *hardware* dedicato.

Microsoft Windows 2000 *Datacenter* è la soluzione *server* ottimizzata per *data warehouse*, *online transaction processing* (OLTP), simulazioni in *real-time*, *Web hosting*.
Microsoft Windows 2000 *Datacenter* ha tutte le caratteristiche di *Microsoft Windows* 2000 *Advanced Server*, ma supporta fino a 64 GB di memoria RAM e fino a 32 processori SMP.

Per concludere, escludendo la funzionalità di *Clustering*, l'unica differenza tra *Microsoft Windows* 2000 *Server* e le versioni *Advanced Server* e *Datacenter* e nel numero di processori e nella quantità di memoria RAM supportata (scalabilità).

Caratteristiche di Windows2000 - Workgroup



Microsoft Windows 2000 permette di implementare due ambienti di rete in cui gli utenti possano condividere risorse (*file*, stampanti, applicazioni) indipendentemente dalle dimensioni della rete: i *workgroup* ed i domini.

Un *Workgroup Microsoft Windows* 2000 è un insieme logico di *computer*, cominicanti tra loro e che condividono risorse.

Un *workgroup* è chiamato anche rete *Peer-to-Peer* (paritetica) per evidenziare quella che è la sua caratteristica saliente: tutti i *computer* che appartengono ad un *workgroup* sono uguali, senza che ci sia un *server* dedicato alla gestione della sicurezza.

Ogni *computer* che esegua sia *Microsoft Windows* 2000 *Professional* sia *Microsoft Windows* 2000 *Server* gestisce un proprio *security database* locale, cioè una lista di utenti ed impostazioni di sicurezza inerenti il *computer* che ospita tale *database*.

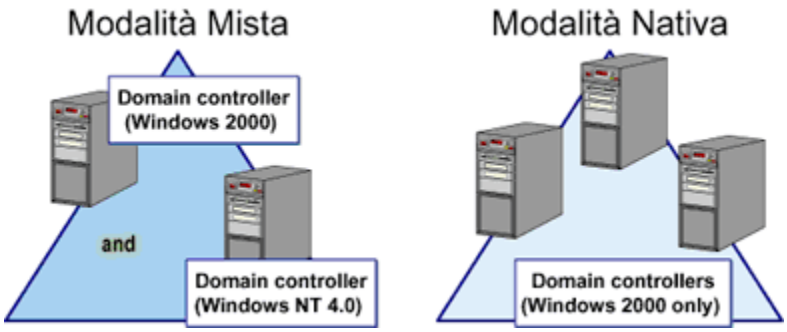
Dunque in un *workgroup* la gestione degli utenti e della sicurezza è decentralizzata, per cui:

- L'utente deve avere un *account* per ogni *computer* che intende utilizzare.
- Se si vuole che l'utente possa utilizzare tutte le macchine ed accedere alle risorse di tutte le macchine, bisogna definire un *account* per ogni *computer*.

Un *workgroup Microsoft Windows* 2000 ha i seguenti vantaggi:

- Non richiede una gestione centralizzata, dunque un *server* dedicato, degli *account* e della sicurezza.
- Non richiede nessuno sforzo particolare relativamente alla progettazione ed all'implementazione.
- È la soluzione più conveniente per ambienti costituiti da pochi *computer* distribuiti su un'area molto limitata (meno di 10 *computer*!).

Caratteristiche di Windows2000 - Domini



Sono in sintesi, gli aspetti essenziali del dominio *Windows* 2000. Come un *Workgroup*, un Dominio *Microsoft Windows* 2000 è un insieme logico di *computer*, cominicanti tra loro e che condividono risorse.

La differenza consiste nel fatto che tali *computer* condividono un *directory database* centralizzato, cioè un *database* che contiene la definizione degli *user account*, dei gruppi e tutte le impostazioni inerenti la sicurezza. Tale *database* è chiamato *Directory* ed è una parte di *Active Directory* che è il *directory services* di *Windows* 2000.

Tale *database* è contenuto su un *server* particolare denominato ***Domain Controller***.

Tale soluzione è affetta da quello che è il tipico problema di tutte le soluzioni centralizzate, il *Single Point of Failure*: in caso di fallimento del *Domain Controller*, le sue funzionalità, tra cui il *database*, vengono a mancare e dunque il dominio cessa di esistere. Per ovviare a questo problema in un dominio è dunque prevista (e consigliata!) la presenza di almeno due *Domain Controller* che dunque garantiscono *Fault Tolerance* e, a regime, bilanciamento dei carichi di lavoro. Infatti tali macchine sono assolutamente paritetiche, entrambe hanno una copia del *Directory* in lettura/scrittura e tali copie vengono periodicamente allineate tramite un processo di replica. Tale ambiente di replica è detto *Multi Master*.

In *Microsoft Windows* NT 4.0 valgono le stesse considerazioni, con la fondamentale differenza che solo uno dei *domain controller*, il *Primary Domain Controller* (PDC), ha il *Directory* in lettura/scrittura mentre tutti gli altri ne hanno una copia in sola lettura e perciò sono denominati *Backup Domain Controller* (BDC). Tale ambiente di replica è detto perciò *Single Master*.

I vantaggi derivanti da un dominio sono:

- Ammnistrazione Centralizzata: da un unico punto amministro tutti gli utenti, tutti i gruppi e la sicurezza.
- *One User One Account*: con un unico *username* ed un'unica *password* l'utente accede al dominio da qualsiasi postazione di lavoro.
- Accesso Universale alle Risorse: una volta validato dal dominio l'utente può accedere a tutte le risorse, indipendentemente dalla loro localizzazione, previa autorizzazione.
- Scalabilità.

Un dominio *Microsoft Windows* 2000 è dunque formato dai seguenti tipi di *computer*:

- *Domain Controllers*: ogni *domain controller* ha una copia in lettura/scrittura del *Directory*, pertanto su ogni *Domain Controller* è possibile creare amministrare e gestire utenti, gruppi, impostare la sicurezza ed ogni *Domain Controller* può effettuare l'attività di validazione. È previsto un meccanismo di replica tra le varie copie del *Directory*.
- *Member Servers*: *computer* che eseguono una qualsiasi versione di *Microsoft Windows* 2000 *Server*, appartengono al dominio ma non svolgono funzionalità di *Domain Controller*.
- *Client Computers*: *computer* che appartengono al Dominio ed eseguono un qualsiasi sistema operativo *Microsoft*, con ovvia preferenza per *Microsoft Windows* 2000 *Professional* per avere garanzie di stabilità,

sicurezza ed affidabilità.

Concludiamo con qualche osservazione relativa alle situazioni di convivenza tra *Domain Controller Windows* 2000 e *Domain Controller Windows* NT 4.0, situazione tipicamente riscontrabile nei casi di *upgrade* a *Windows* 2000.

Per favorire tale convivenza un dominio *Microsoft Windows* 2000 nasce sempre in Modalità Mista, che sostanzialmente disabilita tutte le nuove funzionalità di *Microsoft Windows* 2000 che non sarebbero capite dai *Domain Controller Microsoft Windows* NT 4.0. Non appena questi ultimi sono stati tutti aggiornati al nuovo sistema operativo il dominio può finalmente essere passato in Modalità Nativa che abilita tutte le nuove funzionalità di *Windows* 2000 ma non permette la presenza di *Domain Controller Windows* NT 4.0.

Attenzione poiché il processo di conversione dalla Modalità Mista alla Modalità Nativa è irreversibile, cioè si torna indietro solo installando nuovamente il dominio.

Caratteristiche di Windows2000 - Concetto di Active Directory

Active Directory è il *Directory Services* di *Microsoft Windows* 2000.

Il *Directory Service* è un servizio di rete che ha lo scopo di gestire tutte le informazioni inerenti le risorse di rete per renderle accessibili agli utenti ed alle applicazioni; permette di identificare, descrivere, localizzare, accedere, gestire e rendere sicure tali risorse.

Dunque *Active Directory* fornisce le funzionalità per organizzare, gestire e controllare in maniera centralizzata l'accesso alle risorse di rete, in maniera trasparente rispetto alla topologia di rete ed al protocollo utilizzato. Tramite *Active Directory* è possibile memorizzare ed organizzare un numero praticamente illimitato di oggetti.

Caratteristiche di Linux - Le distribuzioni di Linux

Comunemente col termine *Linux* si intende un sistema completo, composto dal *kernel* (nucleo) più il *software* applicativo, anche se in realtà sarebbe corretto riservare tale denominazione solo per indicare il *kernel* e invece riferirsi al sistema completo come distribuzione di *Linux*.

Al contrario di altri sistemi operativi, in cui il nucleo e i programmi di base vengono realizzati (o forniti in licenza) da un unico produttore, una distribuzione di *Linux* viene realizzata mettendo assieme *software* derivati da progetti e da fonti diverse. Esistono inoltre diverse distribuzioni, realizzate da diversi produttori, che differiscono fra loro in modo anche significativo, sia per i metodi di installazione e configurazione sia per i pacchetti *software* contenuti.

Si deve porre attenzione al fatto che la numerazione delle versioni delle distribuzioni di norma non coincide con quella della versione del *kernel* di *Linux* utilizzata. Ad esempio la versione *Linux Red Hat* 6.1 utilizza un *kernel* versione 2.2.12.

Caratteristiche di Linux - Le licenze Open Source

Il *kernel* di *Linux*, la maggior parte del *software* di sistema ed i programmi più importanti derivano da progetti disponibili sotto una licenza di tipo *Open Source* (a sorgente aperto), delle quali la più utilizzata è la *GNU General Public License* (GPL).

Una licenza *Open Source* differisce in modo sostanziale da una licenza di tipo commerciale, in quanto invece di nascondere il codice sorgente, permette l'accesso ad esso e ne incoraggia la modifica.

È permesso di includere tutto o parte di un programma distribuito secondo la licenza GPL all'interno di un proprio

progetto, nonchè di ridistribuirlo ad altri, a condizione che non se ne cambi la licenza originaria e che si mantenga evidente il *copyright* degli autori precedenti. Una licenza infatti non influisce sulla proprietà intellettuale di un'opera di ingegno, bensì solo sulle modalità con cui l'autore ne concede ad altri l'utilizzo. Per maggiori informazioni si può fare riferimento alla *Copyright FAQ*, che descrive la differenza fra i diversi tipi di licenza:

(**ftp://rtfm.mit.edu/pub/usenet/news.answers/law/copyright**)

Per i motivi precedentemente elencati, il *software Open Source* viene spesso indicato anche come *software* libero (*free software*), anche se questo termine non deve essere frainteso o generare confusione con il *software* di pubblico dominio (che non ha licenza). La volontà di un autore di distribuire il proprio lavoro assieme ai sorgenti e di renderlo accessibile alla comunità degli sviluppatori viene infatti tutelata dalle licenze *Open Source* con delle limitazioni abbastanza pesanti, la cui non osservanza costituisce la violazione di un documento con valore legale ed è pertanto perseguibile.

Non è ad esempio possibile far diventare un *software Open Source* un prodotto commerciale oppure utilizzarne delle parti all'interno di un prodotto che non sia a sua volta *Open Source*. Questo evita che una *software house* si appropri indebitamente di parti di un programma *Open Source* per avere parte del lavoro già fatto da altri senza dare nulla in cambio alla collettività. Non si tratta comunque di una licenza politica, in quanto le sue finalità sono innanzitutto pratiche. La possibilità di apportare modifiche al *software* o di includerne delle parti all'interno di un proprio programma offre infatti parecchi vantaggi, sia per lo sviluppatore, il quale può semplificare il proprio lavoro utilizzando contributi scritti da altri, sia per l'utilizzatore, il quale, avendo accesso al codice sorgente, può adattare (o farsi adattare) il *software* alle proprie esigenze senza dover sottostare a vincoli e tempi di un unico fornitore.

Le peculiarità del *software Open Source* hanno creato un nuovo modello di sviluppo del *software*, basato sul contributo parallelo allo stesso progetto da parte di una ampia base di programmatori coordinati fra di loro. Questo permette di ridurre notevolmente i tempi di sviluppo e test di un *software*, con una migliore qualità del prodotto finale. Ne è riprova il fatto che i maggiori *software* attualmente disponibili per il mondo *UNIX* (*Linux*, *Apache*, *Sendmail*, Samba, X *Window System*, *Gnome*, KDE, Mozilla, ... solo per citare i più conosciuti), sono sviluppati secondo il metodo *Open Source*.

Caratteristiche di Linux - Libero è diverso da gratuito

Un effetto collaterale delle licenze *Open Source* è che il *software* sviluppato secondo esse è copiabile e distribuibile liberamente, nonchè installabile su un numero qualunque di macchine, in maniera assolutamente legale e senza necessità di acquistare più copie dello stesso prodotto. Questo non significa che ad esempio il produttore di una distribuzione di *Linux* non possa farsi pagare l'eventuale valore aggiunto, come le procedure di installazione, il *packaging* o eventuali programmi commerciali distribuiti assieme al prodotto.

La gratuità del *software*, pur essendo un indubbio vantaggio, non dovrebbe essere considerata la discriminante più importante fra un prodotto *Open Source* ed uno commerciale. Le licenze *Open Source* stanno infatti cambiando il mercato del *software* verso un modello in cui il guadagno non deriva più dal vendere licenze, bensì dal fornire supporto, servizi e professionalità.

Caratteristiche di Linux - Avvertenze e riferimenti

Il fatto che la maggior parte del *software* disponibile su *Linux* sia coperto dalla licenza *GNU* GPL non significa che questa sia una regola applicabile a tutti i programmi. Essa non è infatti l'unica licenza di tipo *Open Source* (che è un termine generale per indicare del *software* il cui codice sorgente sia visibile e modificabile), ma ne esistono altre con vincoli diversi.

Alcuni distributori inoltre sono soliti introdurre nelle proprie versioni di *Linux* anche del *software* commerciale, che ovviamente non gode dei vantaggi di quello *Open Source*, in particolare non è possibile ridistribuirlo. Data la varietà del *software* disponibile su una distribuzione di *Linux*, è sempre buona norma prima di compiere operazioni

importanti (ad esempio copiarlo o incorporarlo in lavori derivati) verificarne attentamente caso per caso i termini di licenza.

Networking in Linux - Introduzione

Linux è compatibile con una varietà di *hardware* e supporta differenti protocolli e funzionalità di rete.

Per maggiori informazioni si possono consultare i seguenti documenti del *Linux Documentation Project* di interesse generale, nonché quelli che verranno consigliati trattando i singoli argomenti.

- Autori Vari, *Hardware-HOWTO*
- Autori Vari, *Ethernet-HOWTO*
- Autori Vari, *NET3-4-HOWTO*
- *O.Kirk e T.Dawson*, *The Linux System Administrator Guide, Second edition*

Le Guide di *Linux* e gli *HOWTO* sono disponibili sul sito del ***Linux Documentation Project***.

Networking in Linux - Protocolli di rete

TCP/IP

Il supporto per lo *stack* di protocolli TCP/IP è presente fin dalle prime versioni del sistema operativo. Quella offerta da *Linux* è probabilmente una delle implementazioni più affidabili e versatili ed è uno degli elementi che hanno decretato il successo di questo sistema operativo. Dalle versioni 2.2 del *kernel* è supportata anche la famiglia di protocolli TCP/IPv6. Per informazioni a riguardo si faccia riferimento al documento ***Linux* IPv6 *HOWTO***.

Linux supporta collegamenti punto-a-punto mediante i protocolli **PPP** (*Point-to-Point-Protocol*), SLIP (*Serial Line IP*) e PLIP (*Parallel Line IP*), il quale consente il collegamento fra due *computer* utilizzando la porta parallela, con velocità fino a 20kB/s. Per maggiori informazioni si consulti il documento PPP *HOWTO*.

IPX/SPX/NCP

È il protocollo proprietario utilizzato dai sistemi *Novell NetWare*. *Linux* può condividere *file* e stampanti in una rete *Novell* sia come *client* che come *server*, inoltre sono supportate le funzionalità di *router* e *bridge* IPX ed è possibile interconnettere reti IPX mediante il *tunneling* di pacchetti IPX sopra il protocollo IP.

La versione commerciale di *Linux* Caldera offre supporto commerciale per un *client NetWare* realizzato da *Novell*, il quale, oltre a consentire l'accesso a *fileserver Novell*, offre funzionalità di *NetWare Directory Service* (NDS).

Per maggiori informazioni si faccia riferimento al documento IPX *HOWTO*.

AppleTalk

Netatalk è una implementazione per i sistemi *UNIX* dei protocolli *AppleTalk*, che permette l'integrazione in reti di *computer Apple* e la condivisione di *file* e stampanti secondo il protocollo *AppleShare*. *Linux* 2.4 inoltre offre, compresi nel *kernel*, il supporto per il *filesystem* HFS ed una implementazione del protocollo *AppleTalk* con funzionalità di *routing*.

Per maggiori informazioni sul *software Netatalk* si faccia riferimento ai siti:

http://thehamptons.com/anders/netatalk/

http://www.umich.edu/~rsug/netatalk/

SMB (*NetBIOS*)

I protocolli delle reti *Microsoft* vengono implementati in *UNIX* mediante il programma Samba, che permette la condivisione di dischi e stampanti verso una rete di PC e offre funzioni di *WINS server* e *Primary Domain Controller*.

Nel *kernel* di *Linux* è inoltre compreso il supporto nativo per i *filesystem* condivisi da macchine *Windows* (smbfs). Per maggiori informazioni si consultino il SMB *HOWTO* e il sito **http://www.samba.org/**.

WAN *Networking*: ISDN, X.25, *Frame-relay*, ATM, ...

Il *kernel* di *Linux* fornisce di serie **Isdn4*Linux***, un completo sottosistema ISDN, compatibile con l'*hardware* e con i protocolli più utilizzati (V.110, V.120, X.75, audio, ...).

Molti produttori offrono prodotti *hardware* utilizzabili per trasformare una macchina *Linux* in un *router* WAN (E1, E3, X.25, *Frame Relay*, ...). Assieme ad essi generalmente viene anche fornito il *software* che implementa i protocolli necessari.

È inoltre disponibile il supporto per **ATM** e per ADSL. Per maggiori informazioni sull'utilizzo di *Linux* come *router* WAN si consulti il sito **http://www.secretagent.com/networking/wan.html**

Networking in Linux - Funzionalità avanzate di rete

L'ampia disponibilità di funzionalità e protocolli di rete, unita alle possibilità di personalizzazione ed alla capacità di funzionare su *hardware* limitato, fanno di *Linux* un sistema operativo ideale per creare sistemi *embedded* in grado di sostituire efficacemente *router*, *bridge* ed altri apparati di rete.

Sono disponibili diverse mini-distribuzioni progettate appositamente per questo scopo e sufficientemente ridotte da poter essere installate su un *floppy disk* o su una memoria *Flash* di capacità limitata. Fra le distribuzioni di questo tipo disponibili liberamente vale la pena segnalare ***Freesco*** e ***Linux Router Project***.

È possibile utilizzare le funzioni presenti nel *kernel* per realizzare le seguenti funzioni avanzate:

***Routing* avanzato**

Le versioni recenti del *kernel* di *Linux* offrono funzioni avanzate di *routing* (gestione di più tabelle di *routing*, *policy routing* basato sugli indirizzi IP del mittente o del destinatario o sul contenuto dei pacchetti, gestione delle code e *traffic shaping*, ...) e supportano il *routing multicast* mediante il programma `mrouted`. Mediante opportuni *software*, ad esempio **Zebra**, è inoltre possibile utilizzare i protocolli di *routing* più diffusi come RIP, OSPF e BGP4.

Bridging

Il *kernel* di *Linux* supporta la funzionalità di *bridge*, che, instradando opportunamente i *frame Ethernet*, consente di fare apparire una rete segmentata come se si trattasse di un'unica rete. Grazie al supporto per l'algoritmo di *spanning tree* IEEE802.1, un *bridge* basato su *Linux* può cooperare con apparati di altre marche per formare una rete *bridged* di dimensioni maggiori. È inoltre possibile, sfruttando le funzioni di *firewall* interne al *kernel*, associare alle funzioni di *bridge* dei filtri basati su indirizzi MAC, IP o IPX.

Per ulteriori informazioni si leggano il *Bridge+Firewall-HOWTO* e il *Bridge-HOWTO*.

***Firewall* e *routing* avanzato**

Il *kernel* di *Linux* offre funzioni avanzate di filtraggio di pacchetti (`ipchains` nella versione 2.2, `iptables` nella versione 2.4), che possono essere utilizzate per creare dei *firewall* di tipo *stateful* oppure per manipolare i pacchetti in transito per il sistema. Alcune delle applicazioni possibili sono le seguenti:

- realizzazione di *firewall stateful* basati sull filtraggio dei pacchetti;
- *traffic shaping* e gestione avanzata della priorità nel *routing* dei pacchetti;
- modifica del contenuto dei pacchetti in transito (IP *masquerading*, NAT, ...);
- IP *accounting* e statistiche;
- *port forwarding*;
- *transparent proxy*;
- *load balancing*;
- *tunneling* (*mobile IP*, *virtual private networks*);

Oltre alle funzionalità interne al *kernel*, esistono altri pacchetti *software* che implementano funzioni di *firewall* (TIS, SOCKS) o di *proxy* (Squid) per i protocolli più diffusi come HTTP, FTP, telnet, POP3, H.323, ...

Network management

Per *Linux* è disponibile una implementazione del protocollo SNMP (*Simple Network Management Protocol*), che consente il monitoraggio remoto di apparati di rete (*routers*, *bridges*, ...). Per maggiori informazioni si può fare riferimento al sito: **http://linas.org/linux/NMS.html**.

È inoltre disponibile una vasta gamma di strumenti per la gestione delle problematiche di rete (configurazione, analisi del traffico, ...) e della sicurezza in rete (*intrusion detection*, ...).

Tunneling*, *mobile IP* e *virtual private networks

Il *kernel* di *Linux* permette il *tunneling*, ovvero l'incapsulamento di un protocollo di rete all'interno di un altro. Ad esempio è possibile collegare fra loro via Internet due reti IPX mediante un tunnel di pacchetti IPX all'interno di pacchetti IP.

Il *tunneling* di pacchetti IP all'interno di altri pacchetti IP, permette ad esempio di collegare fra loro due reti in una VPN o di effettuare il *routing* di pacchetti *multicast* su reti non predisposte per tale metdo di indirizzamento. Inoltre facilita l'utilizzo di un *computer* portatile in *roaming* su un'altra rete (*mobile IP*).

Per la realizzazione di VPN, oltre alle semplici **funzioni di *tunneling* presenti nel *kernel***, è possibile utilizzare anche i protocolli standard IPSEC e PPTP (*Point-to-Point Tunneling Protocol*). Per quest'ultimo sono disponibili sia un ***client* PPTP**, che il programma **PoPToP**, che implementa la parte *server*, PopToP.

Per maggiori dettagli si faccia riferimento al **VPN mini-*HOWTO***.

Networking in Linux - Interfacce di rete TCP/IP

Nei sistemi *UNIX*, con il termine interfaccia di rete viene indicata una entità logica in grado di scambiare traffico TCP/IP a cui è assegnato un indirizzo di rete, che viene utilizzato come *source address* per tutti i pacchetti IP uscenti da tale interfaccia, a cui può o meno essere associato un corrispondente dispositivo fisico.

Una macchina non ha pertanto un unico indirizzo di rete, bensì potenzialmente tanti quante sono le interfacce di rete attive, o anche un numero maggiore, essendo possibile assegnare alla stessa interfaccia più indirizzi IP.

Un esempio di interfaccia di rete a cui non è associato un dispositivo fisico è l'interfaccia virtuale lo, che permette il collegamento in *loopback* alla macchina locale (ad essa viene generalmente assegnato l'indirizzo standard 127.0.0.1, che prende il nome di indirizzo di *loopback*).

Ad ogni tipologia di interfaccia di rete viene assegnato dal *kernel* un nome simbolico, ad esempio eth per le interfacce associate alle schede *Ethernet*. Se nel sistema sono presenti più interfacce del medesimo tipo, esse vengono numerate in ordine progressivo partendo da zero. Avremo pertanto eth0, eth1, eth2, ... eccetera, in cui la numerazione progressiva avviene in base al *MAC address* delle schede *hardware*.

Alcune interfacce di rete comunemente usate sono le seguenti:

- *loopback*: lo;
- *ethernet*: eth;
- associate a collegamenti PPP: ppp;
- ISDN: isdn;
- associate a collegamenti PPP sincroni su ISDN: ippp.

Si noti che, al contrario degli altri dispositivi, il nome delle interfacce di rete viene creato direttamente dal *kernel* e non esiste pertanto un corrispondente *file* speciale in `/dev`.

SAMBA

Tra le funzioni dei principali sistemi operativi di rete la *suite software* SAMBA implementa in *UNIX* il protocollo SMB (*Server Message Block*). SMB è un protocollo utilizzato per condividere risorse *hardware* (*file*, stampanti, perifieriche in genere) e risorse *software* di un sistema di elaborazione (*pipe*, *mailbox*, eccetera), ed è il più diffuso nell'integrazione di un sistema *UNIX* all'interno di una rete in tecnologia *Windows*.

L'idea del protocollo SMB nasce in ambito IBM verso la metà degli anni '80 e successivamente è stata ripresa da *Microsoft* fin dal 1987. Il protocollo è stato successivamente sviluppato ulteriormente da *Microsoft* e altri.

I dati SMB possono essere incapsulati e trasportati in datagrammi TCP/IP, oppure NetBEUI e IPX/SPX. Nello schema grafico, si evidenzia la posizione del protocollo SMB nell'ambito dell'architettura di comunicazione.

<table>
<tr><th>Livelli</th><th>OSI</th><th colspan="4">SAMBA</th><th>TCP/IP</th></tr>
<tr><td>Livello 7</td><td>Application</td><td colspan="4" rowspan="2">SMB</td><td rowspan="3">Application</td></tr>
<tr><td>Livello 6</td><td>Presentation</td></tr>
<tr><td>Livello 5</td><td>Session</td><td>NetBIOS</td><td rowspan="3">NetBEUI</td><td>NetBIOS</td><td>NetBIOS</td></tr>
<tr><td>Livello 4</td><td>Transport</td><td rowspan="2">IPX</td><td rowspan="2">DECNet</td><td>TCP/UDP</td><td>TCP/UDP</td></tr>
<tr><td>Livello 3</td><td>Network</td><td>IP</td><td>IP</td></tr>
<tr><td>Livello 2</td><td>Link</td><td>802.2, 802.3, 802.5</td><td>802.2, 802.3, 802.5</td><td>Ethernet V2</td><td>Ethernet V2</td><td>Ethernet or other</td></tr>
<tr><td>Livello 1</td><td>Physical</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

Il boot di un sistema Linux - Device drivers ed organizzazione modulare del kernel di Linux

In *Linux*, oltre ai *driver* delle periferiche, sono state rese modulari e caricabili a *run time* quasi tutte le funzionalità aggiuntive del *kernel*, come il supporto per i *filesystem* o per le funzioni di rete. La strutturazione per moduli permette di ridurre al minimo la dimensione del *kernel* e di caricare in memoria solamente le funzionalità necessarie in un dato momento.

Molti moduli vengono forniti di serie assieme al *kernel* di *Linux* nella *directory* `/lib/modules/versione_del_kernel/`, mentre altri possono essere forniti da terze parti.

Un modulo non è altro che un segmento di codice oggetto (suffisso .o) che per divenire a tutti gli effetti parte del codice del *kernel* deve essere opportunamente linkato ad esso. Tale operazione viene generalmente effettuata mediante i comandi `insmod` o `modprobe`.

La differenza fra `insmod` e `modprobe` consiste nel fatto che il secondo gestisce automaticamente le dipendenze fra i moduli. Infatti il funzionamento di alcuni moduli può dipendere dalla disponibilità di altre funzioni, che pertanto devono essere già caricate in memoria. Ad esempio il *driver* per la scheda di rete NE2000 (ne.o) per funzionare necessita che sia stato prima caricato in memoria il supporto per il *chip* 8390 (8390.o):

```
# insmod /lib/modules/2.2.12-20/net/ne.o
/lib/modules/2.2.12-20/net/ne.o: unresolved symbol ei_open
/lib/modules/2.2.12-20/net/ne.o: unresolved symbol ethdev_init
/lib/modules/2.2.12-20/net/ne.o: unresolved symbol ei_interrupt
/lib/modules/2.2.12-20/net/ne.o: unresolved symbol NS8390_init
/lib/modules/2.2.12-20/net/ne.o: unresolved symbol ei_close
```

È pertanto necessario caricare i due moduli nella sequenza corretta:

```
# insmod /lib/modules/2.2.12-20/net/8390.o
# insmod /lib/modules/2.2.12-20/net/ne.o
```

Utilizzando `modprobe` il caricamento dei moduli necessari viene automatizzato, utilizzando una tabella contenente le dipendenze fra i diversi moduli, `modules.dep`, che viene creata dal comando `depmod -a` in uno degli *script* di avvio del sistema.

```
# modprobe ne.o
```

I messaggi di errore o di *logging* generati da un modulo possono essere letti mediante il comando `dmesg`:

```
# dmesg
...
3c59x.c:v0.99H 11/17/98 Donald Becker
http://cesdis.gsfc.nasa.gov/linux/drivers/vortex.html
eth0: 3Com 3c900 Boomerang 10Mbps Combo at 0x6400,
00:60:97:ac:34:a4, IRQ 12
8K word-wide RAM 3:5 Rx:Tx split, autoselect/10baseT interface.
Enabling bus-master transmits and whole-frame receives.
```

Nonostante la maggior parte dei *driver* sia ormai in grado di autoconfigurarsi, spesso risulta necessario passare ad un modulo dei parametri aggiuntivi. Questi possono essere specificati nella linea di comando di `insmod`:

```
# modprobe /lib/modules/2.2.12-20/net/ne.o io=0x300 irq=5
```

I parametri accettati dai *driver* per le diverse schede di rete sono descritti nel *file* `Documentation/networking/net-modules.txt` presente all'interno dei sorgenti del *kernel* di *Linux*.

Il boot di un sistema Linux - Caricamento automatico dei moduli

Linux offre anche la possibilità di caricare un *driver* o una funzione aggiuntiva solo al momento del bisogno ed in modo automatico. È inoltre possibile fare in modo che il modulo venga scaricato dalla memoria dopo un determinato periodo di non utilizzo.

Questa funzionalità viene gestita direttamente dal *kernel*, attraverso il programma `kmod`, oppure nelle vecchie versioni di *Linux* mediante il demone `kerneld`.

Così facendo non è più necessario caricare esplicitamente i moduli necessari, ma è sufficiente associare ad ogni funzionalità il modulo che la gestisca, attraverso il *file* `/etc/modules.conf`. Ad esempio, volendo fare in modo che quando si attiva l'interfaccia di rete eth0 venga automaticamente caricato il modulo ne.o, è sufficiente inserire in `modules.conf` le seguenti linee:

```
alias eth0 ne
options ne io=0x300 irq=5
```

Attraverso `kmod` è possibile gestire le funzioni di caricamento dei moduli in modo complesso, ad esempio eseguendo automaticamente degli *script*. Un esempio più complesso di `modules.conf` è il seguente:

```
alias eth0 8139too
alias parport_lowlevel parport_pc
alias usb-controller usb-uhci

alias char-major-81 bttv
pre-install bttv modprobe -k tuner
options tuner type=0 debug=1
options bttv pll=1 radio=0 card=39
alias sound-slot-0 via82cxxx_audio
post-install sound-slot-0 /bin/aumix-minimal -f /etc/.aumixrc -L >/dev/null 2>&1 || :
pre-remove sound-slot-0 /bin/aumix-minimal -f /etc/.aumixrc -S >/dev/null 2>&1 || :
```

Per maggiori informazioni è possibile leggere il *file* `Documentation/modules.txt`, presente all'interno dei sorgenti del *kernel* di *Linux*. I parametri accettati dai *driver* per le diverse schede di rete sono descritti nel *file*:

```
/usr/src/linux/Documentation/networking/net-modules.txt
```

Installazione e configurazione di driver per periferiche di rete

Per quanto riguarda *Windows*2000, come per tutti i sistemi *Microsoft*, l'installazione dei *driver* per le periferiche di rete è abbastanza semplice e occorre semplicemente eseguire il programma di installazione fornito in dotazione con la scheda di rete stessa. Se non si disponesse del *driver* relativo o se lo si ritenesse obsoleto di solito si contatta il sito del produttore dell'*hardware*.

Per quanto riguarda i sistemi *Linux* è necessario caricare nel *kernel* l'opportuno modulo contenente il *driver*. Alcuni *kernel* hanno già compilato all'interno i principali *driver* di rete cosicché siano disponibili al momento dell'installazione, in questo caso non occorre fare nulla e all'avvio del sistema sarà disponibile il dispositivo relativo all'*hardware* installato.

In caso contrario ogni specifico *hardware* ha un proprio modulo incluso nella distribuzione dei sorgenti del *kernel*. È bene consultare gli **HOW-TO** relativi al *driver* in questione per poter fornire le opzioni specifiche al momento del caricamento.

Dopo aver caricato il modulo opportuno il dispositivo è utilizzabile e pronto per essere configurato.

Interfacciamento con la rete - Configurazione di una interfaccia di rete

Il comando generalmente utilizzato negli *script* di avvio per configurare ed attivare una interfaccia di rete è:

```
# ifconfig
```

Non sempre il suo utilizzo è obbligatorio: ad esempio le interfacce di rete associate ai link PPP (ppp0, ppp1, ...)

vengono attivate direttamente dal demone `pppd` che sovrintende alla connessione.

Se il comando `ifconfig` viene usato senza parametri, permette di ottenere lo stato delle diverse interfacce attive sulla macchina:

```
# ifconfig
eth0 Link encap:Ethernet HWaddr 00:60:97:AC:34:A4
inet addr:193.43.98.1 Bcast:193.43.98.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:440578 errors:0 dropped:0 overruns:0 frame:0
TX packets:2363317 errors:0 dropped:0 overruns:0 carrier:0
collisions:503318 txqueuelen:100
Interrupt:12 Base address:0x6400
ppp0 Link encap:Point-to-Point Protocol
inet addr:213.255.24.25 P-t-P:213.255.6.251 Mask:255.255.255.255
UP POINTOPOINT RUNNING NOARP MTU:1500 Metric:1
RX packets:93509 errors:0 dropped:0 overruns:0 frame:0
TX packets:101886 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:30
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:3924 Metric:1
RX packets:122056 errors:0 dropped:0 overruns:0 frame:0
TX packets:122056 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
```

In questo caso nel sistema sono presenti l'interfaccia di *loopback* (lo), una scheda *Ethernet* (eth0) ed un collegamento PPP (ppp0). I valori presenti nell'*output* di `ifconfig` hanno il seguente significato:

- *Link encap*: il tipo di interfaccia (*Ethernet* denota che si tratta di una interfaccia *Ethernet*, mentre *Point-to-Point Protocol* che si sta utilizzando il PPP).
- *HWaddr*: nel caso di schede *Ethernet*, contiene l'indirizzo MAC della scheda.
- *Inet addr*: l'indirizzo IP associato alla interfaccia. Nel caso di collegamento punto-a-punto viene indicato anche l'indirizzo presentato all'altro capo della connessione (P-t-P).
- *Bcast*: l'indirizzo di *broadcast* (ha senso solo nel caso di interfacce col parametro *BROADCAST* attivo).
- *Mask*: la *netmask* associata alla interfaccia (nel caso di collegamenti punto-a-punto fra due macchine, essa vale 255.255.255.255).
- *Flags*: le caratteristiche dell'interfaccia e lo stato della stessa. Nell'esempio, l'interfaccia eth0 risulta attualmente attiva (*UP RUNNING*), collegata ad un medium capace di trasmettere pacchetti IP in *broadcast* (*BROADCAST*) ed in *multicast* (*MULTICAST*) e con dimensione massima dei pacchetti trasmissibili (MTU) pari a 1500 *bytes*. Il parametro NOARP nel caso della interfaccia Punto-a-punto indica che su questa interfaccia non viene gestito il protocollo ARP. Il campo *Metric* viene infine utilizzato per definire la priorità della interfaccia nelle decisioni di *routing*.
- Statistiche: le ultime linee indicano delle statistiche sui pacchetti (*frame*) trasmessi e ricevuti e su eventuali errori.

A proposito del campo *flags* è doveroso fare due brevi richiami:

Le interfacce *ethernet*, oltre a ricevere i pacchetti IP destinati al proprio indirizzo (*unicast*), ricevono anche i pacchetti con destinatario l'indirizzo di *broadcast* della rete. In questo modo è possibile spedire un pacchetto di dati in modo che venga ricevuto da tutte le macchine presenti su uno stesso segmento di rete.

In aggiunta è possibile configurare l'interfaccia per ricevere anche i pacchetti destinati ad un gruppo chiuso di macchine (*MULTICAST*). Tale funzionalità viene sfruttata ad esempio per realizzare trasmissioni multimediali verso più utenti senza dover realizzare un collegamento dedicato per ogni macchina.

Il parametro MTU indica la massima dimensione di un pacchetto trasmissibile su una determinata interfaccia. Le interfacce *Ethernet* hanno in ogni caso un valore di MTU pari a 1500, imposto dall'*hardware* e derivante dalla dimensione dei *frame Ethernet*. Eventuali pacchetti di dati di dimensioni maggiori della MTU possono essere comunque trasmessi, ma prima devono essere deframmentati ed in seguito riassemblati (tali operazioni vengono gestite automaticamente dallo *stack* IP del sistema operativo).

Per configurare manualmente una interfaccia *Ethernet* si utilizza il comando:

```
ifconfig eth0 <ip_address> altri_parametri ...
```

Questo comando assegna un indirizzo IP all'interfaccia e la attiva. Se non vengono esplicitamente specificati dei valori, per gli altri vengono utilizzati dei valori di *default*. La *netmask* e l'indirizzo di *broadcast* vengono calcolati automaticamente in base alla classe dell'indirizzo. Ad esempio se l'indirizzo appartiene ad una rete di classe C, la *netmask* viene posta a 255.255.255.0.

Un esempio di configurazione della interfaccia eth0 è il seguente:

```
ifconfig eth0 192.168.1.2 netmask 255.255.255.0 broadcast 192.168.1.255 up
```

Per maggiori dettagli e per le altre opzioni utilizzabili nel comando `ifconfig` si faccia riferimento alla pagina del manuale in linea.

Una volta configurata l'interfaccia è necessario creare un opportuno percorso di instradamento (*route*) verso gli indirizzi della rete raggiungibili mediante essa. Nelle versioni recenti di *Linux* il *kernel* crea automaticamente tale *route* quando si configura l'interfaccia. In caso contrario è possibile farlo a mano mediante il comando `route`:

```
route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0
```

Le nuove versioni di *route* sono in grado di calcolare automaticamente eventuali valori non specificati e l'interfaccia da utilizzare nel caso non la si indichi nella linea di comando, pertanto si può semplificare il comando precedente in:

```
route add -net 192.168.1.0
```

Impartendo a mano i comandi di configurazione appena visti, essi non vengono salvati. Occorre pertanto inserirli all'interno di un *file* di inizializzazione del sistema. A meno di situazioni particolari, di solito non è necessario usare esplicitamente il comando `ifconfig` o creare a mano un *file* di avvio, in quanto la sequenza di operazioni necessarie viene svolta al *boot* da uno degli *script* di inizializzazione del sistema (in `/etc/rc.d`). Nelle distribuzioni moderne di *Linux* esistono degli strumenti di configurazione che permettono una gestione semplice della configurazione delle interfacce di rete. Purtroppo i meccanismi usati non sempre sono standard fra le varie distribuzioni di *Linux*.

Interfacciamento con la rete - Indirizzamento con DHCP

In una rete contenente un numero elevato di nodi risulta macchinoso assegnare manualmente gli indirizzi IP a ciascuna macchina. La maggior parte delle distribuzioni di *Linux* offrono la possibilità di configurare una interfaccia di rete utilizzando i protocolli **BOOTP** (*Boot Protocollo*) oppure **DHCP** (*Dynamic Host Configuration Protocol*), che consentono di centralizzare l'assegnazione degli indirizzi attraverso l'uso di un apposito *server*.

Allo scopo viene utilizzato il demone `dhcpcd`, il quale si occupa di richiedere al *server* l'indirizzo. Ad esso è associato un **periodo di validità (*lease*)**, scaduto il quale il demone, che pertanto deve rimanere attivo, si occupa di richiedere un nuovo indirizzo.

Poiché al momento del *boot* la macchina non dispone ancora di un indirizzo IP, la richiesta al *server* viene fatta mediante un *frame Ethernet* spedito in modalità *broadcast*.

Mediante BOOT o DHCPD è possibile ottenere anche altri tipi di informazioni, ad esempio il nome della macchina o del dominio di appartenenza e gli indirizzi dei *server* DNS da utilizzare per la risoluzione dei nomi. I dati ottenuti da `dhchcp` per le diverse interfacce vengono salvati in un *file* `/etc/dhcpc/dhcpcd-<nome_interfaccia>.info`.

Quando `dhcpcd` ottiene dei dati da un *server*, esso esegue lo *script* `/etc/dhcpc/dhcpcd-eth0.exe`, che contiene i comandi necessari a riconfigurare l'interfaccia con il nuovo indirizzo e ad aggiornare il sistema con i valori ricevuti.

Per ulteriori informazioni si faccia riferimento al DHCP Mini *HOWTO*.

Interfacciamento con la rete - IP aliasing

Mediante `ifconfig` è anche possibile assegnare alla stessa interfaccia fisica più indirizzi IP. Tale funzione è supportata ad esempio nel caso delle schede *Ethernet*, per le quali è possibile creare delle interfacce logiche denominate ethN: M (ad esempio: eth0:1, eth0:2, ...) con indirizzi diversi.

Questo permette di gestire servizi virtuali che rispondano ad indirizzi diversi sulla stessa macchina. Ad esempio è possibile configurare un *server Web* in modo che presenti pagine diverse a seconda dell'indirizzo utilizzato per accedervi.

Per configurare un indirizzo IP secondario su una interfaccia *Ethernet* si utilizza il comando `ipconfig` con la seguente sintassi:

```
ifconfig eth0:1 192.168.1.23 netmask 255.255.255.0 up
```

Se non esiste già, è necessario ricordarsi di creare una *route* verso l'indirizzo della nuova interfaccia:

```
route add 192.168.1.23 dev eth0:1
```

Configurare il routing per la gestione dei pacchetti di una rete locale verso Internet

L'operazione di *routing* consiste nelle decisioni che il *kernel* deve intraprendere per selezionare l'interfaccia di rete mediante cui inviare un pacchetto IP destinato ad un determinato indirizzo: ad esempio un pacchetto destinato ad una macchina appartenente allo stesso segmento di rete *ethernet* di una delle interfacce del sistema viene spedito direttamente al destinatario.

Nel caso il destinatario non sia raggiungibile direttamente, il pacchetto di dati può essere dato in consegna ad un *router* che si occupi del suo inoltro. Questo dispone di una interfaccia sulla stessa rete della macchina e di una interfaccia su un'altra rete. In questo modo, passando per uno o più *router*, è possibile far giungere il pacchetto IP alla destinazione desiderata. Se nella rete locale non esiste una *router* attraverso cui è possibile raggiungere la macchina destinataria di un pacchetto, questo viene scartato. Generalmente viene definito un *router* a cui spedire tutti i pacchetti non recapitabili attraverso altre strade, che prende anche il nome di *default gateway*.

Per *default*, nelle versioni del *kernel* di *Linux* superiori alla 2.2, il *forwarding* di pacchetti IP, ovvero il passaggio per una macchina di pacchetti generati da altre macchine e non diretti ad essa, non è abilitato.

Volendo utilizzare il proprio sistema come un *router*, il *forwarding* deve essere attivato in modo esplicito mediante il seguente comando:

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

Si tratta di un esempio significativo di come sia possibile utilizzare il proc *filesystem* per modificare il funzionamento del *kernel* a *run time*. Altri esempi di *tuning* del *kernel* sono documentati nel *file* `net/TUNABLE` fornito con i sorgenti del *kernel* di *Linux*.

Nei sistemi *Windows* si può abilitare la stessa funzione accedendo a *Start* -> Impostazioni -> Pannello di controllo -> Strumenti di amministrazione -> Servizi -> *Routing* e Accesso remoto. Selezionando le Proprietà con il tasto destro del *mouse* ed abilitando il modo Automatico nella casella Tipo di Avvio.

Per rendere effettive le modifiche apportate, cliccare il pulsante Applica.

Nel caso più banale la decisione sull'instradamento viene effettuata semplicemente confrontando l'indirizzo di destinazione del pacchetto con il contenuto di una tabella di *routing* statico. In reti complesse è possibile utilizzare uno dei protocolli di *routing* dinamico (RIP, OSPF e BGP4, ...) oppure utilizzare le funzioni di filtraggio di pacchetti interne al *kernel*, per decidere i percorsi in base a criteri più complessi, come la porta TCP o UDP sorgente o di destinazione, il tipo di protocollo oppure la qualità di servizio desiderata (QoS).

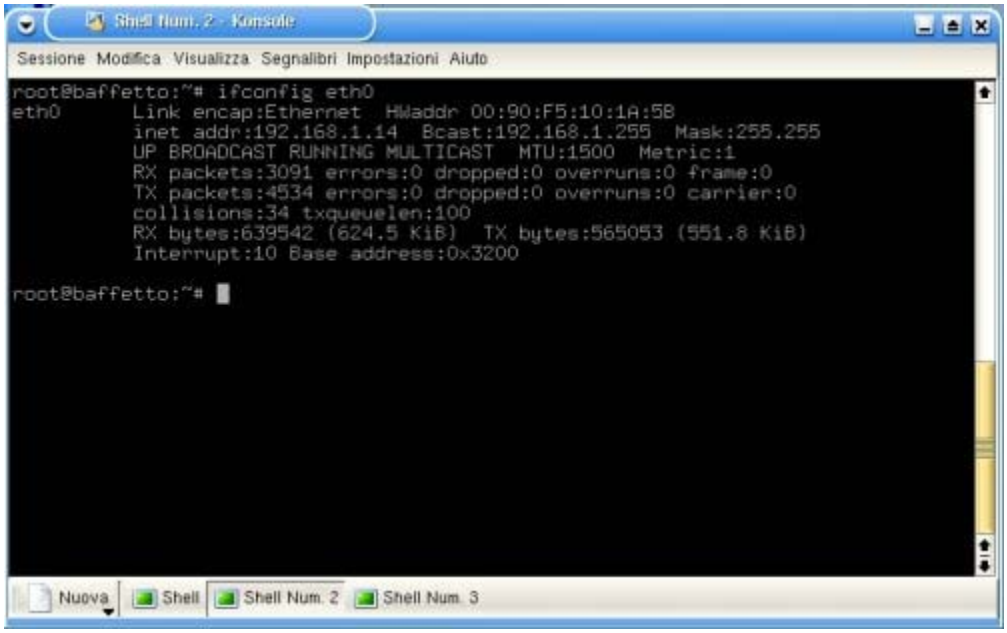
Il comando netstat

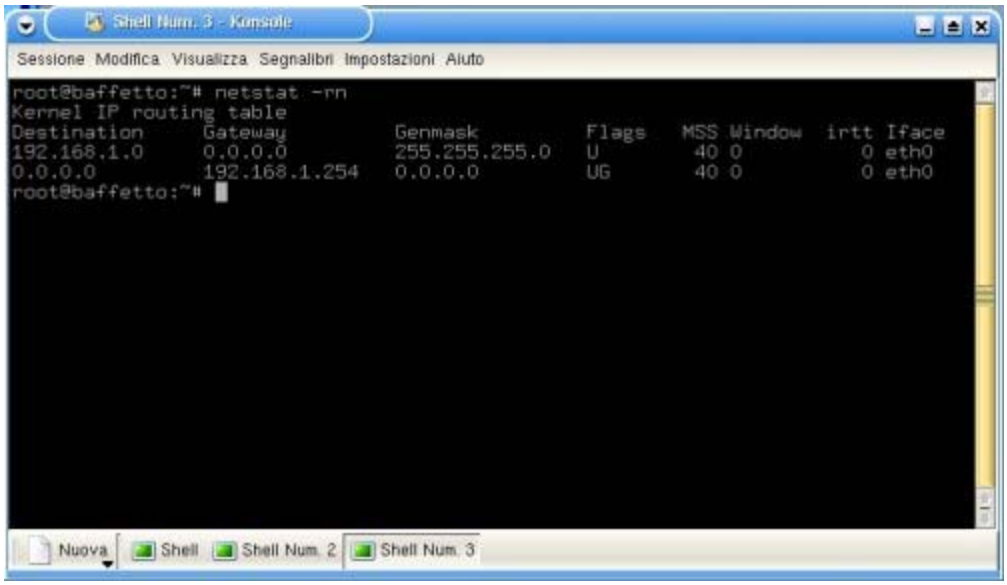
Per visualizzare il contenuto corrente della tabella di *routing* si utilizza per tutti e due gli ambienti operativi il comando `netstat`.

Riportiamo come esempio la tabella di *routing* di una macchina con una interfaccia *Ethernet* sulla rete di classe C 192.168.1.0.
Il suo indirizzo IP è 192.168.1.14 e viene utilizzato come instradamento di *default* per il traffico locale. Nella rete è presente un *router* all'indirizzo 192.168.1.254.

Shell di *Linux*:

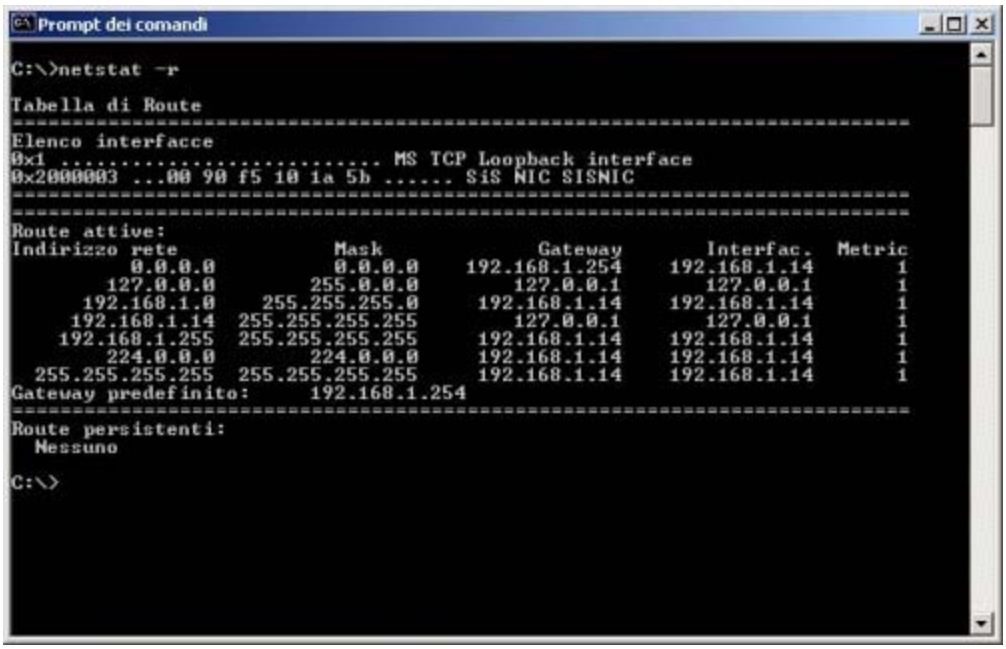
- `netstat -h` fornisce un *help* in linea con la sintassi del comando.
- `netstat -r` stampa il contenuto della tabella di *routing*.
- `netstat -rn` stampa il contenuto della tabella di *routing* senza che gli IP vengano risolti come nomi simbolici.





Prompt di MS-DOS di *Windows*2000:

- `netstat /?` fornisce un *help* in linea con la sintassi del comando.
- `netstat -r` stampa il contenuto della tabella di *routing*.



L'output del comando netstat

I valori significativi che appaiono nell'*output* di `netstat` sono i seguenti:

- *Destination* o Indirizzo rete: identifica la rete di destinazione della *route*. Questo valore per avere significato deve essere letto assieme al valore della *netmask* (*Genmask* o *Mask*).
- *Gateway*: indica l'eventuale *router* da utilizzare per raggiungere la destinazione specificata.

La tabella di *routing* contiene un insieme di percorsi di instradamento (*route*), che possono essere di diverso tipo:

- una *route* di tipo H (*host*) identifica un percorso utilizzato per raggiungere un determinato *host* (*flag* H);
- una *network route* specifica invece il percorso da utilizzare per raggiungere tutte le macchine di una rete specificata (il campo *Genmask* contiene la *netmask* associata alla rete);
- una *default route* specifica quale interfaccia (o *gateway*) utilizzare come ultima risorsa se non sono stati trovati degli altri percorsi validi.

Un instradamento può fare riferimento ad una interfaccia oppure passare attraverso un *gateway* (identificato dal *flag* G).

Per interagire con la tabella, creare nuovi instradamenti o eliminare quelli esistenti, si utilizza il comando *route*.

NOTA: nel caso sopra descritto, non è necessario aggiungere altre *route* statiche, in quanto nella rete è presente un *router* che è in grado di instradare i pacchetti non diretti alla rete locale.

Esempi di utilizzo del comando route nella shell di Linux

Aggiungere una *route* statica per l'instradamento verso la rete 192.168.50.0/24, attraverso l'indirizzo IP 192.168.1.253:

```
route add -net 192.168.50.0 netmask 192.168.50.255 gw 192.168.1.253
```

Cancellare una *route* statica per l'instradamento verso la rete 192.168.50.0:

```
route del -net 192.168.50.0
```

Creare una *route* di *default* utilizzando un *default gateway*:

```
route add default gw 192.168.1.254
```

Cancellare una *route* di *default*:

```
route del default gw 192.168.1.254
```

Le versioni recenti del comando `ifconfig` aggiungono automaticamente delle *route* appropriate per raggiungere le interfacce che vengono configurate, inoltre per rendere permanenti le configurazioni del *routing* di *Linux* occorre modificare gli *script* situati in `/etc/sysconfig/network-scripts/` in *RedHat Linux*.

Esempi di utilizzo del comando route nella finestra prompt di ms-dos di Windows2000

Aggiungere una *route* statica per l'instradamento verso la rete 192.168.50.0/24, attraverso l'indirizzo IP 192.168.1.253:

```
route ADD 192.168.50.0 MASK 255.255.255.0 192.168.1.253
```

Cancellare una *route* statica per l'instradamento verso la rete 192.168.50.0

```
route DELETE 192.168.50.0
```

Creare una *route* di *default* utilizzando un *default gateway*:

```
route ADD 0.0.0.0 MASK 0.0.0.0 192.168.1.254
```

Cancellare una *route* di *default*:

```
route DELETE 0.0.0.0
```

Aspetti gestionali e procedure per gestire più server sulla stessa rete

La parte di gestione di reti e applicazioni è la parte del *networking* che richiede più sforzi per lo sviluppo. Il problema risiede nella poca uniformità delle soluzioni che sono in commercio, dovuto ai diversi approcci utilizzati. Questo rende la gestione uno degli aspetti più delicati e costosi. Per migliorare questa situazione, si sta tentando un approccio di tipo centralizzato, anche se si è lontani da uno standard che possa unificare le soluzioni utilizzate.

In aziende di ogni tipo e dimensione è sempre più essenziale la disponibilità di un efficiente Sistema Informativo Aziendale, la cui dimensione e complessità devono essere ovviamente rapportate alle dimensioni ed alla struttura aziendale.

Quando le dimensioni dell'azienda sono rilevanti (centinaia o migliaia di dipendenti), ed a maggior ragione se l'azienda è anche distribuita territorialmente, le infrastrutture di rete utilizzate come supporto ai trasferimenti di dati nell'ambito del sistema informativo diventano realmente molto complesse. Tali infrastrutture sono prevalentemente

costituite da un numero sempre crescente di reti locali, o LAN (*Local Area Network*), interconnesse tra loro localmente o per mezzo di collegamenti geografici. I responsabili della gestione di tali reti si trovano pertanto a dover gestire sistemi sempre più complessi con risorse umane sempre più limitate. Una delle strade percorribili per ottenere un incremento del grado di efficienza consiste nell'utilizzo di tecnologie che consentano il controllo centralizzato delle strutture di rete e dei servizi realizzati per il loro tramite.

Gli obiettivi da raggiungere, legati alla necessità di gestire le applicazioni che stanno alla base delle attività aziendali, sono essenzialmente i seguenti:

- massimizzare il livello di disponibilità della rete;
- contenere al minimo i costi di gestione;
- ottimizzare le prestazioni e la qualità del servizio fornito;
- identificare per tempo le nuove esigenze al fine di pianificare l'evoluzione della rete.

Alla base della gestione di rete c'è quindi un colloquio tra la stazione di gestione e l'apparato gestito. Tale colloquio si esplica in particolare tra due entità, realizzate per mezzo di processi *software*, denominate rispettivamente *Manager*, nel centro di gestione, ed *Agent*, nel nodo gestito. Il trasferimento di informazioni tra *Manager* ed *Agent* avviene in accordo ad un insieme di regole, sintattiche e semantiche, che costituiscono il protocollo di gestione. Il protocollo di gestione è un protocollo di livello applicativo che si appoggia sulla pila protocollare sottostante.

In ambito **OSI** tale protocollo si chiama **CMIP** (*Common Management Information Protocol*) ed in ambito SNMP si chiama appunto **SNMP** (*Simple Network Management Protocol*).

Installare e configurare software antivirus

Per combattere efficacemente la diffusione dei virus in sistemi non dotati di un controllo efficace sui permessi di lettura e scrittura degli utenti, è bene installare *software* che controllino periodicamente i dati e *file* sensibili del sistema.

Questi *software* sono chiamati Antivirus e vengono aggiornati continuamente per fare fronte alle evoluzioni dei virus esistenti ed alla diffusione di quelli nuovi. Prevalentemente vengono installati su macchine *client* con sistemi *Windows*.

L'installazione non presenta alcuna difficoltà, occorre però mantenere il proprio antivirus aggiornato alle ultime versioni onde evitare di contrarre virus recenti non rilevabili da un antivirus obsoleto.

Citiamo alcuni antivirus tra i più conoscuti:

- *Norton* (**www.symantec.com**)
- *NetShield* (**www.mcafeeb2b.com**)
- *Protector* (**www.pspl.com**)

Client software

Franco Callegati
Paolo Presepi
Riccardo Gori

Installazione dei componenti software di un client di rete

I *computer client* di rete sono generalmente *workstation* su cui girano programmi *client* per i relativi servizi, e che si connettono tramite la rete ai *server* presenti.

Non richiedono particolari caratteristiche *hardware*/*software*, ma necessitano di una configurazione corretta per poter interrogare il *server* ed espletare così il servizio.

La scelta del sistema operativo è legata al programma *client* che si intende eseguire; alcuni di questi sono **multipiattaforma** e possono girare su diversi sistemi operativi, altri invece sono scritti per un unico sistema.

L'installazione dei programmi avviene nella maniera consueta:

Per le distribuzioni *Linux*:

- scaricare ed installare il pacchetto relativo alla distribuzione adottata;
- impostare la configurazione mediante *script* forniti con i sorgenti del programma oppure modificando i *file* di configurazione.

Vedere le documentazioni relative alle varie distribuzioni: **http://www.linux.org/dist/list.html**

Per i sistemi *Windows*:

- eseguire il *setup* di installazione;
- la configurazione generalmente viene richiesta automaticamente ad installazione avvenuta.

Configurazione dei protocolli di rete su di un client

Per il funzionamento dei servizi è fondamentale il supporto e la corretta impostazione dei protocolli di rete incaricati del trasporto dei dati tra *client* e *server*.

Il protocollo maggiormente diffuso è il TCP/IP, ciò non toglie però che *client* specifici possano utilizzarne altri molto diffusi come IPX, NetBEUI, o addirittura protocolli proprietari creati ad hoc per la specifica applicazione.

Il protocollo di rete è un elemento che interagisce fortemente con il *kernel* del sistema operativo. Vediamo la sua installazione nei due ambienti operativi.

Esempi di installazione e configurazione del protocollo di rete TCP/IP: Linux

Controllare se il supporto per il protocollo sia già presente nel *kernel*:

Eseguire `# ls /proc/sys/net/ipv4/`

Il risultato deve mostrare tutti i *file* con le caratteristiche del protocollo, se questo non avviene o se tale *directory* non

esiste, significa che il supporto per il TCP/IP non è presente; in questo caso:

- Abilitare nuovi protocolli di rete nel *kernel*:
  - Scaricare i sorgenti del *kernel* (**www.kernel.org**), scompattarli ed entrare nella *directory* principale di solito nominata *linux*-VERSIONE ed eseguire `# make menuconfig`.
  - Abilitare i protocolli voluti nel sottomenu *Networking options*.
  - Compilare il *kernel*: `# make dep && make clean && make bzImage`.
  - Installare il *kernel* con il comando install*kernel* e riavviare la macchina.

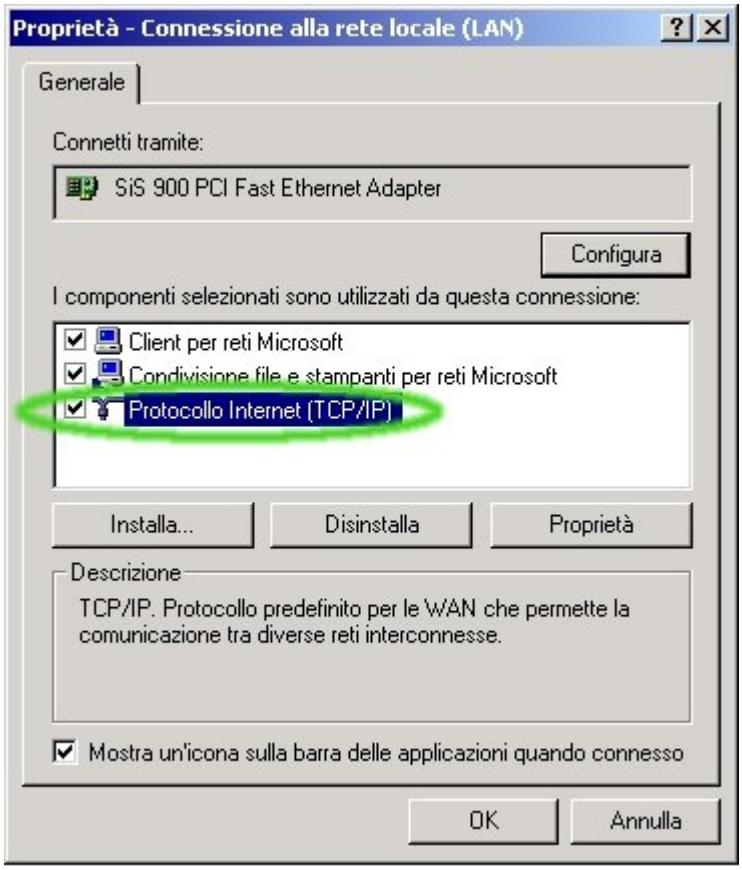
Vedere il *Kernel HOW-TO*: **http://ildp.pluto.linux.it/HOWTO/Kernel-HOWTO.html**

A questo punto i supporti per il protocollo TCP/IP sono presenti ed occorre configurare le interfacce di rete.

Esempi di installazione e configurazione del protocollo di rete TCP/IP: Windows2000

Nel pannello di controllo, selezionare le risorse di rete e connessioni remote e successivamente le proprietà della Connessione alla rete locale LAN.

Nel pannello relativo alle proprietà



accertarsi che sia installato il Protocollo Internet TCP/IP, e in caso contrario installarlo selezionando il tasto Installa e scegliendo il protocollo dalla lista dei protocolli.

Installare e configurare servizi

Franco Callegati
Paolo Presepi
Riccardo Gori

Installazione di applicazioni client/server - Introduzione a Windows 2000 printing

Introduzione a *Windows* 2000 *printing*

- **Terminologia** .
- **Prerequisiti** .

Microsoft Windows 2000 offre all'amministratore tutta una serie di tecnologie e strumenti che rendono molto semplice e flessibile la gestione e configurazione di una stampante condivisa.

Una volta che sia stato configurato un *server* di stampa è possibile configurare *client computer* che eseguono *Windows* 95, *Windows* 98, o *Microsoft Windows* NT 4.0 oltre che, ovviamente, *Microsoft Windows* 2000.

Prima di vedere però come condividere una stampante, come assegnare i permessi di stampa, come installare i *client* ed altri settaggi avanzati, bisogna innanzitutto analizzare la terminologia usata ed i prerequisiti necessari affinché tutto ciò sia realizzabile.

Introduzione a Windows 2000 printing - Terminologia

I seguenti termini definiscono quello che un ambiente di stampa in *Windows* 2000:

- ***Print device***. La periferica *hardware*, cioé la stampante fisica. *Windows* 2000 distingue le due seguenti tipologie di *Print Device*:
  - ***Local print devices***. Periferica *hardware* connessa ad una porta locale del *server* (COM, LPT, ...).
  - ***Network-interface print devices***. Periferica *hardware* connessa al *print server* tramite la rete piuttosto che tramite una porta locale. Richiede una propria scheda di rete ed un proprio indirizzo di rete.
- ***Printer***. Il *software* tramite il quale il sistema operativo dialoga con la stampante.
- ***Print server***. Il *computer* su cui il *printers* e i *drivers* per i *client* risiedono. Tale macchina riceve le richieste di stampa dei *client*, e le processa prima di inviarle alla stampante. Ovviamente si incarica anche di verificare che il *client* abbia opportuni permessi.
- ***Printer driver***. Uno o più *files* che contengono le informazioni che permettono a *Windows* di convertire i comandi di stampa in comandi comprensibili alla specifica stampante (*rendering*).

Introduzione a Windows 2000 printing - Prerequisiti

Di seguito i prerequisiti *hardware* richiesti:

- Una macchina che svolga il ruolo di *Print Server* eventualmente dedicato. Tale macchina può essere:
  - *Windows* 2000 *Server*, *Windows* 2000 *Advanced Server*, o *Windows* 2000 *Datacenter Server*. In tal caso è possibile supportare oltre che un elevato numero di connessioni comtemporanee, anche *clients Macintosh*, *UNIX*, e *NetWare*.
  - *Windows* 2000 *Professional*. In tal caso il numero di connessioni contemporanee è 10, compresi i *client UNIX*.

- Memoria RAM a sufficienza per processare le richieste di stampa.
- Spazio disco a sufficienza per memorizzare i documenti. Bisogna garantire che il *print server* abbia spazio disco a sufficienza, per poter memorizzare i documenti che devono essere stampati.

Aggiungere un printer

- **Aggiungere e condividere un *printer* per un *Local Print Device*** .
- **Configurare i *clients*** .

Quando si installa e condivide una stampante è possibile scegliere tra un *printer* che faccia riferimento ad un *Local Print Device* ed un *printer* che faccia riferimento ad un *Network Print Device*. Questa seconda soluzione è quella ideale in ambienti che prevedono un elevato numero di *client*.

Quando si installa un *printer*, bisogna inoltre verificare che tutti i *client* siano propriamente configurati, in particolare che essi abbiano a disposizione i *driver* relativi alla loro versione del sistema operativo.

Ricordiamo che i *driver* sulla macchina *client* sono necessari affinché quest'ultima possa permettere alle applicazioni di sapere come sarà la stampa del documento (*WYSiWYG, What You See Is What You Get*) e sia possibile configurare le varie proprietà della stampante. Invece il *rendering* del documento avviene invece sul *print server*.

Aggiungere e condividere un printer per un local print device

Per aggiungere e condividere una stampante bisogna utilizzare un *account* utente che appartenga al gruppo *Administrator* su quello che sarà il *print server*.

Per installare e condividere un *Printer* relativo ad un *Local Print Device* selezionare le seguenti opzioni durante l'esecuzione del *wizard Add Printer* che troviamo in `Start\Setting\Printers`:

- ***Local printer***. Selezionando questa opzione stiamo dichiarando di voler configurare il *Print Server*.
- ***Use the following port***. La porta sul *Print Server* a cui è connessa la *print Device*. È possibile selezionare una delle porte esistenti o aggiungerne una nuova, come ad esempio una porta *hardware* non-standard.
- ***Manufacturers e Printers***. Selezionare il *Printer Driver* corretto per la stampante. Se la nostra stampante non è individuabile utilizzando le diverse possibilità in *Manufacturers* e *Printers*, bisogna preoccuparci di fornire tale *driver* o quantomeno uno compatibile.
- ***Printer name***. Un nome che identifica la stampante per gli utenti. Tale nome può essere lungo fino a 31 caratteri ed ovviamente si consiglia che sia il più possibile intuitivo ed esplicativo.
- ***Default printer***. Se questa stampante è la stampante di *default* per le varie applicazioni. Durante l'installazione della prima stampante, tale opzione non viene proposta.
- ***Shared as***. Il nome di condivisione che gli utenti, se opportunamente autorizzati, possono utilizzare per connettersi alla stampante tramite la rete. Tale nome viene visualizzato quando l'utente sfoglia le risorse del *server*. Assicuriamoci che il nome scelto sia correttamente visualizzabile da tutti i *client* (alcuni *client* potrebbero supportare solo nomi in formato 8.3, e dunque per tali *clients* il nostro nome verrà troncato per rispettare tale formato).
- ***Location and Comment***. Informazioni riguardo la stampante.
- ***Do you want to print a test page?*** Permette di verificare che l'installazione sia terminata cons successo. In tal caso selezionare *Yes* nella finestra di dialogo che viene proposta. In caso contrario selezionare No.

Configurare i clients

I seguenti *clients* scaricano automaticamente i *drivers*:

- *Windows* 95.
- *Windows* 98.
- *Windows* NT.

I *client* che eseguono altri sistemi operativi *Microsoft* richiedono l'installazione dei *drivers*.

I *client* non-*Microsoft* richiedono l'installazione di:

- *driver* sul *client*.
- Appropriato servizio sul *server*.

Dopo aver installato e condiviso la stampante sul *Print Server*, è possibile configurare i *computer clients* affinché possano utilizzare tale stampante.

La procedura di installazione del *client* varia in base al sistema operativo installato, ma in ogni caso è richiesta la presenza del *printer driver* sulla macchina *client*.

Distinguiamo i seguenti casi, oltre ai *client Microsoft Windows* 2000:

- ***Client Windows* 95, *Windows* 98, o *Windows* NT 4.0**. Gli utenti di tali *client* devono semplicemente connettersi alla stampante condivisa. Essi scaricheranno automaticamente l'appropriato *Printer Driver*, che sarà stato reso disponibile sul *Print Server*. Inoltre, solo per i *client Windows* NT 4.0, se aggiorniamo la copia del *Printer Driver* memorizzato sul *Print Server*, al prossimo utilizzo della stampante sul *client* essi scaricheranno la copia aggiornata del *Printer Driver*. I *client Windows* 95, *Windows* 98 dovranno provvedere invece manualmente.
- ***Client Microsoft* Non *Windows* 95, *Windows* 98, o *Windows* NT 4.0**. In questo caso, affinche sia possibile utilizzare la stampante condivisa, tali *client* devono connettersi ad essa e bisogna installare manualmente i *Printer Driver* opportuni.
- ***Client* Non *Microsoft***. In questo caso, affinche sia possibile utilizzare la stampante condivisa, tali *client* devono connettersi ad essa, bisogna installare manualmente i *Printer Driver* opportuni e bisogna installare dei servizi aggiuntivi sul *Print Server*.
  - ***Client Machintosh***. Bisogna installare il servizio *Macintosh Services for Macintosh*, compreso in *Windows* 2000 ma non installato di *default*.
  - ***Client UNIX***. Bisogna installare il servizio *UNIX TCP/IP Printing*, compreso in *Windows* 2000 ma non installato di *default*, che contiene il *Line Printer Daemon* (LPD).
  - ***Client Novell Netware***. Bisogna installare il servizio *File and Print Services for NetWare*, non compreso in *Windows* 2000.

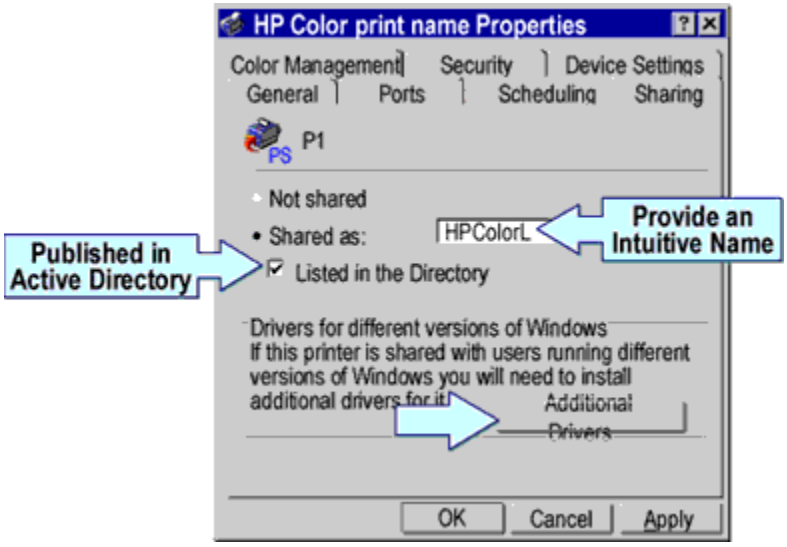
Configurare un network printer

- **Condividere un *printer* esistente** .
- **Definire un *printer pool*** .
- **Impostare le priorità di un *printer*** .
- **Impostare i permessi di stampa** .

Concludiamo ora le nostre osservazioni inerenti la stampa in *Microsoft Windows* 2000, analizzando come condividere un *printer* esistente, definendo quali sono i permessi di stampa ed analizzando, infine, due configurazioni avanzate

che riguardano la realizzazione di un *Pool* di stampanti per realizzare una configurazione performante e *fault tolerant*, e la configurazione della priorità associata ad un *printer* per poter privilegiare la stampa di documenti critici.

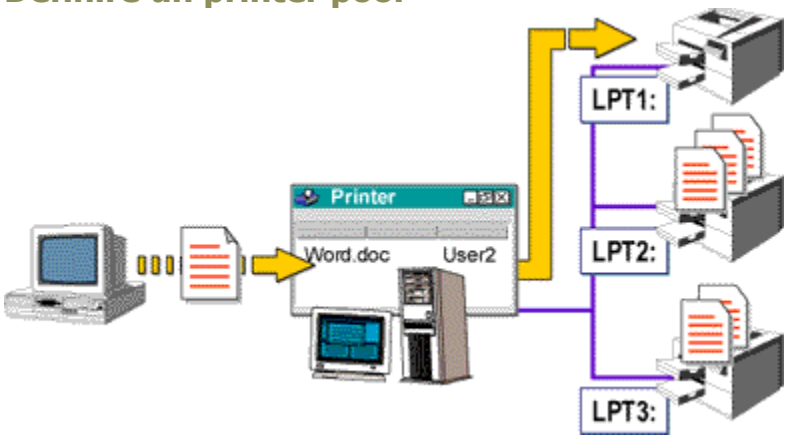
Condividere un printer esistente



Per condividere una stampante già installata su un *server*:

- Accedere alla cartella *Start|Setting|Printers*.
- Selezionare la stampante desiderata.
- Cliccare con il tasto destro e scegliere *Sharing*.
- Specificare le seguenti opzioni:
  - Selezionare *shared AS* e specificare il nome di condivisione.
  - Decidere se pubblicare la stampante in *Active Directory* selezionando *Listed in The Directory* (di *default* è selezionato).
  - Aggiungere *Printer Driver* aggiuntivi per *client Windows* 95, *Windows* 98, o *Windows* NT 4.0 selezionando *Additional Drivers* .

Definire un printer pool



Un *Printer Pool* consiste di un solo *Printer* che è connesso a più di una *Print Device* tramite più porte del *Print Server*. Le *Print Device* possono essere sia *Local* che *Network Interface*, e non necessariamente devono essere identiche, ma

tutte compatibili con lo stesso *Printer Driver*.

L'utente invierà le sue stampe all'unico *Printer* che provvederà e distribuirle sulle varie *Print Device* in maniera del tutto trasparente al *client*.

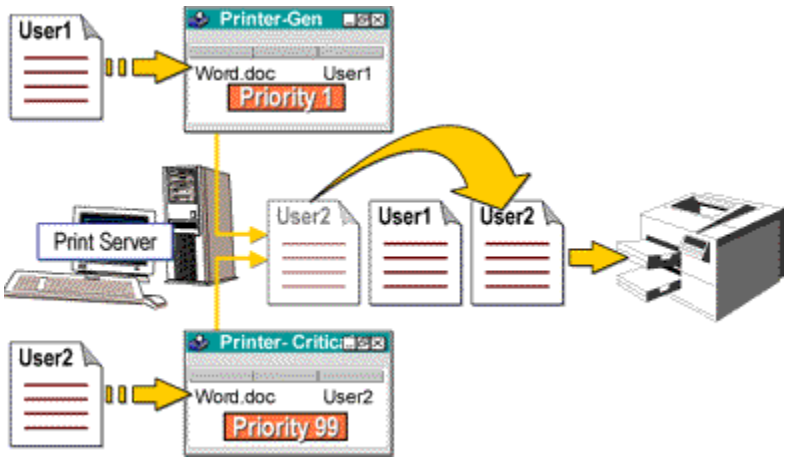
L'utilizzo di un *Printer Pool* presenta i seguenti vantaggi:

- Diminuisce il tempo di permanenza dei documenti sul *Print Server*.
- Semplifica l'amministrazione poiché consente di amministrare più *Print Device* tramite un unico *printer*.
- Crea una configurazione *Fault Tolerant*.

Per creare un *Printer Pool*:

- Accedere alla cartella *Start|Setting|Printers*.
- Selezionare la stampante desiderata.
- Cliccare con il tasto destro e scegliere *Properties*.
- Selezionare la scheda *Ports*
- Selezionare l'opzione *Enable printer pooling*
- Selezionare le porte a cui le *Print Device* sono state connesse.

Impostare le priorità di un printer



Installare su un *Print Server* più *printer* corrispondenti alla stessa *Print Device*, ed assegnare ad ogni *Printer* un diverso livello di priorità permette di stabilire dei livelli di priorità tra dicumenti stampati effettivamente dalla stessa *Print Device*.

Per impostare *Printers* con priorità diverse:

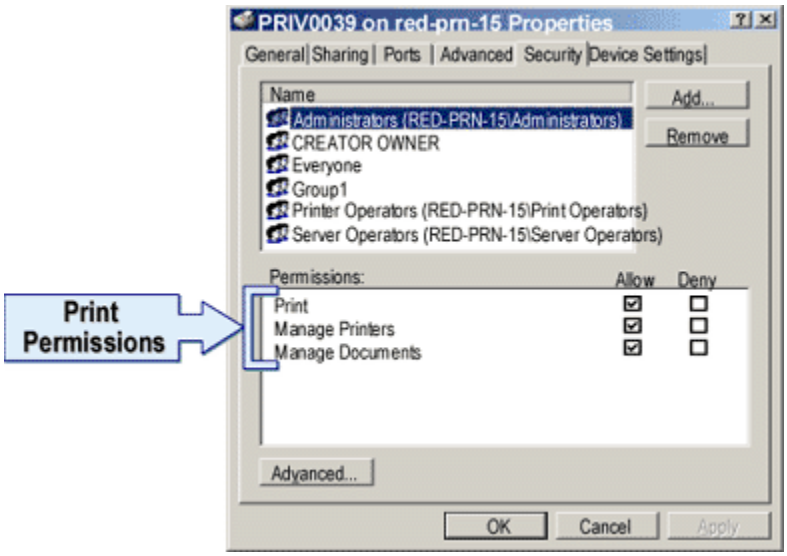
- Installare più volte lo stesso *Printer* e farlo puntare sempre alla stessa porta fisica.
- Impostare diversi livelli di priorità per ognuno dei *Printer*.
- Condividere i vari *printer* ed assegnare i permessi di condivisione in maniera tale che ogni utente acceda al *printer* con il livello di priorità più opportuno.

Per impostare invece la priorità di un *Printer*:

- Accedere alla cartella *Start|Setting|Printers*.
- Selezionare la stampante desiderata.

- Cliccare con il tasto destro e scegliere *Properties*.
- Selezionare la scheda *Advanced* ed impostare la priorità oppoortuna (da 1 a 99) utilizzando il cursore in *Priority*.

Impostare i permessi di stampa



Esistono tre tipi di permessi per quel che concerne la condivisione di una stampante:

- ***Print***. Stampare documenti, gestire i propri documenti nella coda di stampa, connettersi alla stampante.
- ***Manage Documents***. Gestire tutti i documenti nella coda di stampa oltre ai permessi associati a *Print*.
- ***Manage Printer***. Condividere la stampante, interrompere la condivisione, eliminare la stampante, modificare i permessi oltre ai permessi associati a *Manage Documents*.

Di *default* i gruppi *Administrators* e *Print Operators* hanno il permesso *Manage Printer*, mentre il gruppo *Everyone* ha *Print* ed il proprietario *Manage Documents*.

Per modificare i permessi di condivisione o aggiungerne altri:

- Accedere alla cartella *Start*|*Setting*|*Printers*.
- Selezionare la stampante desiderata.
- Cliccare con il tasto destro e scegliere *Properties*.
- Selezionare la scheda *Security*.
- Utilizzare *Add* per aggiungere ulteriori utenti e gruppi ed utilizzare in *Permission* gli opportuni *check box* per assegnare (*Allow*) o negare (*Deny*) un permesso.
- Utilizzare il pulsante Remove per rimuovere gruppi om utenti con i relativi permessi.

DHCP: Nuove Funzionalità

DCHP (*Dynamic Host Configuration Protocol*) è il protocollo ed il relativo servizio che permette la gestione centralizzata ed automatica dei parametri di configurazione relativi al protocollo TCP/IP e necessari ai vari *hosts* presenti sulla rete.

In *Windows* 2000 sono state aggiunte al DHCP tutta una serie di nuove funzionalità, tra cui ricordiamo:

- **Rilevamento di *Server* DHCP non autorizzati** . È possibile impedire che sulla rete siano attivi *server* DHCP non autorizzati che potrebbero portare all'assegnazione di indirizzi IP duplicati.
- **Integrazione con il DNS** (Domain Name System). Quando il *server* DHCP assegna un indirizzo IP ad un *client* provvede anche alla registrazione parziale o totale delle informazioni necessarie sul *server* DNS se quest'ultimo è abilitato agli aggiornamenti dinamici.
- **Supporto esteso agli *scope*** . Oltre che *scope* tradizionali è possibile definire ***Superscopes*** e ***Scope Multicast*** .
- **Supporto delle *Option Classes*** . Tramite le *Option Class* è possibile gestire in maniera estremamente flessibile l'assegnazione dei parametri opzionali ai vari *client* basandosi su caratteristiche quali il tipo di sistema operativo o definendone di personalizzate.
- **Assegnazione Automatica degli Indirizzi IP (APIPA)** . Ai *client* DHCP che non riescono momentaneamente a raggiungere il *server* DHCP viene assegnato un indirizzo IP momentaneo appartenente ad un insieme riservato, non utilizzato in Internet.
- *Funzionalità avanzate di monitoraggio e statistica*.
- *Possibilità di ri-registrare i record sul server DNS*. Quando il *client* DHCP rinnova il proprio indirizzo allo scadere dell'intervallo di '*lease*', tale indirizzo viene anche nuovamente ri-registrato sul *server* DNS. Ciò garantisce un certo livello di *fault tolerance* rispetto ai *client* configurati staticamente che effettuano la registrazione automatica sul DNS solo al momento della partenza.

Autorizzare un DHCP Server in Active Directory

Con le precedenti implementazioni del DHCP chiunque poteva installare un *server* DHCP ed inserirlo su una rete, provocando l'assegnazione di indirizzi IP non controllati ed eventuali fenomeni di indirizzi duplicati.

In *Windows* 2000 è necessario autorizzare un *server* DHCP in *Active Directory* prima che tale *server* possa funzionare correttamente.

Questo poiché quando il servizio DHCP parte, contatta *Active Directory* per verificare se appartiene alla lista dei *server* autorizzati. In seguito a tale controllo possono verificarsi le seguenti condizioni:

- Il *server* DHCP è autorizzato e dunque il servizio parte correttamente.
- Il *server* DHCP non è autorizzato e dunque il servizio registra un errore nel log di sistema e non risponde a nessuna delle richieste dei *client*

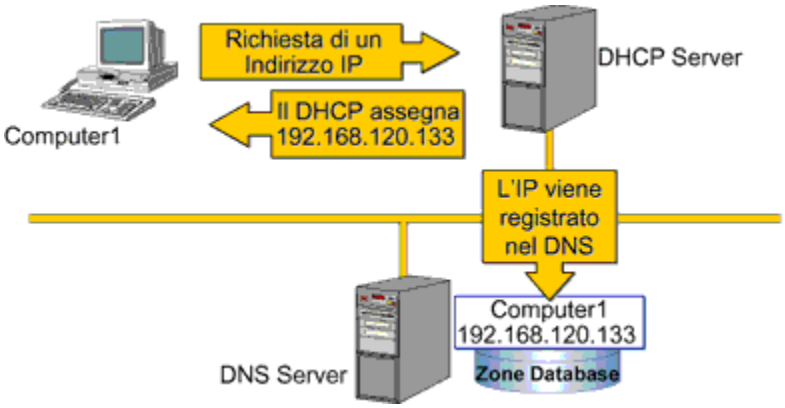
Periodicamente il DHCP continua a controllare se sono avvenute modifiche all'elenco dei *server* autorizzati.

Per autorizzare un *server* DHCP in *Active Directory*, utilizzare il *tool* DHCP presente in *Start|Programs|Administrative Tools* e facendo click con il tasto destro su DHCP scegliere *Manage authorized servers*, selezionare *Authorize* ed inserire il nome o l'indirizzo IP del *server* da autorizzare.

Per autorizzare un *server* DHCP bisogna appartenere al gruppo *Enterprise Admins* presente nel dominio radice della foresta.

Aggiornamento Dinamico del Server DNS

Di *default*, un *server* DHCP *Windows* 2000 è abilitato ad eseguire l'aggiornamento di un *server* **DNS** per il quale sia attivo l'aggiornamento dinamico. Di conseguenza, il *server* DHCP aggiorna automaticamente il record PTR per i *client* *Windows* 2000 che provvedono invece all'aggiornamento del record di tipo A.



Per configurare il *server* DHCP affinché aggiorni automaticamente il *server* DNS, utilizzare il *tool* DHCP presente in *Start|Programs|Administrative Tools* e facendo click con il tasto destro sul *server* in questione scegliere *Properties* e selezionare la scheda DNS.

Sono disponibili le seguenti opzioni:

- ***Automatically update DHCP client information in DNS***. Abilita la funzionalità di aggiornamento dinamico del DNS per il *server* DHCP. Se tale opzione non è selezionata, nessun'altra opzione risulta essere disponibile.
- ***Update DNS only if DHCP client requests***. Aggiorna il record A ed il record PTR in base a quelle che sono le richieste del *client* durante il processo di assegnazione dell'indirizzo IP. Questa opzione è selezionata di *default* e determina che il *client Windows* 2000 provvede alla registrazione del record A ed il *server* DHCP alla registrazione del record PTR. Per *client* diversi da *Windows* 2000 non avviene nessun aggiornamento di tipo dinamico.
- ***Always update DNS***. Indipendentemente dalla richiesta del *client* il DHCP si fa carico del completo processo di registrazione sul *server* DNS.
- ***Discard forward (name-to-address) lookups when lease expires***. Rimuove i record A e PTR relativi ad un *client* per il quale un certo indirizzo IP non è più valido.
- ***Enable updates for DNS clients that do not support dynamic update***. Abilita l'aggiornamento dinamico sia del record A che del record PTR da parte del *server* DHCP per i *client* DHCP non *Windows* 2000.

Se i *client* sono *Windows* 2000 ed il *server* e *Microsoft Windows* NT 4.0 DHCP saranno i *client Windows* 2000 a dover aggiornare sia il record A che il record PTR.

Configurare i DHCP Scopes in Windows 2000

Windows 2000 estende le funzionalità di un *server* DHCP supportando oltre allo *scope* tradizionale anche i cosiddetti ***Superscopes*** e ***Scope Multicast*** .

Tramite queste due nuove funzionalità è possibile assegnare, ad esempio, indirizzi IP a reti fisiche che contengono più di una sottorete logica.

Oltre a queste nuove tipologie di *scope* è comunque presente un *wizard* che semplifica estremamente il processo di creazione di un qualsiasi tipo di *scope*.

Configurare uno Scope

Windows 2000 utilizza il *wizard* '*Create Scope*' per rendere più semplice il processo di creazione di uno *scope*.

Per avviare tale *wizard* utilizzare il *tool* DHCP presente in *Start|Programs|Administrative Tools* e fare doppio click sul *server* che ospiterà lo *scope*. Fare click con il tasto destro sul *server* e selezionare *New Scope*.

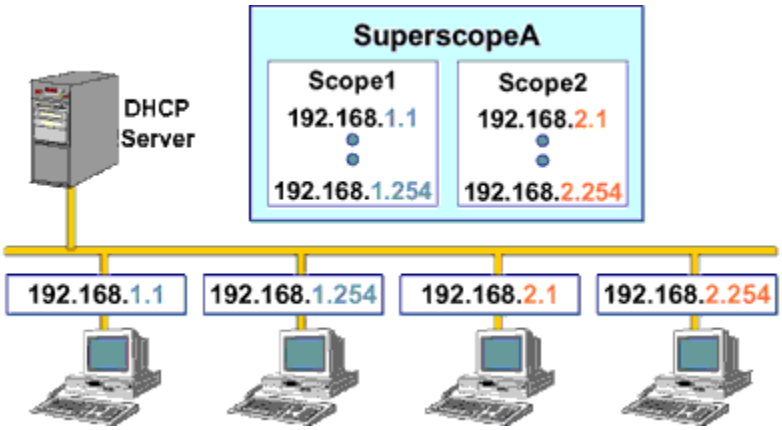
Verranno richieste le seguenti informazioni:

- Nome dello *scope*.
- *Range* di indirizzi IP da assegnare.
- *Subnet mask* da associare agli indirizzi definiti nel *range*.
- Eventuale intervallo di esclusione del *range*.
- Durata dell'assegnazione.
- Parametri DHCP comuni, tra cui:
  - Indirizzo dei *gateways* di *default*.
  - Nome del dominio DNS ed indirizzo dei *server* DNS.
  - Indirizzo IP di ogni *server WINS*.

Di *default* uno *scope* non è attivo. Per attivarlo, cliccare con il tasto destro sullo *scope*, selezionare *All Tasks* e quindi *Activate*.

Configurare un Superscope

In una rete basata su *Windows* NT 4.0, gli indirizzi IP assegnati ai *client* DHCP dovevano essere definiti in modo che ad ogni sottorete logica corrispondesse una singola sottorete fisica e viceversa.



L'implementazione del DHCP in *Windows* 2000 prevede, tramite i *Superscopes*, la possibilità che ad un'unica sottorete fisica siano associate più sottoreti logiche *Windows* NT 4.0 *Service Pack* 2 supporta i *Superscopes*.

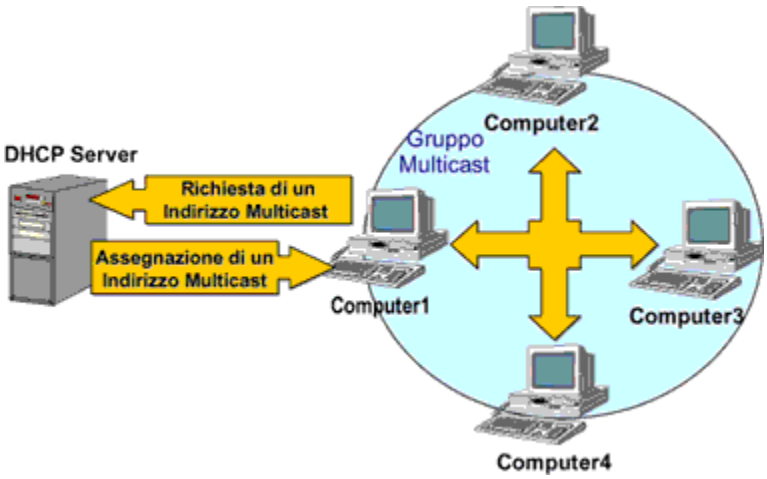
Ad esempio, è possibile utilizzare i *Superscopes* nelle seguenti situazioni:

- Bisogna aggiungere ad una sottorete più *host* di quanti pianificato in fase di progettazione e configurazione del *server* DHCP.
- Bisogna sostituire un *range* di indirizzi esistente con un nuovo *range*.

Per configurare i *Superscopes* utilizzare il *tool* DHCP presente in *Start|Programs|Administrative Tools* e fare doppio click sul *server* che ospiterà il *Superscopes*. Fare click con il tasto destro sul *server* e selezionare *New Superscope*. Verranno richiesti il nome e quali sono gli *scope* esistenti da includere.

Configurare un Multicast Scope

Ricordiamo che gli indirizzi IP il cui primo ottetto varia in un intervallo che va da 224 a 239 (Classe D) sono i cosiddetti Indirizzi *Multicast*, cioé quella classe di indirizzi IP che vengono utilizzati per indirizzare, non un singolo *host*, ma un insieme di *host* ben definito. Tali *host* hanno la caratteristica di appartenere allo stesso Gruppo *Multicast* identificato appunto da uno di tali indirizzi. Tramite tale indirizzo è possibile raggiungere con un solo messaggio diretto un numero di macchine maggiore di uno.



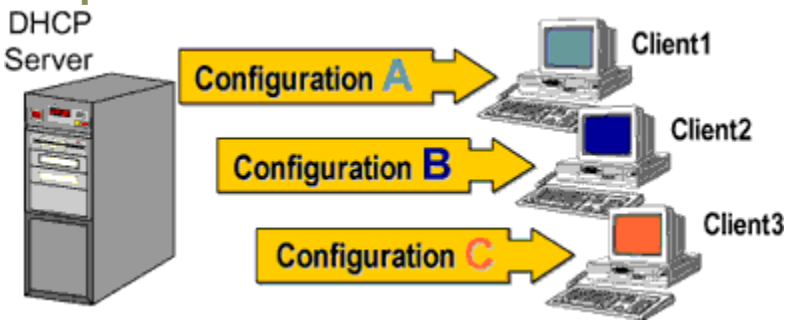
Tipicamente le applicazioni basate sull'utilizzo di flussi audio e videoconferenza si basano su tecnologie *multicast*.

Tramite la definitione di *Multicast scopes* è possibile automatizzare e gestire in maniera centralizzata l'assegnazione di tale tipologia di indirizzi ai *client* che ne necessitano.

Per configurare i *Multicast scopes* utilizzare il *tool* DHCP presente in *Start|Programs|Administrative Tools* e fare doppio click sul *server* che ospiterà il *Superscopes*. Fare click con il tasto destro sul *server* e selezionare *New Multicast Scope*.

Verranno richiesti il nome e il *range* di indirizzi *multicast*. Attivare lo *scope* quando verrà richiesto.

Le Option Classes



In una rete basata su *Microsoft Windows* NT 4.0, il *server* DHCP fornisce gli stessi parametri di configurazione opzionali per tutti i *client* di uno stesso *server* o, al massimo, appartenenti allo stesso *scope*. Inoltre, utilizzando le *reservation* è possibile specificare parametri di configurazione particolari per uno specifico *client* utilizzando il suo *MAC address*.

Al di là di questi ambiti (*server*, *scope*, singola macchina) non è possibile definire un qualsivoglia sottoinsieme di macchine, basandosi su una qualche caratteristiche ad esse comune, e specificare delle opzioni riservate a tali

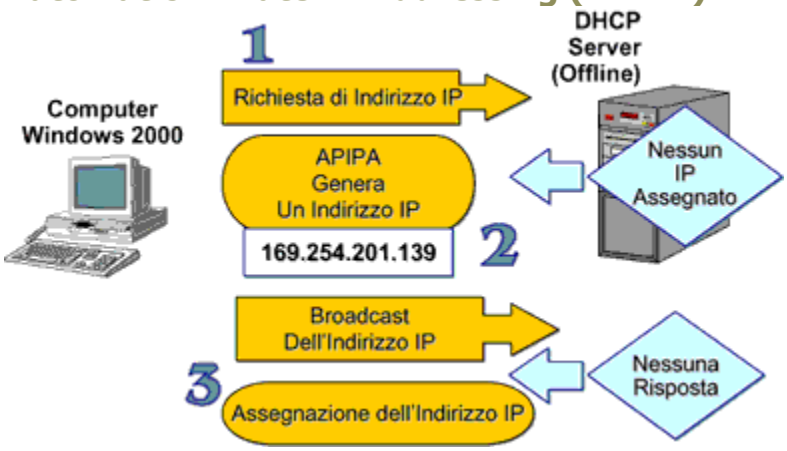
macchine.

L'implementazione del DHCP in *Windows* 2000, supporta le *Option Classes*, che permettono, appunto, di definire insiemi di macchine in base ad una qualsiasi caratterizzazione, e riservare a tale insieme parametri di configurazione specifici. Esempi di *Option Classes* possono essere tutte le macchine che abbiano una certa versione del sistema operativo, o le macchine di un certo tipo (*desktop*, *laptop*) o tutte le macchine abilitate all'accesso ad Internet, eccetera ...

Windows 2000 supporta due tipi di *Option Classes*:

- ***Vendor-defined classes***. Permettono di identificare un *client* dalla versione del suo sistema operativo
- ***User-defined classes***. Permettono di identificare i *client* in base ad una qualche caratteristica personalizzata, ad esempio *desktop* o *laptop*, necessità di accedere ad Internet, locazione geografica e quant'altro possa risultare conveniente.

Automatic Private IP Addressing (APIPA)



Windows 2000 supporta un nuovo meccanismo per l'assegnazione automatica di indirizzi IP nel caso di reti LAN di piccole dimensioni.

Questo meccanismo, denominato '*Automatic Private IP Addressing* (APIPA)', permette di assegnare un indirizzo IP ad un *host* senza ricorrere alla configurazione statica (manuale) e nel contempo senza ricorrere ad un *server* DHCP.

Il suo funzionamento è il seguente:

- Quando *Windows* 2000 parte, il TCP/IP prova a cercare un *server* DHCP che possa fornire un indirizzo IP valido.
- In assenza del *server* DHCP, il *client* non resterebbe sprovvisto di indirizzo IP. Se invece APIPA risulta abilitato, all'*host* viene assegnato un indirizzo IP del tipo 169.254.x.y con *subnet mask* 255.255.0.0.
- Dopo la generazione di tale indirizzo il *client* verifica tramite un *broadcast* che tale indirizzo non sia in uso ed in questo caso lo usa come suo, altrimenti ne genera un altro ed il processo si ripete.
- Il *client* continua ad utilizzare tale indirizzo finché non scopre che un DHCP ha un indirizzo disponibile e valido (la ricerca di tale DHCP viene effettuata ogni 5 minuti).

Per implementare tale funzionalità *Microsoft* si è riservata per l'utilizzo tramite APIPA il *range* di indirizzi da 169.254.0.1 a 169.254.255.254.

Purtroppo APIPA è in grado di generare l'indirizzo IP e la *subnet mask* ma non ogni altro eventuale parametro (indirizzo del *gateway*, indirizzo dei DNS, indirizzo dei *WINS* ...) di cui il *client* dovesse aver bisogno. Dunque, gli *host*

che hanno ricevuto un indirizzo tramite APIPA possono comunicare solo con *host* dello stesso segmento che abbiano ricevuto l'indirizzo IP nello stesso modo.

APIPA è abilitata di *default*. Per disabilitare APIPA, basta aggiungere il valore `IPAutoconfigurationEnabled` di tipo `REG_DWORD` con valore 0 al *path* di registro:
`HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\GUID_d`
`_scheda_di_rete`

Configurare un Web Server

Il processo di creazione di un *Web Server* è un processo molto simile a quello relativo alla creazione di un *file server*.

Come nel caso di ogni altra tipologia di *server* bisogna innanzitutto installare il sistema operativo, decicere l'appartenenza ad un gruppo di lavoro (*stand-alone server*) o ad un dominio (*member server*), ed organizzare in maniera opportuna i dischi.

Il passo successivo consiste nell'installare e configurare in maniera appropriata i servizi relativi alla funzionalità di *Web Server*.

Ricordiamo che tramite un *Web Server* un *client* accede a *file* che costituiscono le pagine *Web*, ma anche a complesse applicazioni *client/server* basate su *database*.

Windows 2000 *server* comprendono un *Web Server* che va sotto la denominazione di *Internet Information Services* (IIS) 5.0.

IIS viene installato di *default* come servizio di rete durante l'installazione di *Windows* 2000 e permette di supportare anche configurazioni di *Web Server* abbastanza complesse (*Server Web* Virtuali, Cartelle Virtuali, ...).

Per poter installare e configurare con successo IIS 5.0 su un *member server*, si necessita di:

- *Transmission Control Protocol/Internet Protocol* (TCP/IP).
- Almeno un indirizzo IP.
- *Domain name*. Se si vuole accedere il *server* utilizzando un *Fully Qualified Domain name* (FQDN) piuttosto che il suo indirizzo IP, c'è bisogno di installare e configurare opportunamente un *server* DNS.

Bisogna poi ovviamente provvedere alla creazione e configurazione del *Server Web* tramite la definizione della sua *Directory* Radice e la definizione di tutte le strategie atte a permettere un accesso controllato e sicuro (autenticazione, cifratura ...) al *server* in questione.

Linux - DHCP server

Il protocollo DHCP permette di assegnare in modo dinamico gli indirizzi IP alle macchine della rete locale. Vi sono diversi *server* DHCP disponibili per i sistemi *UNIX*, sia *software* commerciali che *Open Source*. Su *Red Hat Linux* viene utilizzato DHCPd di *Paul Vixie*.

La configurazione della parte *client* del protocollo su *Linux* è stata descritta nell'unità didattica relativa alla configurazione di rete.

Per il funzionamento del protocollo DHCP è necessario che nel *kernel* sia configurato il supporto per il *multicast*. È possibile vedere se una interfaccia è correttamente configurata utilizzando il comando `ifconfig`:

```
# ifconfig eth0
eth0 Link encap:10Mbps Ethernet HWaddr 00:C0:4F:D3:C4:62
inet addr:183.217.19.43 Bcast:183.217.19.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:2875542 errors:0 dropped:0 overruns:0
TX packets:218647 errors:0 dropped:0 overruns:0
Interrupt:11 Base address:0x210
```

Se l'etichetta *MULTICAST* non è presente, si deve ricompilare il *kernel* aggiungendo il supporto per il *multicast*. Alcuni *client* DHCP richiedono che nel *server* esista una *route* per la rete 255.255.255.255. Questa può essere difficoltosa da creare, in quanto alcune versioni del comando *route* per *Linux* insistono nel cambiare l'indirizzo 255.255.255.255 nell'indirizzo di *broadcast* della rete locale. Per verificare se la versione in uso è affetta da questo problema è sufficiente scrivere il comando

```
# route add -host 255.255.255.255 dev eth0
```

se viene restituito un errore 255.255.255.255: *Unknown host*, è possibile provare ad inserire in `/etc/hosts` la seguente linea

```
255.255.255.255 all-ones
```

ed aggiungere la *route* utilizzando la linea

```
/sbin/route add -host all-ones dev eth0
```

Per rendere permanente la modifica bisognerà ovviamente inserire tale comando in uno *script* di avvio.

Il *server* DHCPd viene generalmente eseguito in modalità *standalone*, lanciandolo da uno degli *script* di avvio in `/etc/rc.d`, dopo averlo configurato mediante il *file* `/etc/dhcpd.conf`. Per associare il servizio ad una interfaccia diversa da quella di *default* (`eth0`), è sufficiente scriverne il nome come parametro nella linea di comando (ad esempio: `/usr/local/bin/dhcpd eth1`).

In ambiente KDE è possibile utilizzare per la configurazione l'interfaccia grafica `kcmdhcpd`.

Il seguente esempio mostra come assegnare ai *client* degli indirizzi IP scelti negli intervalli 192.168.1.10-192.168.1.100 oppure 192.168.1.150-192.168.1.200, con periodo di validità (*lease*) di 600 secondi come *default* e di 7200 secondi come periodo massimo. Vengono inoltre passate ai *client* le informazioni relative alla *netmask*, all'indirizzo di *broadcast*, al *gateway* e ai *server* DNS e *WINS* (*netbios-name-servers*).

```
# Sample /etc/dhcpd.conf
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option routers 192.168.1.254;
option domain-name-servers 192.168.1.1, 192.168.1.2;
option netbios-name-servers 192.168.1.1;
option domain-name mydomain.org;

subnet 192.168.1.0 netmask 255.255.255.0 {
range 192.168.1.10 192.168.1.100;
range 192.168.1.150 192.168.1.200;
}
```

È possibile assegnare ad un *client* con dato *MAC address* un indirizzo IP fisso mediante una linea del tipo:

```
host pc222 {
hardware ethernet 08:00:2b:4c:59:23;
fixed-address 192.168.1.222;
}
```

La lista degli indirizzi assegnati ai vari *client* viene mantenuta nel *file* `/var/state/dhcp/dhcpd.leases`. Essa è in formato testo e pertanto mediante semplici *script* può essere utilizzata per tenere sincronizzata la tabella dei nomi locali nel DNS.

Per ulteriori informazioni si consulti il documento DHCP-Mini-*HOWTO*.

Linux - Servizi di stampa

Nei sistemi *UNIX* lo *spooler* lpd può essere configurato in modo da accettare, alla porta TCP/512, richieste di stampa provenienti da *client* remoti. Il controllo degli accessi viene gestito mediante il *file* `/etc/hosts.lpd`. Per configurare una macchina *Linux* in modo da spedire una stampa ad un lpd remoto è necessario definire una stampante di rete in `/etc/printcap`, specificando in modo opportuno il campo `rm` (*remote machine*). Un esempio è il seguente:

```
net|laserjetlp:\
:sd=/var/spool/lpd/net:\
:mx#0:\:sh:\
:rm=printsrv.mydomain.com:\
:rp=lp:\
:if=/var/spool/lpd/net/filter:
```

Per ulteriori informazioni si faccia riferimento alla documentazione di lpd oppure ai seguenti *HOWTO*:

- *Printing-HOWTO*.
- *Printing-Usage-HOWTO*.

Eventuali stampe da e verso sistemi *Windows* possono essere gestite mediante il *software* Samba, descritto in una unità didattica a parte.

Linux - NIS (Network Information System)

È un sistema, inizialmente sviluppato da *Sun* col nome di *Yellow Pages* (`yp`), che permette la gestione centralizzata delle informazioni amministrative in ambiente *UNIX* (*password*, gruppi, *hosts*...), che in questo modo possono essere condivise da tutti i *client* appartenenti ad uno stesso dominio NIS. Pertanto un utente può collegarsi su macchine diverse utilizzando il medesimo *account* senza dover tenere sincronizzate fra i diversi *computer* le tabelle delle *password*. Le tabelle NIS prendono il nome di mappe.

Le tabelle vengono rese disponibili da un *server master* e possono essere replicate su uno o più *server* con funzioni di *backup* (*slave*). La modifica dei dati contenuti nelle zone è possibile da macchine remote opportunamente abilitate.

Linux - Configurazione dei client NIS

Se si vuole abilitare un *client* all'accesso ad un *server* NIS, è necessario attivare il demone `ypbind`, mediante uno *script* di avvio. Il dominio NIS viene impostato mediante il comando *`domainname`*, mentre la modalità di accesso al *server* viene definita nel *file* `/etc/yp.conf` nel seguente modo:

```
domain <dominio-NIS> server <host>
domain <dominio-NIS> broadcast
ypserv <host>
```

Nei *client* è necessario inoltre definire mediante il meccanismo di NSS (*Name Server Switch*), l'ordine con cui deve essere consultato il *server* NIS rispetto ai *file* di sistema. È buona norma consultare per primi i *file* di sistema, in modo da evitare che da un malfunzionamento di questo derivi l'impossibilità di collegarsi alla macchina. Nel caso della tabella delle *password* questo comportamento può essere ottenuto inserendo la seguente linea in `/etc/nsswitch.conf`:

```
passwd: files nis
```

Occorre inoltre aggiungere in coda a `/etc/passwd` la seguente linea:

```
+::::::
```

Di norma gli *account* amministrativi come *root*, *bin*, *wheel*, *demon* ed altri non vengono condivisi tramite NIS, per motivi di sicurezza e di *performance*, ma vengono invece utilizzati quelli definiti nel *file* `/etc/passwd` locale.

Per ulteriori informazioni su NSS e NIS si può fare riferimento anche alla unità didattica relativa all'autenticazione nei sistemi *UNIX*.

Configurazione del *server* NIS

Il *server* NIS per mantenere le informazioni non utilizza i classici *file* di testo, come ad esempio `/etc/passwd`, bensì dei *file* speciali detti mappe, contenenti coppie di valori nella forma chiave-valore, ordinati per chiave in modo da poter ricercare velocemente i dati.

Ad esempio partendo da `/etc/passwd` vengono generate le due mappe passwd.byname e passwd.byuid, che permettono rispettivamente ricerche veloci per nome utente e per numero identificativo (UID).

Le mappe vengono di norma salvate in `/var/nis` oppure in `/var/yp`, a seconda della configurazione che si utilizza. Per generare le mappe a partire dai *file* di sistema ogni implementazione mette a disposizione un comando apposito, ad esempio `makedbm`. Generalmente viene fornito un *Makefile* che permette la rigenerazione della mappe semplicemente spostandosi nella *directory* dove si trovano le tabelle e digitando *make*.

Per definire quali informazioni un *server* NIS può fornire alle macchine appartenenti ad un dominio si utilizza il *file* `/etc/ypserv.conf`, mentre gli *host* di fiducia possono essere impostati mediante `/etc/yp/securenets`.

Il demone che fornisce il servizio di *server* NIS è `ypserver`, che viene attivato in modalità *standalone* mediante uno *script* di avvio. Essendo NIS basato su RPC è necessario che sia attivo nel *server* il servizio `portmapper`.

Aggiornamento dei dati nelle mappe NIS

Per l'aggiornamento dei dati nel sistema NIS non è sufficiente utilizzare i classici comandi *UNIX*, ad esempio `passwd`. Vediamo ad esempio cosa succede se l'utente cambia la propria *password* su una macchina del dominio NIS. I casi possibili sono due:

- se la macchina non è il *server*, il *file* locale verrà aggiornato ma non verrà mai utilizzato, in quanto la macchina è configurata per prelevare le *password* via NIS.
- Se la macchina è il *server* NIS, questo dispone potenzialmente della nuova *password* ma le mappe contengono ancora la *password* precedente. Pertanto la nuova *password* funziona solo sul *server*, fintantoché non vengono rigenerate le mappe.

La soluzione consiste nell'utilizzo di comandi specifici per NIS, in questo caso `nispasswd`. Nel *server* è necessario attivare il servizio `rpc.passwdd`, che permette agli utenti di cambiare *password* da remoto. È anche possibile fare in modo che `rpc.passwdd` tenga automaticamente aggiornate le mappe a seguito di una modifica su uno dei *client*. In questo caso il servizio deve essere attivo quindi sia sul *server* che sui *client*.

Configurazione di un *server slave*

Per la configurazione di un *server* di tipo *slave* è sufficiente attivare normalmente le funzioni di NIS *server* (escluso `nispasswd`) e configurare il sistema come un *client* del *server master*. Per la replica delle mappe si utilizza il comando

```
/usr/lib/yp/ypinit -s <master-NIS>
```

generalmente mediante uno degli *script* `/usr/lib/yp/ypxfr*`.

NIS+

NIS+ è una estensione di NIS che permette una miglior gestione di domini con un numero elevato di *client*. Con questa nuova implementazione è stata anche migliorata la sicurezza dell'intero sistema con una riprogettazione dei meccanismi di scambio dati. Allo stato attuale esistono implementazioni di NIS+ disponibili come *Open Source* solo per quanto riguarda la parte *client*. Se si vuole quindi realizzare una rete NIS+ è necessario preventivare l'utilizzo di un *server Sun*.

LDAP

LDAP (*Lightweight Directory Access Protocol*) è un altro sistema di gestione centralizzata di informazioni, il quale offre un approccio più generico rispetto a NIS, in quanto non si limita ai soli dati amministrativi ma è utilizzabile anche per gestire dati generici quali *bookmarks* o indirizzari.

L'aggettivo *lighweight* (leggero), suggerisce una progettazione del sistema che privilegia le prestazioni. L'architettura di LDAP è infatti studiata per avere tempi di risposta molto veloci nella ricerca e lettura dei dati, a scapito di una maggiore lentezza nelle operazioni di scrittura e aggiornamento. Questo non rappresenta un problema grave, in quanto le operazioni di scrittura avvengono di norma meno frequentemente rispetto a quelle di lettura (si pensi al numero di volte in cui la propria agenda telefonica viene consultata rispetto a quelle in cui viene modificata).

LDAP gestisce insiemi di dati strutturati secondo una gerarchia ad albero (*directory*), nei quali i singoli elementi possono essere oggetti che possiedono diversi attributi. Gli elementi di una *directory*, nodi intermedi o foglie dell'albero, possono anche essere riferimenti ad altre *directory* residenti su altri *server* LDAP. Ogni elemento deve avere un nome che permetta di individuarlo in modo univoco nella *directory*.

Alcuni dei tipi possibili utilizzabili per definire gli attributi di un oggetto sono i seguenti:

- *bin*: dato binario;
- ces (*case exact string*): stringa in cui vengono distinte le lettere maiuscole dalle minuscole;
- cis (*case ignore string*): stringa in cui le lettere maiuscole e minuscole sono trattate allo stesso modo;
- tel: numero telefonico;
- dn (*distinguished name*): nome univoco.

Un *server* che fornisce in *UNIX/Linux* il servizio LDAP è `slapd`. Esso mantiene i dati in un formato denominato LDBM, simile al DBM ma ottimizzato per le ricerche (per ognuno degli attributi degli oggetti contenuti in una *directory* viene creato un indice).

Linux - Network File System

Il protocollo NFS (*Network File System*), originariamente sviluppato da *Sun Microsystems*, permette di condividere (esportare) *filesystem* verso altre macchine *UNIX*. Applicazioni possibili sono la condivisione delle *home directory* degli utenti fra un *cluster* di macchine oppure la realizzazione di sistemi *diskless*, in cui il *filesystem* di *root* viene montato da un *server* di rete.

In *Linux* per utilizzare la parte *client* è sufficiente avere un *kernel* compilato con il supporto per NFS. Il *mounting* avviene mediante un comando del tipo:

```
# mount -t nfs server01:/disk2 /test
```

dove `server01` è il nome del *server* e `/disk2` la *directory* da montare. Il parametro `-t` è opzionale. Come con gli altri tipi di *filesystem*, è possibile automatizzare il *mount* utilizzando la tabella `/etc/fstab`.

Per quanto riguarda la parte *server*, NFS utilizza come trasporto il protocollo RPC (*Remote Procedures Call*). È pertanto necessario verificare di avere attivo il *portmapper* e i demoni `rpc.mountd` (gestore del montaggio) e `rpc.nfsd` (gestore delle richieste dei *client*).

È possibile definire i *filesystem* da esportare mediante la tabella `/etc/exports`, secondo la seguente sintassi:

directory indirizzo(opzioni)

Gli indirizzi delle macchine a cui permettere il *mount* dei *filesystem* possono essere specificati anche mediante espressioni regolari. Ad esempio, per esportare in lettura e scrittura la *directory* `/disk2` verso una singola macchina si può utilizzare la seguente linea:

```
/disk2 lnxserver001(rw, no_root_squash)
```

L'opzione `no_root_squash` specifica che l'utente *root* del *client* deve essere mappato nell'utente *root* del *server* (di *default* verrebbe mappato nell'utente *nobody*).

Per esportare il CDROM a tutte le macchine della rete locale con indirizzi 192.168.10.x si può utilizzare la seguente linea:

```
/mnt/cdrom 192.168.10.0/255.255.255.0(ro, insecure)
```

Dopo aver modificato `/etc/exports` è necessario riavviare il *server* NFS.

Per ulteriori dettagli si faccia riferimento al documento NFS-*HOWTO*.

Linux - Condivisione di filesystem e stampanti con altri sistemi operativi

Coda è un *filesystem* di rete simile a NFS, che supporta operazioni a *filesystem* disconnesso, il *caching* persistente e altre funzionalità avanzate che lo rendono particolarmente adatto ad un utilizzo su reti lente o inaffidabili e su *computer* portatili. Esso è incluso di serie nelle versioni di *Linux* a partire dalla 2.2. (**Ulteriori informazioni**)

Linux permette di condividere *filesystem* e stampanti anche secondo protocolli propri di altre piattaforme, come *Apple Macintosh*, mediante il pacchetto *Netatalk* (***Netatalk***, ***Netatalk***) o *Novell* NCP (si veda a proposito il documento IPX-*HOWTO*). Volendo condividere *filesystem* con macchine *Windows* si può utilizzare il pacchetto Samba, che verrà descritto in una unità didattica a parte.

Linux - AUTOFS (automount)

Nei sistemi *UNIX* ogni disco contenente un *filesystem* per essere accessibile deve essere montato in una *directory* mediante il comando *mount* (oppure al *boot* mediante `/etc/fstab`). `Autofs` permette di automatizzare tale l'operazione anche a sistema funzionante, in modo da non dover mantenere sempre montati i *filesystem* che non si utilizzano. In particolare può essere usato assieme a NFS per permettere ad un utente di utilizzare indifferentemente più macchine di una rete **NIS** senza dover tenere perennemente montate su ciascuna tutte le *home directory* degli utenti.

Per usufruire di tale funzionalità il *kernel* di *Linux* deve essere compilato con il supporto per `autofs` (in alternativa può essere caricato il modulo `autofs.o`). Inoltre sono necessari **i programmi utente e il demone**.

Tutte le richieste di accesso effettuate all'interno di una determinata *directory* (*master-directory*) vengono intercettate dal sistema e se la richiesta in questione riguarda una *subdirectory* presente nella configurazione del servizio *automount* ma attualmente non montata, questa viene montata automaticamente nel *filesystem*, in modo trasparente all'utente.

Anche l'operazione di rilascio viene gestita in automatico non appena la *directory* risulti inutilizzata per un certo periodo di tempo. Il *filesystem* da montare può essere un disco locale, un dispositivo rimovibile (cdrom, *floppy*) o un volume condiviso da un *server* di rete.

La configurazione di *automount* consta di un *file* di configurazione principale `/etc/auto.master` che contiene la lista delle *master-directory* nel formato

```
<master-dir> <file configurazione> <opzioni>
```

Per ogni *master-directory* viene definito il relativo *file* di configurazione ed eventuali opzioni, ad esempio `--timeout=xxx` con la quale si imposta il tempo di inutilizzo del *filesystem* prima che questo venga automaticamente smontato.

I *file* con configurazione di ogni *master-directory* ha il seguente formato:

```
<dir> <options> <filesystem>
```

in cui per ogni *subdirectory* si specificano le opzioni da passare al comando *mount*, ad esempio:

```
floppy -fstype=auto :/dev/floppy
cdrom -fstype=iso9660,ro :/dev/cdrom
```

Linux - Servizi Internet/Intranet

In questo capitolo viene descritto come configurare i principali servizi per Internet/Intranet. Gli esempi si riferiscono in modo specifico a *Linux Red Hat* 7.x.

I servizi descritti sono i seguenti:

- Telnet, **SSH** , RCP: servizi di **accesso al sistema** .
- ***Bind*** : *Domain Name Server*.
- **FTP** : trasferimento di *file*.
- ***Apache*** : *Web server*

Linux - Servizi di accesso al sistema

L'accesso al sistema in emulazione di terminale è possibile mediante il protocollo telnet. Il *client* telnet può essere utilizzato anche come strumento diagnostico, in quanto consente il collegamento diretto con qualunque porta TCP. Ad esempio, volendo testare il funzionamento del *server* POP3, è possibile eseguire il comando telnet *nameserver* 110.

Per fornire accesso al sistema mediante il protocollo telnet, in *Linux* si avvia il *server* `in.telnetd` con `inetd`, filtrando l'accesso mediante il TCP *wrapper*, conetenete una linea simile alla seguente in `/etc/inetd.conf`:

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

Data la funzione del servizio telnet, è buona norma restringerne l'accesso solamente ai *client* fidati. Il *server* telnetd non necessita di particolari configurazioni, salvo la possibilità di personalizzare il messaggio che viene mostrato al *client* all'inizio del collegamento (`/etc/issue.net`).

In alternativa a telnet è possibile utilizzare i cosiddetti programmi r* (`rlogin`, `rsh`, `rcp`, ...) i quali oltre all'accesso in emulazione di terminale permettono l'esecuzione remota di comandi e la copia di *file* da e verso *server* remoti (`rcp`). È possibile utilizzare il *file* di configurazione `/etc/hosts.equiv` oppure un *file* `.rhosts` posto nella *directory* di un utente per permettere l'accesso a particolari utenti/indirizzi IP senza richiesta di *password*. Tale funzione è particolarmente utile all'interno di *shell script*, ma pericolosa dal punto di vista della sicurezza.

Linux - SSH

L'utilizzo di telnet e dei programmi `r-*` è sconsigliato, principalmente perché il traffico e le *password* vengono trasferite in chiaro. Al loro posto si dovrebbe utilizzare `ssh`, che offre funzioni analoghe e codifica il traffico mediante algoritmi crittografici. Esistono due versioni del *software*, `ssh` e `ssh2`, parzialmente incompatibili fra loro. In entrambi i casi si tratta di *software* non disponibile come *Open Source*.

Per ovviare a tale mancanza, è stata creata *OpenSSH*, disponibile sotto licenza GPL. Questa viene fornita di serie con *Linux* e verrà utilizzata come riferimento in questo capitolo.

Le funzioni offerte da *OpenSSH* sono molteplici e pertanto vedremo solamente le più semplici. Alcuni esempi di utilizzi sono i seguenti:

- connessione sicura ad un *server* in emulazione di terminale (`ssh nomeserver`);
- scambio sicuro di *file* (`scp utente@nomeserver:/disk2/documenti/file.doc file.doc`);
- *tunnelling* codificato per il protocollo X11;
- realizzazione di versioni codificate dei servizi;
- ridirezione di porte TCP su canali sicuri;
- realizzazione di tunnel VPN per collegamenti sicuri fra reti (PPP-over-SSH);
- utilizzo all'interno di *shell script*: è possibile l'autenticazione sia mediante il meccanismo `rhosts` tipico di rsh, sia utilizzando chiavi crittografiche;

OpenSSH viene lanciato come un *server standalone*. La configurazione avviene mediante i *file* che si trovano nella *directory* `/etc/ssh`. Il controllo degli accessi può avvenire sia mediante le apposite direttive *AllowHosts* e *DenyHosts* in `/etc/ssh/sshd_conf`, sia utilizzando il TCP *wrapper*, in quanto sshd è compilato con il supporto della libreria `libwrap`.

Per il corretto funzionamento di SSH è necessario generare delle chiavi crittografiche per l'autenticazione. Tale operazione in genere viene svolta in modo automatico dagli *script* di installazione del pacchetto.

Per maggiori informazioni si può fare riferimento alla documentazione in linea oppure al sito
http://www.openssh.org/

Linux - Name Server

Per la comprensione degli esempi che seguono è necessario disporre delle conoscenze di base relative al funzionamento del sistema DNS.

Il servizio di risoluzione dei nomi, ovvero la trasformazione di un indirizzo da nome simbolico ad indirizzo IP, viene fornito in molti sistemi *UNIX* attraverso il *server* DNS *bind*.

Il programma che gestisce le richieste (*named*) viene attivato come un demone. Il *file* di configurazione di *bind* è `/etc/named.conf` (nelle versioni più vecchie di *named*, al posto del *file* `/etc/named.conf`, si usava `/etc/named.boot`, con una diversa sintassi).

Per il controllo del *server* DNS si utilizza il programma `ndc`. In particolare i comandi `ndc start` e `ndc stop` permettono rispettivamente di attivare e disattivare il servizio.

Le zone, ovvero le tabelle contenenti le definizioni dei domini, si trovano nella *directory* definita nel *file* di configurazione:

```
options {
directory /var/named;
};
```

La configurazione del *server* DNS dipende dalla funzione che esso deve effettuare, in particolare dal fatto che esso debba essere utilizzato per fornire delle zone (DNS principale o secondario per un dominio Internet) oppure debba solamente occuparsi di risolvere i nomi per conto dei *client* di una rete locale (*cache* DNS).

Per ulteriori informazioni si può fare riferimento alla **FAQ di *BIND*** oppure consultare i seguenti documenti:

- *DNS and BIND, 4th edition, Cricket Liu and P. Albitz, O'Reilly & Associates*.
- *Linux DNS Server Administration, Craig Hunt, Sybex*.
- ***Bind Homepage***.

Linux - Servizio di risoluzione di nomi per i client della rete locale

Il seguente esempio mostra come configurare *BIND* per fornire il servizio di nomi per i *client* di una rete locale:

```
options {
directory /var/named;
};

zone . {
type hint;
file named.root;
};
```

Poiché il meccanismo di risoluzione dei nomi opera in modo gerarchico e non sono definite altre zone, tutte le richieste di risoluzione dei nomi avvengono utilizzando la zona speciale ., mappata in un *file* che contiene la lista dei *root nameserver*.

Ad esempio se un *client* locale richiede la risoluzione del nome *www.sottodominio.dominio.com*, *named* verifica se è definita localmente la zona sottodominio.dominio.com. Non essendo questa definita, la ricerca continua con le zone dominio.com, .com e infine ., corrispondente al livello gerarchico più elevato nello spazio di nomi del DNS.

La zona . è contenuta nel *file* `named.boot`, che dovrebbe essere tenuto aggiornato con quello fornito da **Internic**. Essa ovviamente non contiene il *database* di tutti i nomi di dominio definiti, bensì gli indirizzi dei *server* a cui richiedere tali informazioni (*root nameserver*).

Uno di essi viene contattato per ottenere informazioni su quali *server* DNS hanno autorità per la zona .com e la procedura si ripete fino ad ottenere l'indirizzo dei *server* DNS che contiene le informazioni sulla zona sottodominio.dominio.com, a cui viene chiesto l'indirizzo (IN A) corrispondente a www.sottodominio.dominio.com.

Ovviamente *named* non ripete ogni volta tutte queste operazioni, ma tiene in memoria una *cache* delle zone a cui ha già acceduto, in modo da velocizzare le operazioni ed evitare richieste inutili (il tempo di validità di una zona nella *cache* dipende dai valori definiti all'interno della zona stessa).

Volendo limitare l'utilizzo del *nameserver* ai soli *client* della rete locale si possono definire delle apposite *access list*:

```
options {
allow-query { 192.168.10.0/24; localhost; };
};
```

Un altro metodo per fornire la risoluzione dei nomi ad una rete locale è quello di non utilizzare la zona . ma di passare invece tutte le richieste ad un altro DNS. Il *file* `named.conf` in questo caso diventa:

```
options {
directory /var/named;
forwarders { 111.112.113.114; };
};
```

Linux - Risoluzione degli indirizzi di rete LAN

Nel caso si desideri utilizzare *BIND* per la risoluzione degli indirizzi delle macchine nella rete locale, si deve definire in *named.conf* la zona relativa al dominio utilizzato:

```
zone intranet.miodominio.com {
type master;
file intranet.miodominio.com.zone;
};
```

Inoltre è buona norma definire anche la zona per la risoluzione inversa da indirizzo IP a nome simbolico:

```
zone 1.168.192.in-addr.arpa {
type master;
file 192.168.1.rev;
};
```

Le relative tabelle devono essere create all'interno della *directory* definita in `named.con`, ad esempio `/var/named/intranet.miodominio.com.zone`:

```
@ IN SOA server.miodominio.com. root.miodominio.com. (
2002031800 ; Serial
28800 ; Refresh
7200 ; Retry
```

```
604800 ; Expire
86400 ) ; Minimum
NS server.miodominio.com.
pc001 A 192.168.1.1
pc002 A 192.168.1.2
```

La tabella di riversa `/var/named/192.168.1.rev` diventa:

```
@ IN SOA server.miodominio.com. root.miodominio.com. (
2002031800 ; Serial
28800 ; Refresh
7200 ; Retry
604800 ; Expire
86400 ) ; Minimum
NS server.miodominio.com.
1 PTR pc001.intranet.miodominio.com.
2 PTR intranet.miodominio.com.
```

Le due tabelle devono essere tenute sincronizzate fra di loro ed i valori del campo *Serial* incrementati ad ogni modifica dei *file* (non è importante che l'incremento sia unitario: nell'esempio è stata usata la convenzione diffusa di utilizzare la data di modifica seguita da due cifre usate per distinguere eventuali modifiche effettuate lo stesso giorno).

Per ulteriori dettagli sul formato delle zone si faccia riferimento ad un documento di informazioni specifico sull'argomento.

Linux - Utilizzo come nameserver per un dominio Internet

Volendo utilizzare *BIND* in un *server* accessibile su Internet per la risoluzione dei nomi di un dominio, la definizione della zona è del tutto analoga a quanto visto nel caso del *nameserver* locale. Si deve solamente aver cura di definire i record relativi ai *server* a cui deve essere consegnata la posta elettronica del dominio (MX - *Mail Exchangers*).

Generalmente per un dominio sono richiesti più *server* DNS, possibilmente posti su reti diverse, in modo da assicurare un certo grado di ridondanza. I *nameserver* aggiuntivi vengono anche chiamati DNS secondari. Essi prelevano copia delle zone da un *nameserver* primario, che viene definito mediante una linea simile alla seguente in *named.conf*:

```
zone test.com {
type slave;
masters { 213.222.212.195; };
file test.com.zon;
};
```

Nel *server* primario è consigliabile restringere la possibilità di trasferimento di una zona solo ai *nameserver* secondari, in modo da evitare che la lista completa delle macchine presenti nella propria rete possa cadere in mano di estranei:

```
zone test.it {
type master;
file test.it.hosts;
allow-transfer { 193.6.122.33; };
};
```

Linux - Sicurezza del server DNS

Esistono diversi tipi di attacchi che possono essere portati a *named*. Pertanto è bene curare con attenzione il fattore sicurezza. Nelle versioni di *BIND* recenti esistono parecchie direttive che possono essere inserite nel *file* di configurazione allo scopo di rendere più difficili eventuali attacchi.

In particolare, *BIND* dovrebbe essere sempre fatto funzionare con i permessi di un utente non privilegiato. Per far questo è necessario modificare lo *script* di avvio `/etc/rc.d/init.d/named` aggiungendo nella linea di comando di *named* le opzioni:

- -u utente;
- -g gruppo;

Eventualmente è anche possibile *restringere la visibilità ad una porzione limitata del filesystem* mediante il meccanismo di `chroot` utilizzando l'opzione -t *directory*. In questo caso è necessario spostare i *file* di configurazione in una posizione tale che essi siano accessibili a *named* con il percorso corretto anche dopo il `chroot`.

Ad esempio, volendo restringere la visibilità al solo sottoalbero `/home/named` occorrerà creare il *file* di configurazione in `/home/named/etc/named.conf` e le zone in `/home/named/var/named`. Nel caso si utilizzi *BIND* come DNS secondario per un dominio, sarà necessario rendere disponibili con i percorsi corretti anche i programmi che si occupano della replica delle zone dal DNS principale (*named-xfer*) e le eventuali librerie dinamiche da essi utilizzate. I dettagli sono spiegati nel documento Chroot-*BIND*8-*HOWTO* oppure sul documento **chrootdns**.

Per altri dettagli sulle funzioni relative alla sicurezza si faccia riferimento alla documentazione fornita col *software* oppure al DNS-*HOWTO*.

Linux - FTP

Il *server* per il protocollo FTP (*File Transfer Protocol*) fornito di serie con *Linux* è `wu-ftp`, della *Washington University*. Esso viene fornito preconfigurato per essere lanciato mediante `inetd`, ma può essere utilizzato anche in modalità *standalone* richiamandolo da uno *script* di avvio mediante l'opzione -s.

I *file* di configurazione di `wu-ftpd` sono i seguenti:

- `/etc/ftpusers`: contiene la lista degli *account* con cui non possibile effettuare una connessione al *server* mediante FTP. Per motivi di sicurezza di solito l'accesso viene vietato agli utenti di sistema (*root*, *bin*, *mail*, ...).
- `/etc/ftpaccess`: se `ftpd` viene eseguito con l'opzione -a, questo *file* viene utilizzato per la gestione degli accessi. È possibile definire diverse classi di utenti a cui associare determinati permessi sulle *directory* (`chmod`, *`delete`*, *`overwrite`*, *`rename`*, ...). Per ogni classe di utenti è inoltre possibile scegliere il numero massimo di connessioni accettabili contemporaneamente.
- `/etc/ftpconversions`: viene usato per determinare le modalità di conversione dei *file* compressi. Ad esempio è possibile fare in modo che richiedendo il trasferimento di un *file* esistente aggiungendo al nome l'estensione `.gz`, esso venga compresso al volo con `gzip`. Analogamente si può fare in modo che, dato il nome di una *directory* con aggiunta l'estensione `.tar.gz`, `ftpd` crei automaticamente il corrispondente archivio compresso.
- `/etc/ftphosts`: permette di filtrare l'accesso al servizio FTP da determinati utenti o indirizzi IP. Se il *server* viene attivato mediante `inetd`, è possibile creare delle *access list* anche utilizzando il TCP *wrapper*.

Lo stato del servizio FTP può essere tenuto sotto controllo mediante i comandi `ftpcount` (visualizza il numero di utenti connessi ed il numero massimo di connessioni ammissibili per ogni classe di utenza) e `ftpwho` (mostra la lista e le operazioni che stanno compiendo gli utenti attualmente collegati al sistema). Il log completo delle operazioni viene tenuto nel *file* `/var/log/xferlog`.

Linux - FTP anonimo e modalità chroot

Se si vuole permettere l'accesso al *server* FTP in modo anonimo, è necessario creare in `/etc/passwd` l'utente speciale ftp. La *password* ad esso associata non è importante, in quanto non viene controllata (è comunque necessario assegnarla o sostituirla col valore * nel *file* delle *password* onde evitare accessi indesiderati attraverso altri servizi).

Per utilizzare l'FTP anonimo si accede al sistema identificandosi come ftp, oppure come *anonymous*. Al posto della *password* di norma si utilizza il proprio indirizzo di posta elettronica.

L'accesso anonimo si differenzia da quello autenticato per il fatto che viene eseguito un `chroot` nella *home directory* dell'utente ftp (ad esempio `/home/ftp`). In tale *directory* pertanto deve essere ricreata una struttura di *filesystem* che contenga i seguenti *file*, necessari al demone `ftpd` per listare il contenuto delle *directory*:

```
/home/ftp/etc/passwd
/home/ftp/etc/group
/home/ftp/etc/ld.so.cache
/home/ftp/bin/ls
/home/ftp/bin/gzip
/home/ftp/lib/libc.so.6
/home/ftp/lib/ld-linux.so.2
```

Per motivi di sicurezza sarebbe preferibile che tali *file* fossero di proprietà dell'utente *root* e non scrivibili ad altri utenti.

Il contenuto dei *file* `passwd` e *`group`* può essere fittizio, in quanto vengono utilizzati solamente dal comando `ls` per visualizzare lo *username* e il nome del gruppo proprietario dei *file*. In caso contrario nell'*output* di `ls` verrebbero indicati i valori numerici di UID e GID.

Mediante la direttiva `guestuser nome` di `/etc/ftpaccess` è possibile definire degli utenti per i quali l'accesso deve essere eseguito in modalità `chroot`. Così facendo, l'utente non può risalire sopra la propria *home directory* e vedere il contenuto dell'intero *filesystem*. Per ogni utente per cui si attiva tale opzione deve essere ricreata la stessa struttura di *directory* vista per l'utente ftp anonimo. Inoltre il campo contenente la *home directory* nel *file* `/etc/passwd` deve essere modificato come nell'esempio che segue:

```
mrossi:x:306:100:Mario Rossi:/home/mrossi/./:/bin/bash
```

Volendo creare molti *account* che utilizzino il meccanismo di `chroot`, conviene sostituire il *server* `wu-ftpd` con il programma `Pureftpd`, il quale non necessita di dover ricreare la struttura di *directory* per ogni utente.

Linux - Apache

Il *server* HTTP più utilizzato in ambiente *UNIX* è *Apache*. In *Linux* esso viene avviato come un *server* a se stante mediante lo *script* `/etc/rc.d/rcX.d/S85httpd`, tuttavia è possibile anche utilizzarlo attraverso `inetd`, avendo cura di inserire la direttiva *`ServerType`* `inetd` nel *file* di configurazione `httpd.conf`. Per utilizzi meno impegnativi esistono dei *server* HTTP più leggeri e semplici da configurare, ad esempio BOA.

Apache ha una struttura modulare, in cui è possibile aggiungere nuove funzionalità mediante la ricompilazione del codice oppure, nelle versioni recenti, semplicemente aggiungendo un nuovo modulo, fornito sotto forma di libreria dinamica, nella *directory modules* ed adattando opportunamente il *file* di configurazione.

In *Red Hat Linux Apache* viene fornito precompilato sotto forma di pacchetti RPM contenenti il *server* base (http) e i

moduli per le funzionalità aggiuntive più utilizzate (`mod_ssl`, `mod_perl`, ...).

Nel sito del progetto ***Apache***, è presente una lista di tutti i moduli disponibili per *Apache* (*Module Registry*), comprendente sia quelli inclusi nella distribuzione del *server* che quelli realizzati da terzi. Esistono moduli adatti per diverse funzioni (*parsing* di linguaggi, metodi di autenticazione, *logging* ...).

Per maggiori informazioni si invita a leggere la documentazione fornita in linea col prodotto oppure a **visitare il sito**, che contiene una lista di siti e libri adatti ad approfondire l'argomento. Ulteriori informazioni sono presenti anche nei seguenti *HOWTO* di *Linux*: *Apache*-Introduzione-*HOWTO*, *Apache-Compile-HOWTO*, *ISP-Setup-RedHat-HOWTO*, *SSL-RedHat-HOWTO*, WWW-*HOWTO*.

Linux - Configurazione di Apache

La configurazione di *Apache* avviene mediante i *file* contenuti nella *directory*

`/etc/httpd/conf`

(è possibile specificare un altro percorso utilizzando l'opzione -d nella linea di comando di `httpd`), il più importante dei quali è `httpd.conf`, che permette di definire tutti gli aspetti della configurazione del *server*. In alcune installazioni la configurazione è divisa fra i *file* `httpd.conf`, `srm.conf` e `access.conf`. Una volta modificato un *file* di configurazione è necessario riavviare il servizio con `/etc/rc.d/init.d/httpd restart`.

Volendo sono disponibili delle interfacce grafiche o via *Web* che consentono una configurazione semplificata del *server* (si faccia riferimento al sito per una lista del *software* disponibile).

Particolare importanza ha la definizione delle due *directory*:

- *ServerRoot* (esempio: `/etc/httpd/conf`), contenente i *file* di configurazione;
- *DocumentRoot* (esempio: `/var/www/html`), a partire dalla quale si inseriscono i documenti che si desidera rendere disponibili attraverso il protocollo HTTP.

Mediante la direttiva *UserDir* è anche possibile permettere agli utenti del sistema di creare una propria *directory* personale, ad esempio `/home/nomeutente/public_html`, che sarà visibile come `http://nomeserver/~nomeutente`.

Mediante la direttiva *Directory* di `httpd.conf` si possono definire i permessi di accesso e le *Options* di *default* per la diverse *directory*, che vengono applicati in modo ricorsivo anche alle sotto*directory*.

Se si specifica l'opzione *AllowOverride* è possibile permettere la modifica di permessi e *Options* da parte degli utenti mediante un *file* di nome `.htaccess` posto nella *directory* desiderata, che vale in modo ricorsivo anche per tutte le sotto*directory*.

```
<Directory /var/www/html>
Options Indexes Includes ExecCGI
AllowOverride None
order allow,deny
allow from all
</Directory>
```

Linux - Restrizione di accesso ad una directory

I metodi base di autenticazione forniti da *Apache* filtrano gli accessi in base all'indirizzo del *client* oppure richiedendo

uno *username* ed una *password* che vengono confrontati con quelli contenuti in un *file* creato mediante il programma `htpasswd`. Esistono moduli aggiuntivi (`mod_auth_*`) che consentono altri tipi di autenticazione, ad esempio utilizzando il metodo standard PAM, un *server* di autenticazione (*Radius*, *Windows* NT, ...) oppure una tabella di *database* (`mod_auth_mysql`, ...).

Il seguente esempio di *file* `.htaccess` mostra come limitare l'accesso *GET* e *POST* ad una *directory* ai soli utenti definiti nel *file* `passwd_sito`, che per ragioni di sicurezza viene conservato in una posizione del *filesystem* non accessibile mediante HTTP.

```
AuthUserFile /home/pippo/SECURE/passwd_sito
AuthGroupFile /home/pippo/SECURE/group_sito
AuthName Password_Approvazione
AuthType Basic
<Limit GET POST>
require valid-user
</Limit>
```

Un esempio di restrizione in base all'indirizzo del *client* è invece il seguente:

```
<Limit GET POST>
order deny,allow
deny from all
allow from .test.it
allow from 192.168.22.
</Limit>
```

Mediante la direttiva *Options* è possibile abilitare o restringere l'utilizzo di alcune funzioni problematiche per la sicurezza, ad esempio inibire l'utilizzo di *script* CGI in particolari *directory* oppure fare in modo che il *server* non segua i link simbolici.

```
Options ExecCGI FollowSymLinks
```

Linux - Sicurezza

Oltre alle restrizioni appena viste, esistono altre funzioni di *Apache* che permettono di gestire in modo sicuro il servizio HTTP, come la possibilità di far funzionare il *server* con i permessi di un utente non privilegiato, mediante le direttive *User* e *Group*, oppure di fare girare i programmi CGI con i permessi dell'utente proprietario. Nelle versioni recenti di *Apache* non è possibile avviare il *server* con i permessi di *root*, a meno di ricompilare il programma attivando una particolare opzione.

Linux - Supporto per SSL

Esistono due metodi per fornire ad *Apache* il supporto per la codifica crittografica dei dati mediante SSL:

- *Apache*-SSL: si tratta di una versione del *server* modificata in modo da contenere il supporto per SSL;
- `mod_ssl`: è un modulo aggiuntivo che può essere utilizzato nella versione standard di *Apache*.

In entrambi i casi il funzionamento si basa sulla libreria *OpenSSL*.

Linux - Log degli accessi e server virtuali

Il *logging* degli accessi avviene nel *file*

- `/var/log/httpd/access_log`,

mentre eventuali errori vengono mandati in

- `/var/log/httpd/error_log`.

È possibile variare il percorso di tali *file* utilizzando le direttive `TransferLog` e `ErrorLog`. Il formato standard per questi *file* è il *Common Logfile Format*, compatibile con molti programmi per generare statistiche di accesso. È comunque possibile utilizzare anche altri formati standard (esempio: *Combined*) oppure definire un proprio formato.

È possibile utilizzare *Apache* per creare *server* virtuali, ovvero per gestire i siti *Web* di più domini diversi, sia utilizzando per ogni sito un indirizzo IP diverso, sia condividendo uno stesso indirizzo IP fra più siti.

Mediante la direttiva di configurazione *VirtualHost* è possibile ridefire per ogni *server* virtuale molte opzioni di *Apache*, fra cui quelle relative ai *file* di log, in modo da avere un report separato per ogni dominio.

Linux - Pagine dinamiche e scripting

Oltre che per fornire pagine statiche, è possibile utilizzare *Apache* anche per creare siti dinamici, utilizzando dei programmi CGI oppure uno dei molti linguaggi di *scripting* disponibili (PHP, Perl, TCL, SSI, ...). Una piattaforma di sviluppo molto diffusa in ambiente *Linux* è composta del *server Apache* utilizzato assieme al linguaggio PHP ed al *database MySQL*.

Maggiori dettagli sul linguaggio PHP possono essere reperiti, assieme ad una ampia bibliografia, sul **sito**, nel manuale fornito col linguaggio e nel PHP-*HOWTO*.

Sicurezza in Linux - Proxy

Per *Linux* esistono diversi *software* adatti ad essere utilizzati come *proxy* per permettere l'accesso all'esterno da parte degli utenti di una rete interna con indirizzi privati. Utilizzando assieme diversi prodotti è possibile utilizzare i protocolli più diffusi (HTTP, FTP, telnet, POP3, H.323, ...).

Una soluzione di *firewall/proxy* molto completa e compatibile con quasi tutti i protocolli è ***SOCKS***. Tuttavia esso non è molto utilizzato in quanto ha lo svantaggio di richiedere delle modifiche ai programmi preesistenti.

TIS Firewall Toolkit (FWTK), è una raccolta di *firewall* basati su *proxy*, compatibile con i protocolli FTP, HTTP, NNTP, RLOGIN, TELNET e SSL.

Per ogni protocollo che si intende utilizzare è necessario lanciare mediante `inetd` o `xinetd` un apposito demone, che riceve le richieste di connessione su una porta TCP (ad esempio 2323 per telnet-gw), le filtra in base all'indirizzo IP sorgente, il nome dell'utente o altri parametri, e le passa al servizio reale.

Nonostante FWTK sia utilizzabile gratuitamente, non si tratta di *software Open Source* e per ottenerlo è necessaria una procedura di registrazione sul sito del produttore.

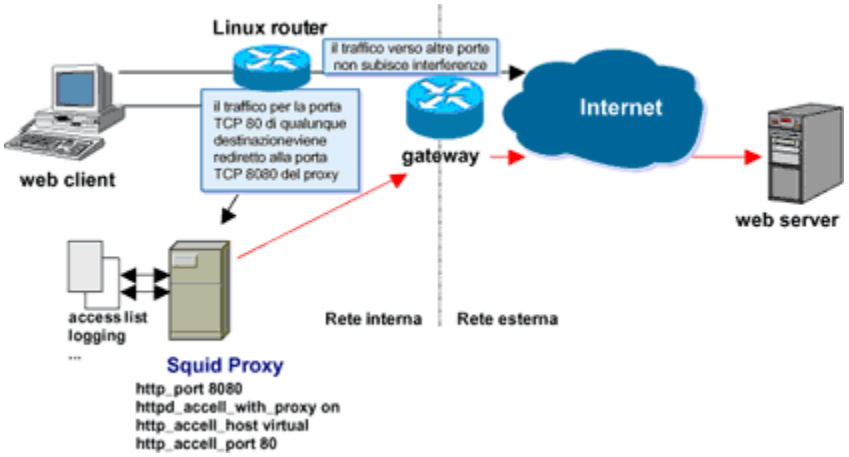
Sicurezza in Linux - Squid

Fra i molti *proxy* per il protocollo HTTP, il più diffuso è **Squid**. Si tratta di un programma molto flessibile e completo, dotato di una vasta serie di programmi aggiuntivi. Squid offre fra le altre cose la possibilità di creare delle gerarchie di *proxy*.

Squid offre anche funzionalità di *cache* delle pagine a cui si è acceduto. Questo permette di ridurre il traffico di rete, in quanto le pagine già in memoria non vengono ricaricate, ma bensì ne viene solamente verificato lo stato di aggiornamento. Nel caso di *file* di dimensioni significative, come le immagini, questo comporta notevoli vantaggi sia in termine di velocità di accesso che di *bytes* trasferiti.

La configurazione di Squid avviene di solito mediante la modifica diretta del *file* di testo `/etc/squid/squid.conf`, ma viene anche fornito un semplice *script* CGI che permette la configurazione di alcuni aspetti attraverso una interfaccia *Web*. Altri programmi di configurazione più complessi possono essere reperiti nel sito **SQUID** (ad esempio esistono dei moduli per `webmin` e `linuxconf`).

La gestione delle *access list* di Squid può essere estesa utilizzando un *authenticator* esterno, che consente di implementare nuovi metodi di autenticazione, ad esempio basati su *username*/*password* o sull'orario di accesso. È inoltre possibile utilizzare un *redirector* esterno per riscrivere gli indirizzi URL inseriti dagli utenti. Questo consente la creazione di *black list*.



È possibile utilizzare Squid assieme alle funzionalità di filtraggio del traffico presenti in *Linux* per costruire un *transparent proxy*. Esso consiste nel redirigere in modo automatico verso il *proxy* tutto il traffico generato dalle macchine della rete interna e destinato alla porta TCP 80 di un indirizzo esterno. Per far ciò è ovviamente necessario predisporre la topologia della rete in modo che tutto il traffico della rete passi per una singola macchina su cui sia possibile intercettarlo e ridirigerlo.

Essendo la funzione di transparent *proxy* svolta a livello di pacchetti IP, non sarà necessario configurare il supporto per il *proxy* sui singoli *browser*, che, anzi, non si accorgeranno neppure di passare attraverso un *proxy*. Tale soluzione permette di effettuare il *caching* automatico del traffico *Web* della propria rete, oppure di applicare funzioni di *logging*, *access list* o *black list*.

Approfondimenti

Configurazione di un collegamento PPP su un server Linux Red Hat

Franco Callegati
Paolo Presepi
Riccardo Gori
7.3.1 (Installare e configurare applicazioni client/server su un server quali: e-mail, FTP, Web, sistemi di messaggistica, chat, eccetera)

Configurazione della rete in Red Hat Linux

Il *software* di configurazione presente nella distribuzione *Red Hat Linux* consente una gestione semplificata delle interfacce di rete e delle funzionalità di *networking*. Le informazioni relative vengono tenute nei *file* di configurazione `/etc/sysconfig/network` e `/etc/sysconfig/network-scripts` e vengono utilizzate dagli *script* di *boot* che si occupano di attivare la rete (`/etc/rc.d/init.d/network`).

Nel caso si modifichino a mano tali *file*, per rendere attive le modifiche è necessario far ripartire i servizi di rete:

```
# /etc/rc.d/init.d/network restart
Shutting down interface eth0 [ OK ]
Disabling IPv4 packet forwarding [ OK ]
Disabling IPv4 automatic defragmentation [ OK ]
Enabling IPv4 packet forwarding [ OK ]
Bringing up interface lo [ OK ]
Bringing up interface eth0 [ OK ]
```

Volendo fermare o far ripartire singolarmente una determinata interfaccia di rete, si possono utilizzare i comandi `ifup` e `ifdown`:

```
# ifdown eth0
# ifup eth0
```

Il *file* `/etc/sysconfig/network` contiene le impostazioni generali del *software* di rete, come il nome del sistema, comprensivo del dominio di appartenenza (FQDN, *Full Qualified Domain Name*) e l'indirizzo del *gateway* di *default*.

```
NETWORKING=yes
FORWARD_IPV4=true
HOSTNAME=mionome.miodominio.com
DOMAINNAME=miodominio.com
GATEWAY=193.43.98.254
```

La linea `FORWARD_IPV4=true` indica che la macchina deve eseguire il *forwarding* dei pacchetti IP fra le diverse interfacce, operazione che non è abilitata per *default*.

Nella *directory* `/etc/sysconfig/network-scripts/` sono presenti i *file* contenenti i dati di configurazione relativi alle singole interfacce presenti nel sistema. Ad esempio `ifcfg-eth0` contiene i dati di configurazione della prima scheda *ethernet* presente nel sistema:

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.1.255
IPADDR=192.168.1.23
NETMASK=255.255.255.0
NETWORK=192.168.1.0
```

```
ONBOOT=yes
```

L'esempio presentato si riferisce ad una configurazione statica (*BOOTPROTO=static*) della interfaccia `eth0`, in cui vengono specificati esplicitamente l'indirizzo IP, quello della rete, la *netmask* e l'indirizzo di *broadcast*. L'ultima riga del *file* indica che l'interfaccia deve essere attivata al *boot* del sistema.

È possibile in alternativa configurare l'interfaccia per acquisire il proprio indirizzo mediante il protocollo BOOTP (*BOOTPROTO=bootp*) oppure da un *server* DHCP (*BOOTPROTO=dhcp*).

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

Interfacce Point-to-Point

In *Linux* sono disponibili diversi protocolli di connessione punto-a-punto su collegamento seriale, di cui i due più usati sono PPP e SLIP. Analizzeremo solamente il primo in quanto più diffuso.

Trattandosi di un protocollo *peer-to-peer*, è possibile utilizzare PPP, oltre che per accedere ad Internet attraverso un Internet *Provider*, anche per connettere fra loro due reti oppure per offrire l'accesso ad altre macchine.

Generalmente il collegamento avviene via modem oppure mediante un cavo seriale null-modem, ma è anche possibile incapsulare una connessione PPP all'interno di pacchetti IP in modo da realizzare una VPN. Un caso particolare è dato dal protocollo PPPoE, utilizzato dai modem ADSL, il quale permette il *tunnelling* di PPP all'interno di *frame Ethernet*.

La configurazione di una interfaccia PPP è più complessa rispetto a quella di una interfaccia *Ethernet*, in quanto oltre alla connessione devono essere gestiti anche gli aspetti riguardanti la negoziazione degli indirizzi, l'autenticazione, la compressione, ...

PPP è composto da tre strati: un metodo per incapsulare pacchetti di dati su una linea seriale, un protocollo che gestisce il link (LCP, *Link Control Protocol*) ed uno strato che si occupa di gestire i differenti protocolli di rete che possono funzionare sopra PPP (nel caso del protocollo IP viene usato IPCP, IP *Control Protocol*).

In *Linux* l'incapsulamento dei dati avviene internamente al *kernel*, mentre il controllo della connessione viene gestito mediante il demone esterno `pppd`.

La presenza del supporto per il PPP nel *kernel* è confermata dal seguente messaggio al *boot*:

```
PPP: version 2.3.7 (demand dialling)
PPP line discipline registered.
```

La configurazione di `pppd` avviene specificando i parametri necessari nella linea di comando, oppure inserendoli nel *file* `/etc/ppp/options`. Nel caso di collegamento telefonico è possibile utilizzare il comando *chat* per inizializzare il modem, comporre il numero di telefono ed eseguire il *login* sul terminal *server* remoto. Esso utilizza un linguaggio di tipo *send-expect* per descrivere le stringhe da spedire in risposta a quelle ricevute dal modem.

Generalmente il collegamento PPP viene gestito mediante degli *script* in *shell* (ppp-*on*, ppp-*off*) contenenti i comandi necessari, tuttavia è possibile semplificare le operazioni utilizzando delle interfacce grafiche, come KPPP (compresa in KDE), WvDial, RP3 (*Red Hat Linux*). La configurazione e la gestione di PPP sono possibili anche mediante lo strumento standard *Linuxconf*.

Autenticazione

L'autenticazione in PPP può avvenire in diversi modi:

- eseguendo il *login* sull'*access server* e successivamente richiedendo una sessione in modalità PPP: in questo caso le relative procedure possono essere automatizzate mediante l'utilizzo del comando `chat`.
- mediante i metodi standard PAP/CHAP/MSCHAP: in questo caso è sufficiente specificare nella linea di comando di PPP oppure in `/etc/ppp/options` il metodo prescelto ed il nome dell'*account* ed inserire la *password* (*secret*) in uno dei *file* `/etc/ppp/pap-secrets` o `/etc/ppp/chap-secrets`. Essi possono contenere una o più linee nel seguente formato: `client server secret IP addresses`. La *password* da utilizzare viene selezionata dal `pppd` utilizzando come chiavi di ricerca in *chap-secrets* il nome specificato nella opzione *user*, l'indirizzo che viene assegnato alla macchina dal *router* remoto e l'indirizzo all'altro capo della connessione. Nel caso alcuni di questi valori vengano gestiti in modo dinamico, ad e esempio l'indirizzo IP, è possibile sostituire la *wildcard* * al posto dei dati che non si conoscono: `* * secret *`

Esempio di script PPP

Il seguente *script* mostra un esempio di come possa essere instaurato un collegamento PPP in modalità client. Per concludere la connessione è sufficiente terminare il processo `pppd`.

```
#!/bin/sh

TELEPHONE=0862123456 # Numero di telefono del provider
ACCOUNT=test # Username fornito dal provider
PASSWORD=test # Password fornita dal provider
LOCAL_IP=0.0.0.0
REMOTE_IP=0.0.0.0
export TELEPHONE ACCOUNT PASSWORD

exec /usr/sbin/pppd debug lock modem crtscts /dev/ttyS0 38400 \
asyncmap 20A0000 escape FF kdebug 0 $LOCAL_IP:$REMOTE_IP \
noipdefault netmask 255.255.255.0 defaultroute \
connect /etc/ppp/ppp-on-dialer
```

Le opzioni che nell'esempio sono specificate direttamente nella linea di comando di `pppd`, possono essere in alternativa inserite, una per riga, nel *file* `/etc/ppp/options`.

Nel caso si effettui una connessione con IP fissi, essi dovranno essere specificati nelle variabili `$LOCAL_IP` e `$REMOTE_IP`. Se si utilizza un indirizzamento dinamico, si possono invece lasciare gli indirizzi fittizi 0.0.0.0 e inserire fra le opzioni *noipdefault*.

Si noti che alcune delle variabili definite nello *script* vengono esportate in modo da renderle disponibili anche ai programmi lanciati dal `pppd`:

L'inizializzazione del modem e la composizione del numero di telefono del *provider* non vengono effettuati direttamente da `pppd`, ma vengono demandate allo *script* definito dalla opzione *connect*:

```
#!/bin/sh
exec chat -v "" AT OK ATZ OK ATM1L3X3 OK ATDT$TELEPHONE CONNECT
```

L'esempio precedente mostra una semplice sequenza *send-expect* necessaria a resettare il modem, inviargli una semplice stringa di inizializzazione e fargli comporre un numero di telefono.

Uno *script* come quello appena visto non è molto robusto, in quanto non è in grado di riconoscere eventuali errori, se non quello dovuto al *timeout*. Uno esempio più completo è il seguente, in cui vengono definiti alcuni dei messaggi di errore che il modem può ritornare.

```
exec chat -v TIMEOUT 3 ABORT '\nBUSY\r' ABORT '\nNO ANSWER\r'
ABORT '\nRINGING\r\n\r\nRINGING\r' '' \rAT 'OK-+++\c-OK' ATH0
TIMEOUT 30 OK ATDT$TELEPHONE CONNECT
```

È possibile utilizzare *chat*, oltre che per comporre il numero, anche per autenticare l'utente nel caso l'*access server* remoto preveda una procedura di *login*. Un esempio è il seguente:

```
chat -v ATZ OK ATD$TELEPHONE CONNECT \r ername: test ssword: test > ppp
```

Si noti che il carattere > è stato quotato per evitare che venga interpretato dalla *shell* come un operatore di ridirezione. Analogamente per il carattere \r, che serve per inviare un a capo.

Quelle appena viste solo solamente alcune delle opzioni usabili nel comando *chat*. Per ulteriori informazioni si può fare riferimento al manuale in linea (*man chat*).

Debugging

Inserendo fra le opzioni di `pppd` *debug* e k*debug* n è possibile tenere traccia nel log di sistema delle operazioni compiute da `pppd`. Essa può essere utilizzata per risolvere eventuali problemi di funzionamento.

```
Jan 24 15:56:28 freddy kernel: registered device ppp0
Jan 24 15:56:28 freddy pppd[6204]: pppd 2.3.10 started by root, uid 0
Jan 24 15:56:28 freddy pppd[6204]: Using interface ppp0
Jan 24 15:56:28 freddy pppd[6204]: Connect: ppp0 <--> /dev/modem
Jan 24 15:56:30 freddy kernel: PPP BSD Compression module registered
Jan 24 15:56:31 freddy kernel: PPP Deflate Compression module registered
Jan 24 15:56:31 freddy pppd[6204]: Unsupported protocol (0x803f) received
Jan 24 15:56:31 freddy pppd[6204]: local IP address 193.43.99.222
Jan 24 15:56:31 freddy pppd[6204]: remote IP address 193.43.99.1
...
Jan 24 16:27:49 freddy pppd[6204]: Terminating on signal 2.
Jan 24 16:27:49 freddy pppd[6204]: Connection terminated.
Jan 24 16:27:49 freddy pppd[6204]: Connect time 31.4 minutes.
Jan 24 16:27:49 freddy pppd[6204]: Sent 3370 bytes, received 4663 bytes.
Jan 24 16:27:49 freddy pppd[6204]: Exit.
```

Gli script /etc/ppp/ip-up e /etc/ppp/ip-down

Mediante lo *script* `/etc/ppp/ip-up` si possono eseguire automaticamente delle operazioni quando viene instaurato un collegamento PPP. Analogamente, *`ip-down`* permette di compiere delle operazioni quando esso termina.

In questo modo è per esempio possibile tenere un log della connessione, oppure prelevare la posta elettronica dal *provider* o aggiungere delle tabelle di *routing*.

Se fra le opzioni di `pppd` si inserisce `usepeerdns`, è possibile ottenere dall'*access server* remoto gli indirizzi dei *nameserver* da utilizzare: essi vengono passati allo *script* `/etc/ppp/ip-up`, come variabili d'ambiente DNS1 e DNS2, da utilizzare per aggiornare in modo automatico il *file* `/etc/resolv.conf`.

Un esempio di *script* *`ip-up`* è il seguente, che compie operazioni differenti a seconda che ci si connetta o meno alla

rete aziendale (la variabile $5 contiene l'indirizzo assegnato alla connessione).

```
#!/bin/sh
# tiene un log della connessione
echo `/bin/date` INIZIO >>/var/log/isdn.log

case "$5" in
192.168.*) # se collegato alla LAN aziendale, aggiunge una route
/sbin/route add -net 192.168.0.0 dev ppp0
# e monta un disco dal server NT
/bin/mount -t smbfs //ntserver/share /share
;;

*) # se collegato all'ISP, scarica la posta
/usr/local/bin/SCAMBIA_POSTA &
# e aggiorna la configurazione dei DNS
if [ ! "$DNS" = "" ]
then
echo DNS1 >/etc/resolv.conf
echo DNS2 >>/etc/resolv.conf
fi
;;
esac
```

Collegamento on demand

Mediante il programma `diald` è possibile automatizzare il collegamento PPP in presenza di traffico ed abbattere la connessione allo scadere di un determinato *timeout*. Si tratta di un programma versatile, che consente di filtrare i pacchetti che possono instaurare il collegamento e di assegnare un *timeout* diverso a seconda della tipologia di traffico (ad esempio aumentandolo nel caso siano attive delle sessioni di lavoro il cui funzionamento possa essere pregiudicato da una eventuale interruzione della linea).

Linux come Access Server

Mediante opportune configurazioni è possibile utilizzare `pppd` anche per permettere collegamenti entranti da parte di altre macchine. Per far ciò è necessario configurare il programma `getty`, che gestisce i modem in ingresso, in modo che esso riconosca il tipo di connessione e lanci il PPP invece della normale richiesta di *login* in modo testo. Esistono diversi modi per fare ciò, che sono spiegati nel PPP *HOWTO*.

Volendo utilizzare *Linux* per sostituire un *access server* commerciale, conviene indirizzarsi verso il pacchetto *Portslave*, il quale permette l'autenticazione degli utenti utilizzando il metodo standard PAM, oppure appoggiandosi ad un *server* TACACS o RADIUS.

ISDN

Mentre i *Terminal Adapter* ISDN attivi possono essere in generale utilizzati mediante il demone `pppd` standard, quelli passivi necessitano di un opportuno supporto da parte del *kernel*, fornito dal sottosistema **ISDN4*Linux***.

Se si utilizza il protocollo sincrono X.75, è necessario ricorrere al demone `ipppd` (recentemente i due programmi `pppd` e `ipppd` sono stati riuniti in un unico *software* in grado di gestire entrambi i protocolli). Salvo lievi differenze, la configurazione di `ipppd` è simile a quanto abbiamo visto per il `pppd`.

ADSL (PPPoE)

Il collegamento ADSL è possibile sia mediante il protocollo PPPoA (PPP-*over*-ATM), che mediante PPPoE (PPP-*over*-*Ethernet*). Nel primo caso è necessario compilare il *kernel* (versione 2.4) con il supporto per ATM, applicando le apposite *patch* disponibili su:

http://linux-atm.sourceforge.net/

Le istruzioni per l'installazione di PPPoA si trovano nel sito:

http://www.sfgoth.com/~mitch/linux/atm/pppoatm/

Per utilizzare PPPoE è invece possibile utilizzare sia una soluzione *kernel-based*, sia una soluzione più semplice, basata su un apposito demone, che realizza il *tunnelling* di una connessione PPP all'interno di *frame ethernet* sfruttando il normale `pppd` (PPPoE della *Roaring Penguin*, disponibile per *Linux* 2.2). Il supporto necessario a livello di *kernel* è già incluso in *Linux* 2.4 (dettagli).

Per maggiori informazioni sulla configurazione di ADSL si può fare riferimento al DSL-*HOWTO*.

Comandi basilari per il test della rete

Per verificare che le tabelle di *routing* funzionino in modo corretto si può utilizzare il comando ping, che testa la raggiungibilità di un dato indirizzo spedendo dei pacchetti di tipo ICMP ed aspettando una risposta dalla macchina remota. Esso permette fra le altre cose di conoscere il tempo impiegato dai pacchetti per compiere il tragitto di andata e ritorno (RTT, *Round Trip Time*). Per terminare si premano i tasti CTRL-C:

```
# ping www.pluto.linux.it
PING www.pluto.linux.it (147.162.126.3) from 62.98.87.226 :
56(84) bytes of data.
64 bytes from 147.162.126.3: icmp_seq=0 ttl=242 time=105.4 ms
64 bytes from 147.162.126.3: icmp_seq=1 ttl=242 time=141.1 ms
64 bytes from 147.162.126.3: icmp_seq=2 ttl=242 time=100.5 ms

--- www.pluto.linux.it ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 100.5/115.6/141.1 ms
```

Se si desidera conoscere il percorso esatto che compiono i pacchetti per raggiungere una data destinazione, si può utilizzare invece il programma *traceroute* (*tracert* nella versione *Windows*). Esso spedisce un tipo particolare di pacchetto ICMP che fa in modo che tutti i *router* per cui il esso transita spediscano indietro una risposta. In questo modo è possibile generare una traccia del percorso del pacchetto, che permette di capire se e dove ci sono problemi di connettività o quale *router* è la causa di eventuali rallentamenti nella connessione.

```
# traceroute www.interbusiness.it
traceroute to www.cs.interbusiness.it (151.99.250.4), 30 hops max, 40 byte packets
1 217.57.6.193 (217.57.6.193) 1.132 ms 1.806 ms 1.086 ms
2 62.86.50.61 (62.86.50.61) 7.795 ms 14.604 ms 5.071 ms
3 r-pd48-fa2.interbusiness.it (212.131.52.79) 7.945 ms 4.991 ms 4.998 ms
4 151.99.101.97 (151.99.101.97) 7.851 ms 14.313 ms 11.904 ms
5 r-mi213-fa4.interbusiness.it (151.99.75.218) 11.279 ms 12.783 ms 8.197 ms
6 r-rm99-ts21.interbusiness.it (151.99.98.109) 23.966 ms 17.211 ms 15.916 ms
7 r-rm179-fa4.interbusiness.it (151.99.29.213) 19.518 ms 17.472 ms 16.214 ms
8 r-rm80-fa2.interbusiness.it (151.99.29.8) 23.093 ms 28.061 ms 32.447 ms
9 212.131.81.38 (212.131.81.38) 40.216 ms 31.431 ms 62.16 ms
```

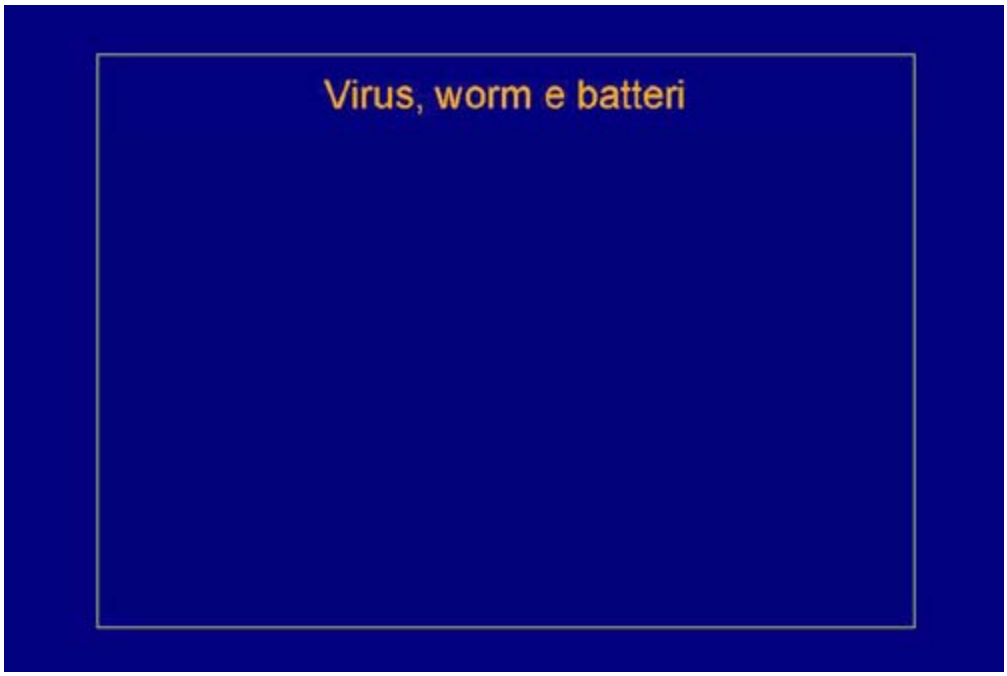
```
10 www.cs.interbusiness.it (151.99.250.4) 42.342 ms 34.359 ms 64.41 ms
```

Poiché in Internet il *routing* viene eseguito in modo dinamico, non è detto che ogni pacchetto segua necessariamente lo stesso percorso del precedente o di quello successivo (ad esempio perché una tratta diventa congestionata o interrotta). Pertanto *traceroute* di solito spedisce 3 pacchetti ICMP e calcola una media dei tempi impiegati.

Antivirus

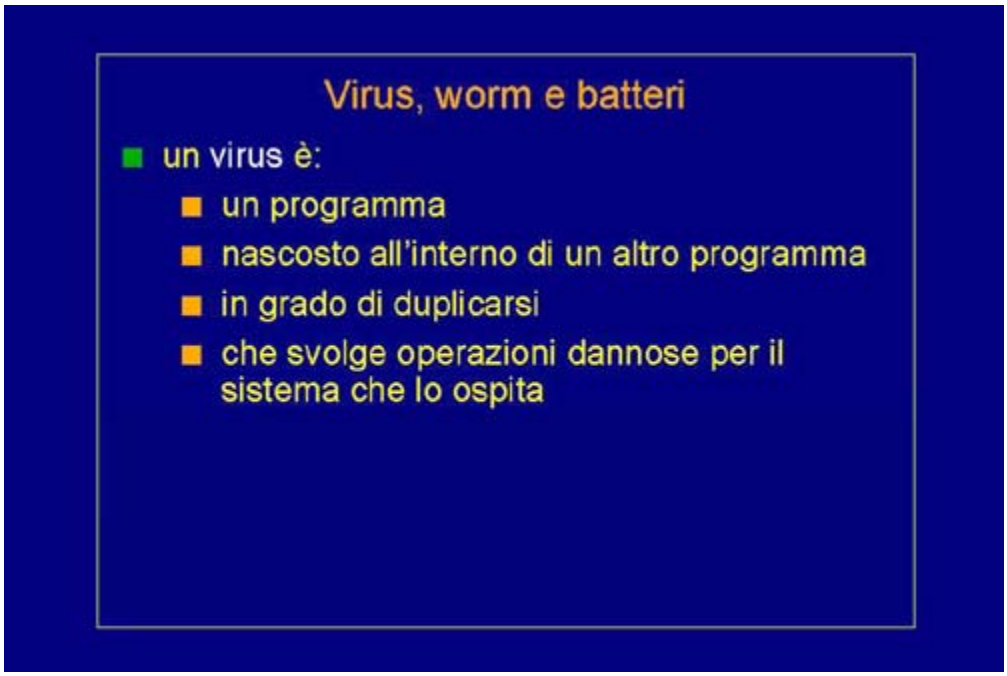
Franco Callegati
Paolo Presepi
Riccardo Gori
7.1.8 (Installare e configurare software antivirus)

Virus, worm e batteri



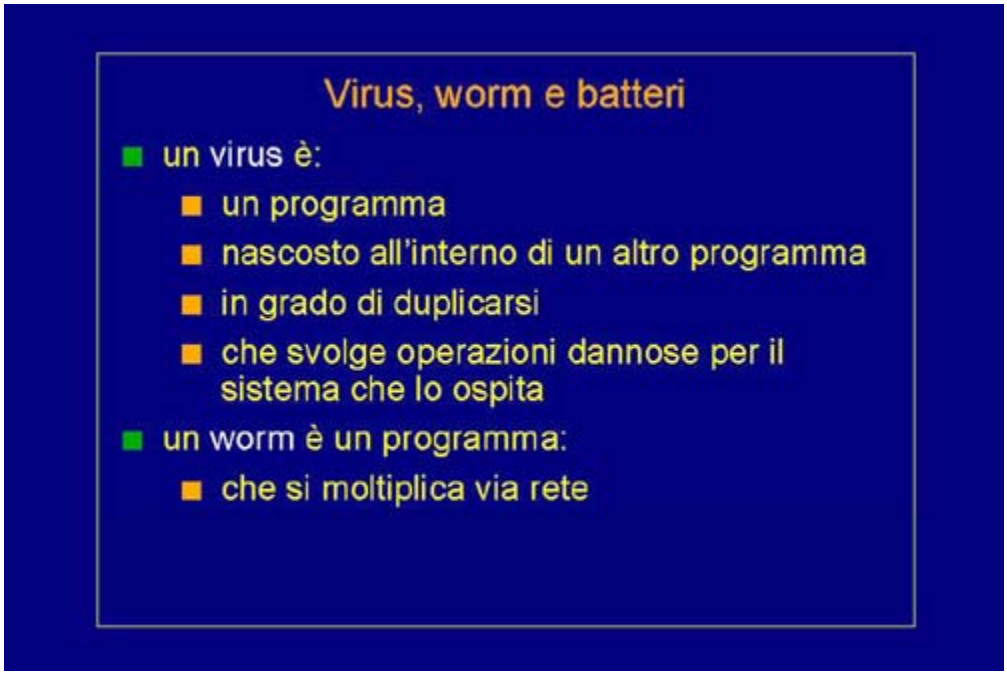
In questa lezione tratteremo della problematica di come proteggere i nostri sistemi informativi dai virus. I virus costituiscono oggigiorno una grossa minaccia perché continuano a diffondersi grazie alla complicità degli utenti, che si scambiano facilmente dati, sia tramite Internet sia tramite dischetti, senza controllare la loro reale provenienza. In particolare i virus sono soltanto uno degli elementi di un insieme di minacce che possono arrivarci: le altre minacce sono chiamate *worm* (vermi) o batteri. Vediamo quali sono le cose che contraddistinguono questi tre tipi di attacchi.

I virus



In particolare cominciamo dai virus. Il virus è un programma quindi è concettualmente del codice eseguibile che viene nascosto all'interno di un altro programma. Questo è il modo detto di infezione, ossia il virus si fa trasportare all'interno di un altro programma. In più, una delle caratteristiche di questo codice virale è quella di essere in grado di duplicarsi: ossia significa che il virus, nel momento in cui va in esecuzione, è in grado di vedere se esistono altri *file* che possono essere utilmente infettati. Inoltre, il virus è stato di solito sviluppato per compiere delle operazioni dannose nei confronti del sistema che lo ospita. Quindi il virus è un programma di attacco che viene inserito all'interno di altri programmi per compiere le azioni criminose e per moltiplicarsi e attaccare altri sistemi.

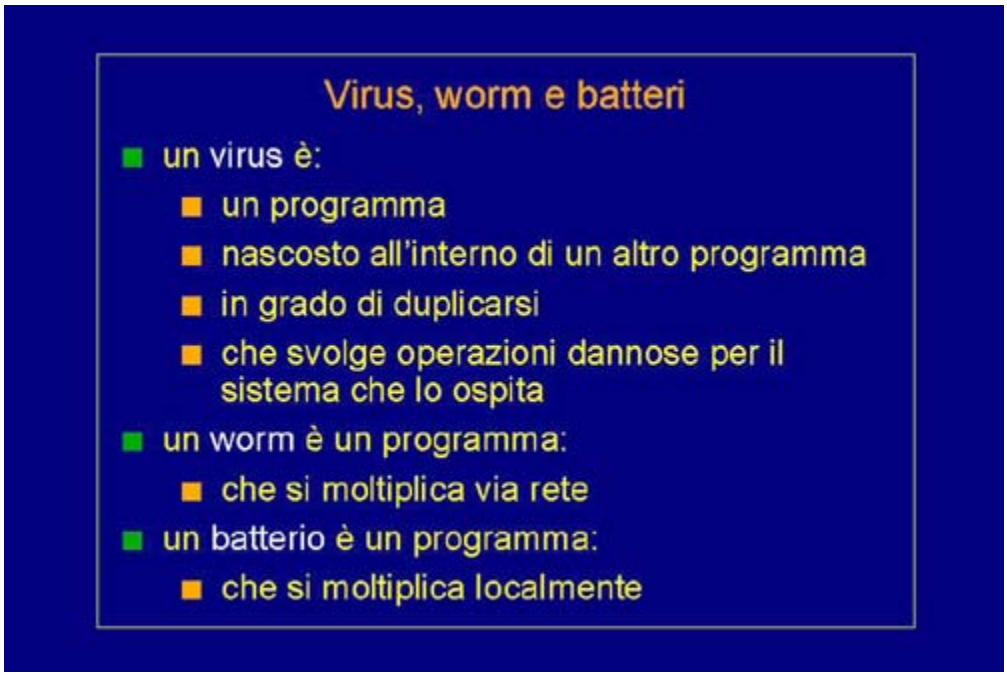
I worm



Un *worm*, invece, è un tipo di virus molto più semplice. Un *worm* non è nient'altro che un programma che attacca i

sistemi indirettamente, generando molte copie di se stesso, tramite la rete. Ossia, in generale, un *worm* tende ad avere una copia di se stesso in esecuzione su tutte le macchine collegate a una rete. È ovvio che deve trovare un meccanismo di trasmissione via rete. Quindi, i *worm*, non si possono propagare se i due calcolatori non sono collegati in rete e non si possono propagare se i due calcolatori sono adeguatamente protetti contro le intrusioni via rete.

I batteri

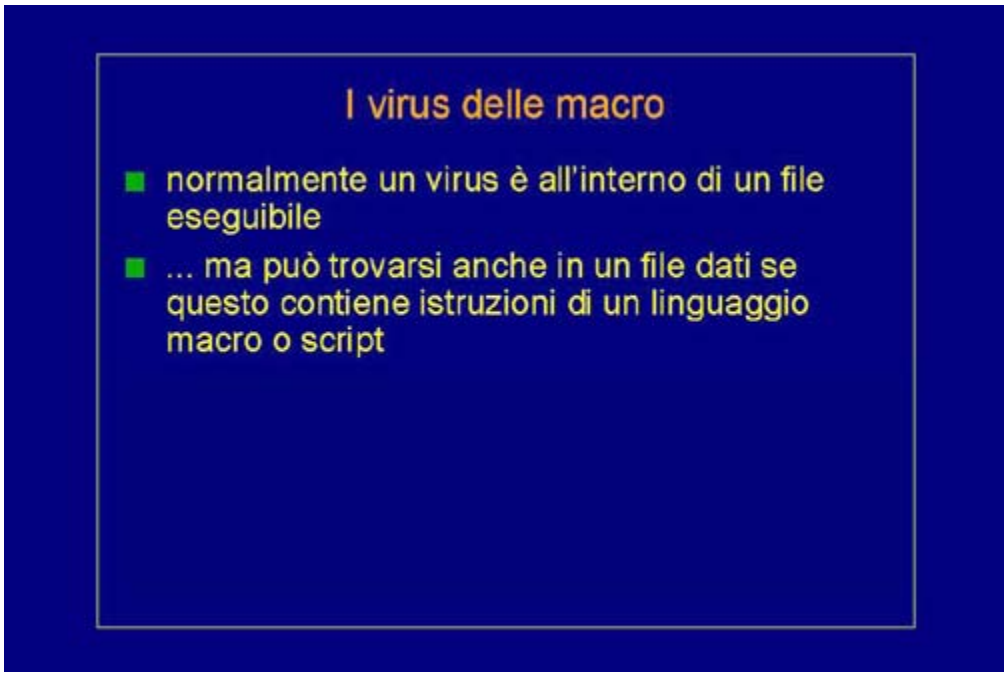


Si parla, infine, di batteri nel caso di programmi che tendono ad attaccare il sistema semplicemente generando infinite copie di se stessi, saturando quindi le risorse di calcolo. Un batterio è l'analogo di un *worm* però, al posto di propagarsi via rete, si limita ad effettuare le proprie copie all'interno del sistema stesso, fino a saturarne completamente la memoria, il tempo di calcolo o i dischi. In questa lezione noi ci concentreremo in particolare sui virus, perché oggigiorno costituiscono la maggior minaccia.

Anatomia di un virus

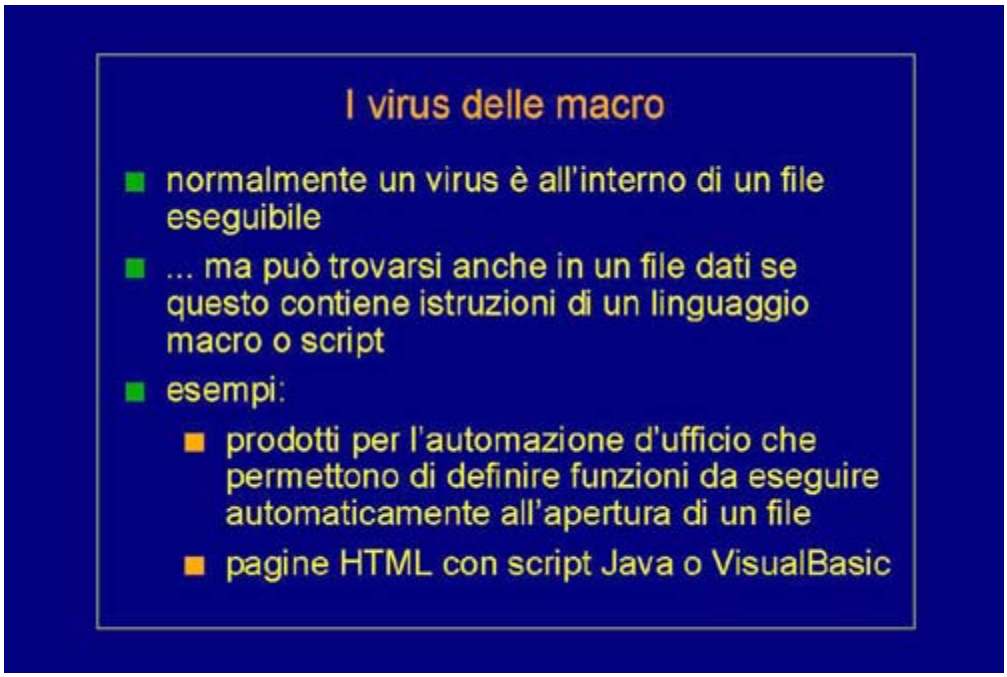
Vediamo com'è fatto: il virus è costituito da una parte di codice che serve ad inizializzare le proprie funzioni. Questa parte di codice cede ben presto il posto alla funzione di propagazione. Questo significa che il virus, durante la sua vita, controlla continuamente se esiste la possibilità di fare una copia di se stesso all'interno di altri *file* e all'interno di altri dischi, in modo da assicurarsi la propria moltiplicazione e la propria sopravvivenza. Inoltre, all'interno del virus è presente una sezione di codice che è il cosiddetto *trigger*, o grilletto, ossia una parte di codice che controlla se si è verificato un certo evento. Ad esempio, ci sono dei virus burloni che controllano se la data del sistema coincide con il primo di aprile: se è il primo di aprile allora compiono una qualche azione, che si spera non troppo dannosa vista la particolare data scelta. In altri casi potrebbe trattarsi di operazioni molto distruttive. In ogni caso, la parte di *trigger* è quella che scatena il comportamento cattivo del nostro virus, quindi il virus deve contenere un'ultima sezione che è quella relativa all'azione criminosa che intende intraprendere. Quindi, tutte e quattro queste parti sono quelle che noi troviamo all'interno di quasi ogni tipo di virus e corrispondono a concetti diversi, tutti egualmente necessari alla funzionalità del nostro virus.

I virus nelle macro



In particolare abbiamo detto che, in generale, un virus è situato all'interno di un *file* eseguibile. Che cos'è un *file* eseguibile? Un *file* eseguibile può essere di diverso tipo. In generale, non è detto che un *file* eseguibile sia un vero e proprio *file* binario, perché oggigiorno si sta diffondendo sempre di più l'abitudine, per chi sviluppa applicazioni, di nascondere pezzi di programmi, quindi pezzi di applicazioni, anche all'interno di *file* dati. In particolare notiamo che esistono dei linguaggi macro, o dei linguaggi di *script*, che vengono allegati all'interno di *file* dati.

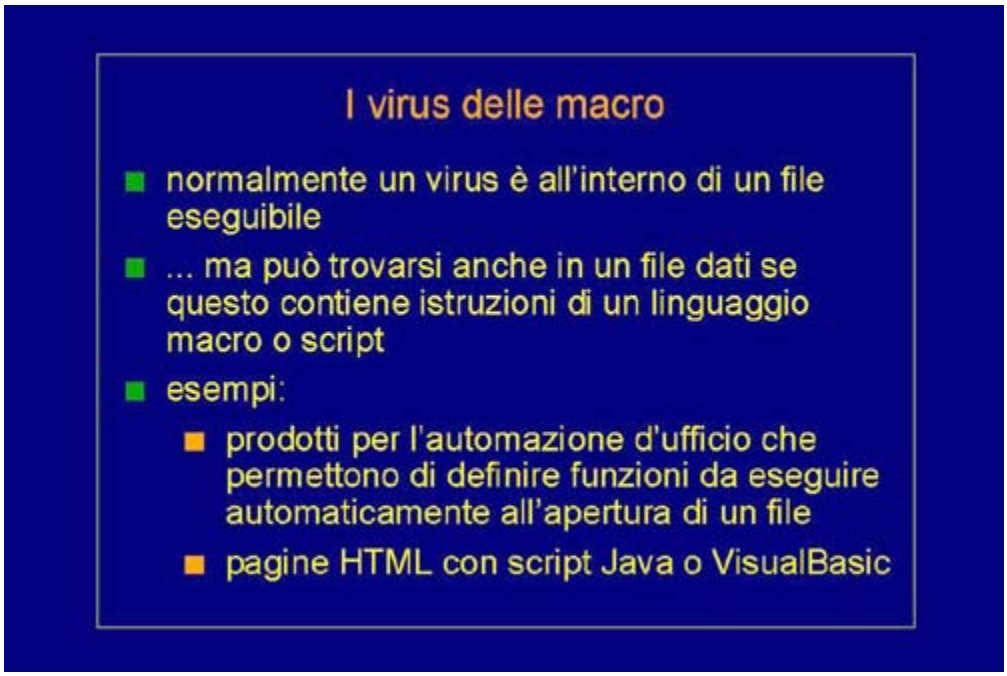
I virus nelle macro - esempi



Ad esempio, questo è tipico nei prodotti per l'automazione d'ufficio. Ci sono molti *word processor*, *spread sheet* e sistemi di presentazione grafica che permettono di definire funzioni da eseguire automaticamente all'apertura del *file*. Allora, i cosiddetti macrovirus sono quelli che si annidano non all'interno di un *file* eseguibile binario, ma all'interno

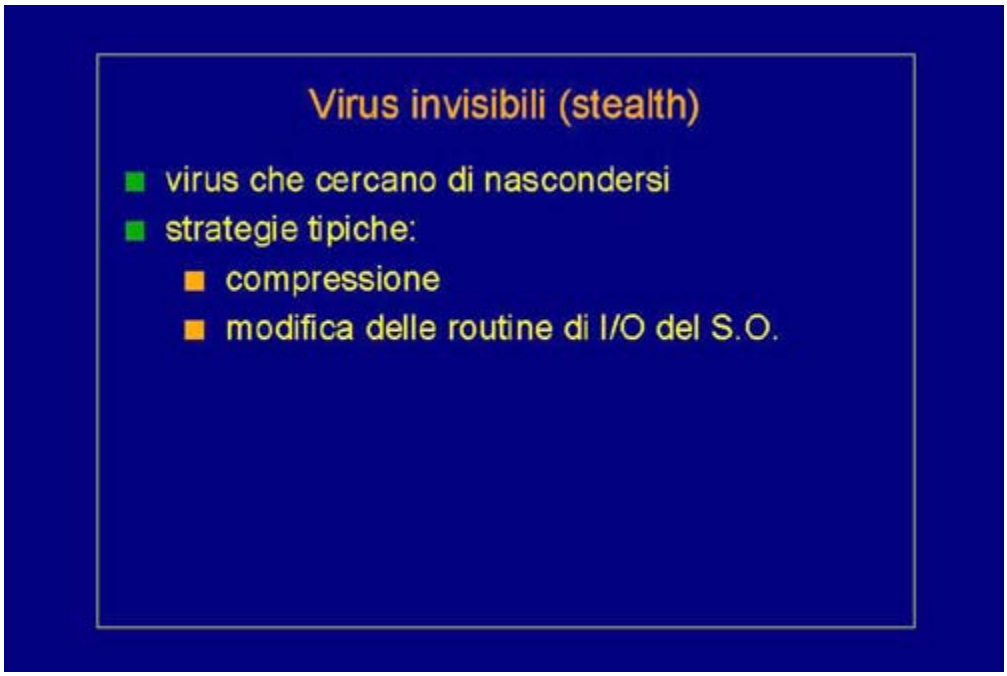
della macro, quindi di quel pezzo di codice scritto con un opportuno linguaggio di *script* e che servirebbe teoricamente a completare le funzionalità di uno di questi prodotti per l'automazione d'ufficio.

I virus nelle macro - esempi - HTML



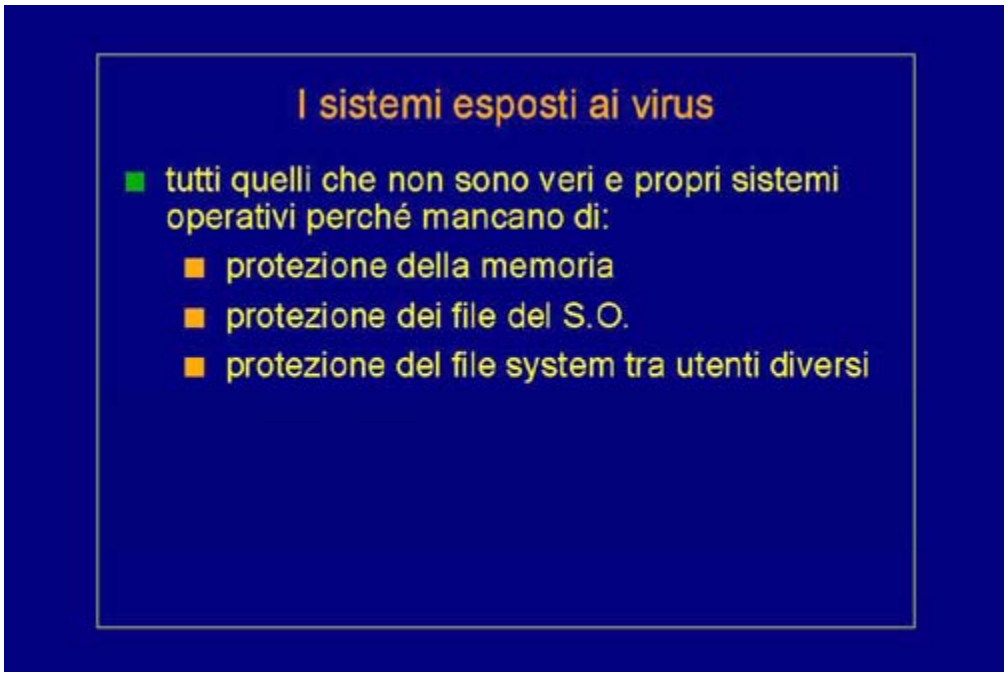
In modo analogo cominciano a diffondersi virus anche attraverso le pagine HTML. Di per sé una pagina HTML non dovrebbe essere nient'altro che una pagina che presenta del testo, della grafica, eventualmente dei filmati o dei suoni. Ma, recentemente, è stato possibile inserire all'interno delle pagine HTML anche dei brevi *script*, quindi diciamo delle funzioni, scritti con degli appositi linguaggi, ad esempio: *Java Script* e *Visual Basic Script*. Allora, a questo punto, diventa pericoloso anche semplicemente navigare via *Web*. Perché, navigando via *Web*, posso visualizzare una pagina HTML che contiene questo codice eseguibile e, nel momento in cui la visualizzo, questo codice può essere eseguito automaticamente sul mio calcolatore. Se il codice allegato all'interno della pagina HTML è un codice cattivo, è un codice virale, ecco che questo può condurre degli attacchi. Quindi sarebbe consigliabile, per chi scrive le pagine HTML, cercare di limitare o meglio ancora non utilizzare queste capacità di *scripting*, per chi naviga via *Web* sarebbe fortemente consigliabile, soprattutto quando visita pagine di siti non fidati, di disabilitare l'esecuzione degli *script* annidati all'interno delle pagine HTML.

Virus invisibili



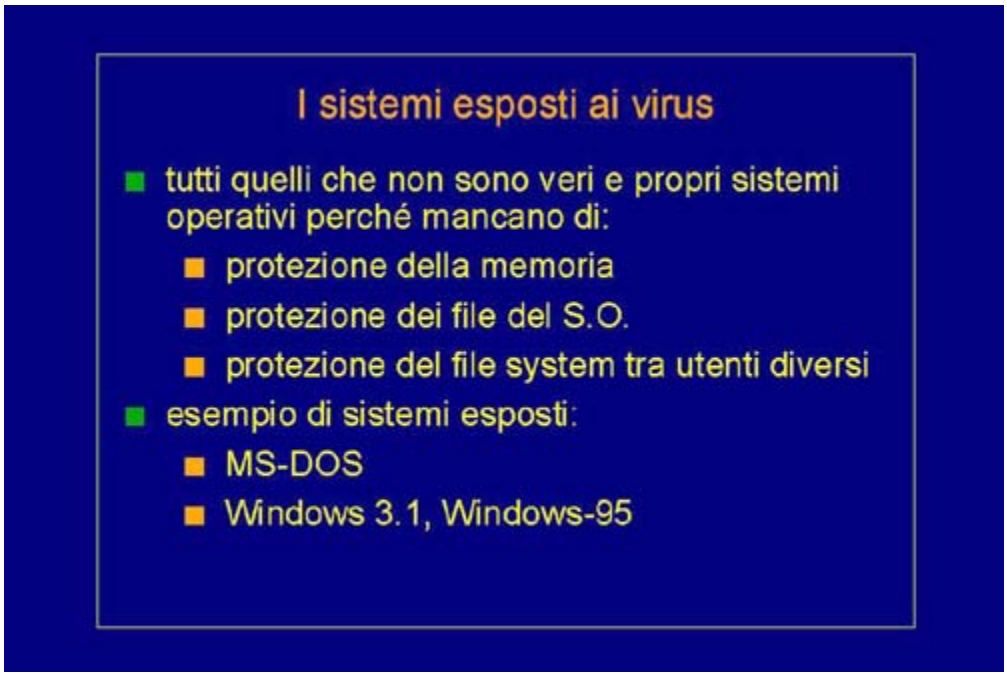
Esiste una continua battaglia tra chi sviluppa il *software* antivirus e gli *hacker* che scrivono il codice dei virus. In particolare, chi scrive il codice dei virus cerca di non rivelare la propria presenza, cerca di far sì che il virus possa passare inosservato. Si parla quindi di virus invisibili, o col termine inglese *stealth virus*, per tutti quanti quei casi in cui i virus cercano di nascondere la propria presenza all'interno del sistema. Esistono due strategie tipiche con cui gli scrittori di virus tendono a nascondere il proprio codice. La prima strategia tipica è quella di comprimere il codice. Il codice compresso non assomiglia più in nessun modo alle tipiche istruzioni *assembler* di un calcolatore, assomiglia piuttosto a dei normalissimi dati e quindi questo potrebbe trarre in inganno i *software* antivirus. È ovvio che il virus, prima di entrare in esecuzione, dovrà decomprimere le proprie istruzioni, proprio perché altrimenti queste non potrebbero essere mandate in esecuzione. Una seconda possibilità che alcuni virus mettono in atto, è quella di andare a modificare le *routine* di *input/output* del sistema operativo al cui interno loro operano. In questo modo, quando il *software* antivirus cerca di leggere i dati relativi al virus, loro al posto di fornire i veri e propri dati e quindi farsi rivelare, forniscono dei dati fasulli che non permettono la rivelazione. È chiaro che un sistema in cui sono stati manipolati addirittura le *routine* di *input/output* del sistema operativo è un sistema fortemente compromesso.

Virus polimorfici e cifrati



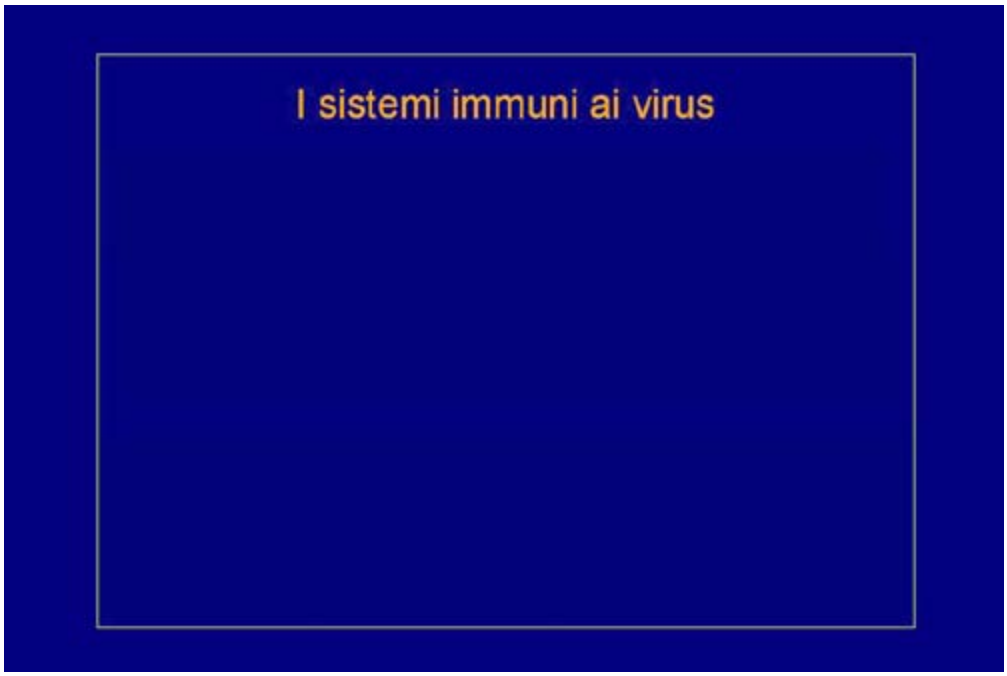
Altre due categorie di virus molto aggressivi e molto diffusi sono i virus polimorfici e cifrati. Sono quei virus che per evitare di essere rivelati cambiano il proprio codice ad ogni infezione. Questi sono i cosiddetti virus polimorfici: ossia, poiché la medesima operazione può essere in genere compiuta in diverse sequenze di istruzioni *assembler*, ci potrebbero essere dei virus che cambiano il modo con cui svolgono le proprie azioni ogni volta che si duplicano. Questo fa si che i virus assumano ogni volta una forma diversa e quindi sono più difficili da rivelare da parte dei *software* antivirus. Ancora meglio sono i virus che cifrano una parte del proprio codice tramite un algoritmo di crittografia. Questi sono i cosiddetti virus cifrati ed ovviamente sono ancora più difficili dei precedenti da visualizzare. In un certo senso sono simili a quelli che comprimono una parte del proprio codice, con l'aggiunta che, mentre noi potremmo provare a decomprimere i virus compressi, nel caso di virus cifrati è praticamente impossibile decifrare il virus se non si conosce la chiave di crittografia che è stata utilizzata. Quindi i virus cifrati sono ancora più difficili da rivelare, sia dei virus compressi, sia dei virus polimorfici.

I sistemi esposti ai virus



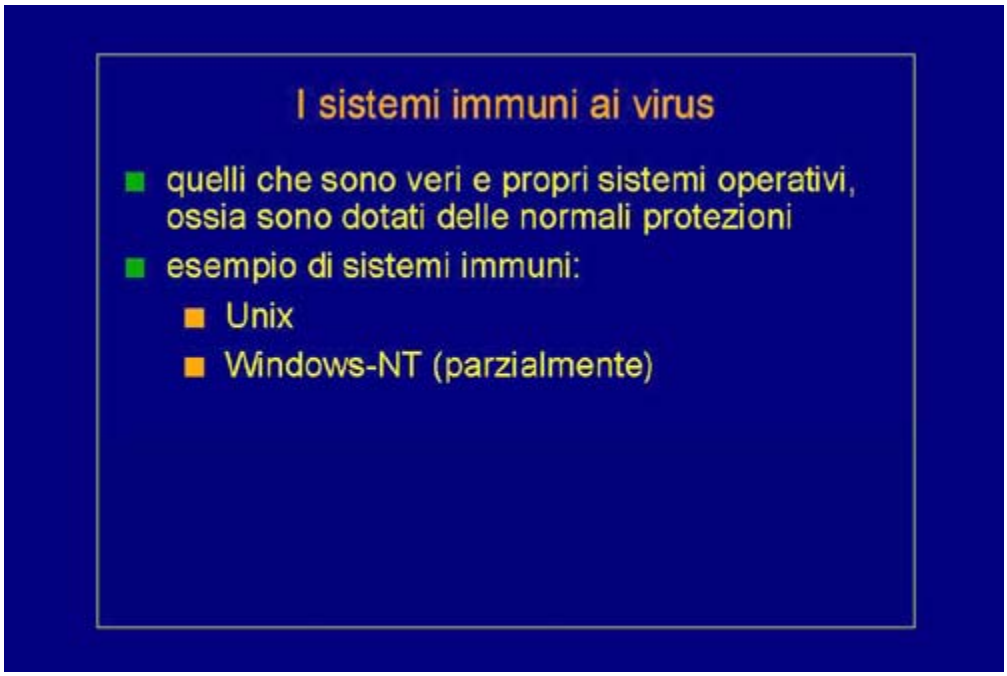
In generale, quali sono i sistemi esposti ai virus? Bisogna che qualunque persona in possesso di un calcolatore elettronico sia preoccupato dell'esistenza dei virus? La risposta è: no. Esistono certe categorie di *computer* che sono soggette ai virus e certe categorie di *computer* che sono invece immuni ai virus. In particolare, i sistemi esposti ai virus sono tutti quelli che non sono dotati di un vero e proprio sistema operativo. Ovverosia, sono dotati di pseudo-sistemi operativi che mancano delle tipiche caratteristiche di protezione. Ossia tutti quei sistemi che non proteggono la memoria fisica presente all'interno del sistema, ma permettono ad un qualunque programma di scrivere in una qualunque locazione di memoria. Sono quei sistemi che non impediscono ai programmi di modificare i *file* stessi del sistema operativo. Sono tutti quei sistemi che non hanno un *file system* che protegge i dati di utenti diversi e quindi permette ad un programma di manipolare i dati anche di un altro utente.

I sistemi esposti ai virus - esempi



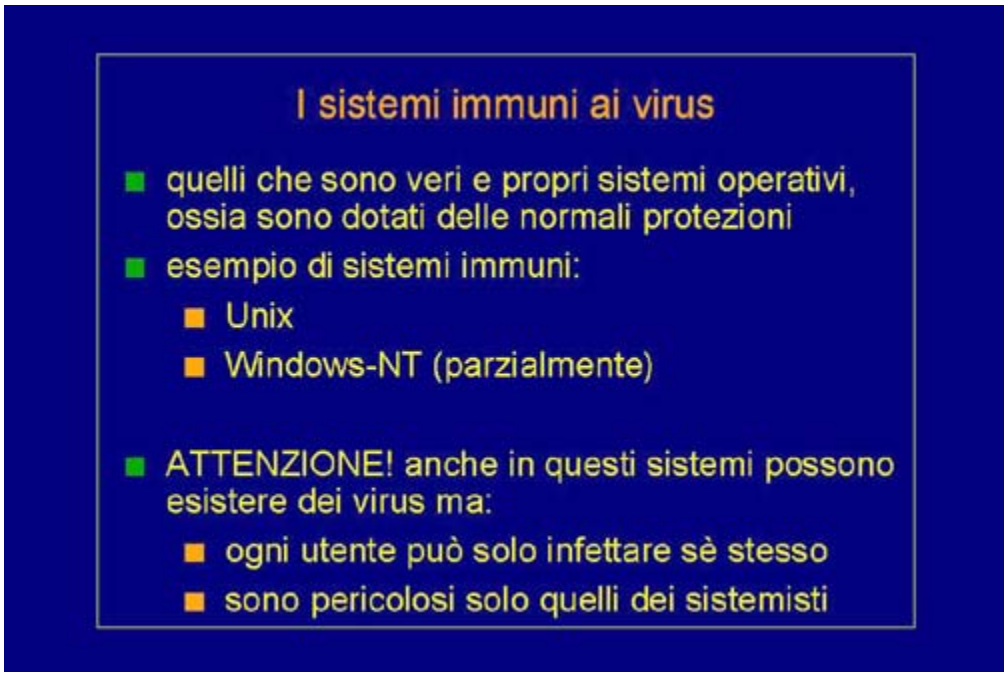
Esempi sono i primi sistemi di uso personale: il sistema operativo MS-DOS, il sistema operativo *Windows* 3.1 e *Windows* 95. Sono sistemi operativi particolarmente semplici da utilizzare, ma questa loro semplicità è stata ottenuta a scapito della protezione e del grado di sicurezza. E quindi questi, in generale, sono i sistemi operativi più esposti agli attacchi dei virus, proprio perché permettono al virus di andare a scrivere in qualunque locazione di memoria, su qualunque *file*, di qualunque utente e addirittura di andare a modificare i dati del sistema operativo stesso. Questi sono i tipici ambienti in cui i virus nascono, si sviluppano, crescono e si moltiplicano.

Sistemi immuni ai virus 1/2



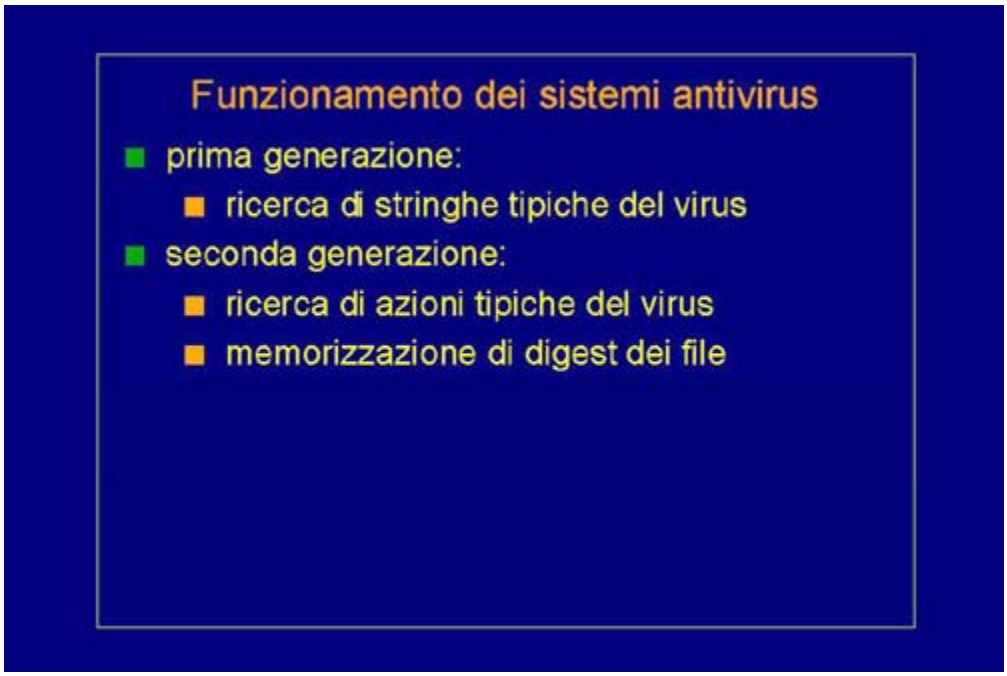
Quali sono invece i sistemi immuni ai virus? Sono ovviamente l'insieme complementare. Sono tutti quanti quei sistemi dotati di un vero e proprio sistema operativo, ossia dotati di tutte le normali protezioni che abbiamo visto mancare invece nella prima categoria di sistemi. Esempi di sistemi immuni, allora, sono *Unix* e *Windows* NT, ossia tutti quei sistemi che implementano delle forti politiche di protezione. Nel caso di *Windows* NT bisogna fare attenzione, perché al suo interno esiste un sottoinsieme di esecuzione che fornisce compatibilità con i *Windows* di tipo precedente. Come tale, questo ambiente, offrendo la compatibilità con le applicazioni *Windows* di tipo a 16 bit o a 32 bit, offre anche la compatibilità con i virus sviluppati per DOS, per *Windows* a 16 bit e per *Windows* a 32 bit. Quindi, *Windows* NT è insensibile ai virus, ma sono ed esclusivamente per la parte che riguarda le applicazione native NT e non per le applicazioni invece in modalità di compatibilità Dos o *Windows* di tipo più vecchio.

Sistemi immuni ai virus 2/2



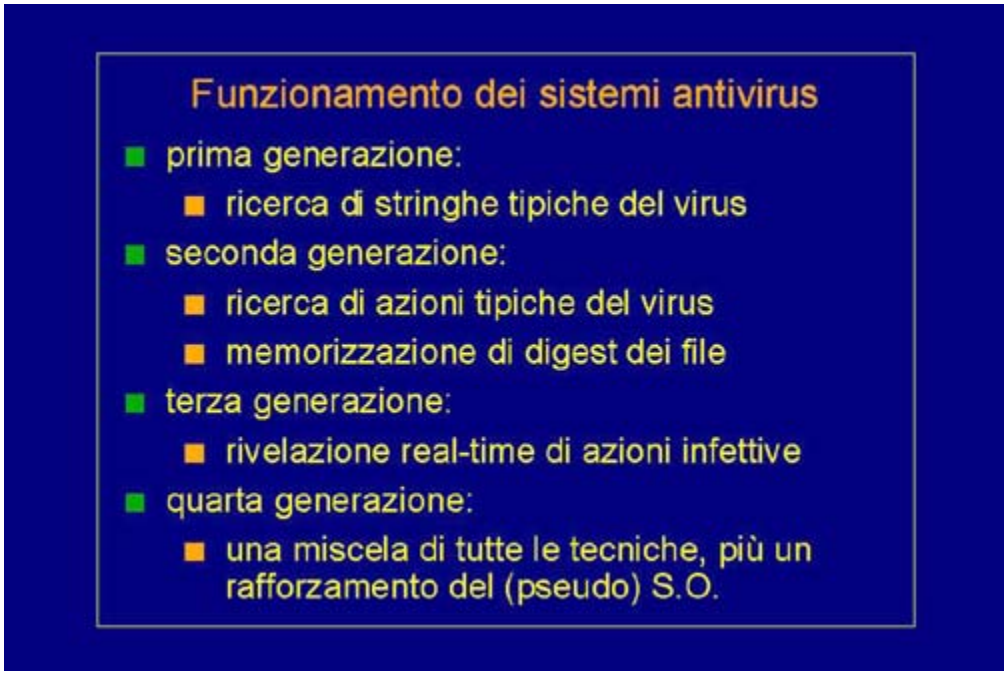
Bisogna però fare attenzione ad una cosa. Questi sistemi offrono protezione nei confronti dei virus che cercano di alterare i dati di altri utenti o i dati del sistema. Questo non significa che i virus non possano esistere mai all'interno di questi sistemi, ma significa piuttosto che un singolo utente può introdurre un virus nel sistema ma, questo virus, sarà in grado di svilupparsi, moltiplicarsi e colpire solo ed esclusivamente l'utente stesso. Il virus non potrà propagarsi, proprio per i vincoli che il sistema operativo pone. Quindi, in particolare, bisogna fare molta attenzione a quali sono i *file* introdotti dai sistemisti, perché i sistemisti hanno il permesso di andare a modificare i dati di qualunque utente e anche i dati del sistema operativo stesso. Ovviamente, se un sistemista introducesse per sbaglio un virus all'interno del sistema, allora questi sarebbero dei problemi molto grossi, perché questo potrebbe infettare tutti i dati di tutti quanti gli utenti.

Funzionamento dei sistemi antivirus 1/2



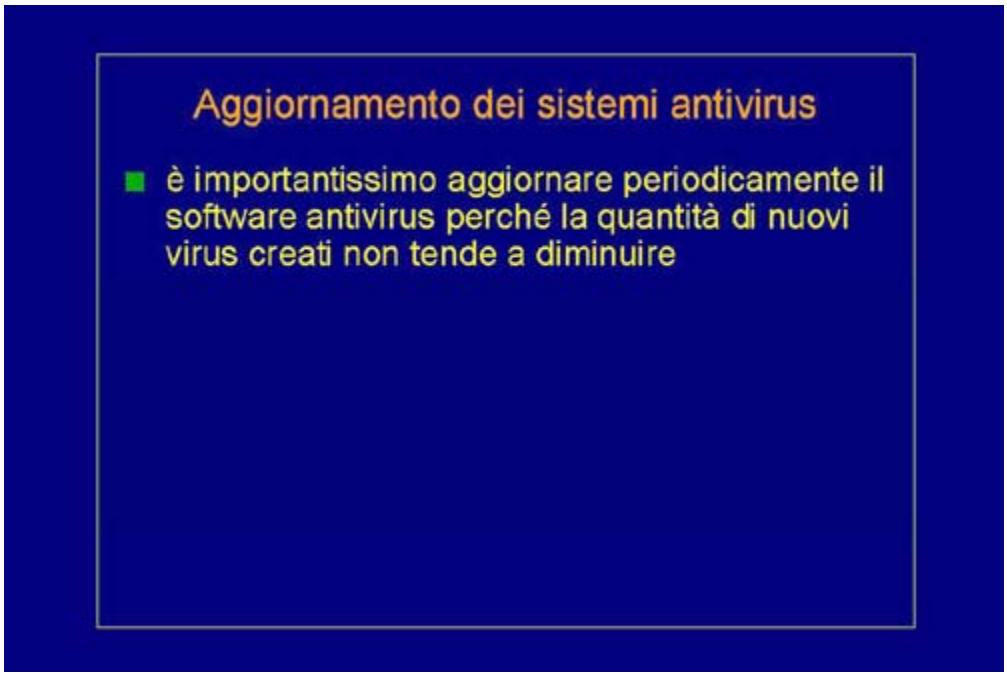
Come funziona, in generale, un sistema antivirus? Esistono varie strategie, alcune più efficaci, alcune meno efficaci; in particolare le strategie sono state affinate nel corso degli anni per rispondere ai nuovi tipi di virus che venivano sviluppati. I sistemi antivirus di prima generazione erano dei sistemi monto semplici, che non facevano nient'altro che ricercare quelle sequenze di istruzioni tipiche di un certo virus. Per ogni virus, veniva definita quella che è la sua *signature*: ossia le istruzioni tipiche che si trovano solo in quel virus. Il nostro *software* antivirus non faceva nient'altro che cercare quella sequenza di istruzioni all'interno di tutti i *file* presenti nel sistema. Gli antivirus di seconda generazione, invece, sono diventati un po' più sofisticati e, oltre a fare la ricerca delle stringhe tipiche dei virus, vanno anche a considerare il codice *assembler* generico, andando a vedere quali sono le tipiche modalità di operazione dei virus. Ad esempio: quando trovano all'interno di un *file* delle istruzioni che cercano di copiare un pezzo di questo *file* all'interno di un altro *file*, questo potrebbe essere un indizio della presenza di un virus. Oppure, ci sono *software* antivirus che calcolano il *digest* (il numero che permette di controllare l'integrità dei *file*) e vanno a verificare se il *file* è stato modificato dopo che questo *digest* è stato calcolato. Questo può essere indicazione di un'infezione virale in corso.

Funzionamento dei sistemi antivirus 2/2



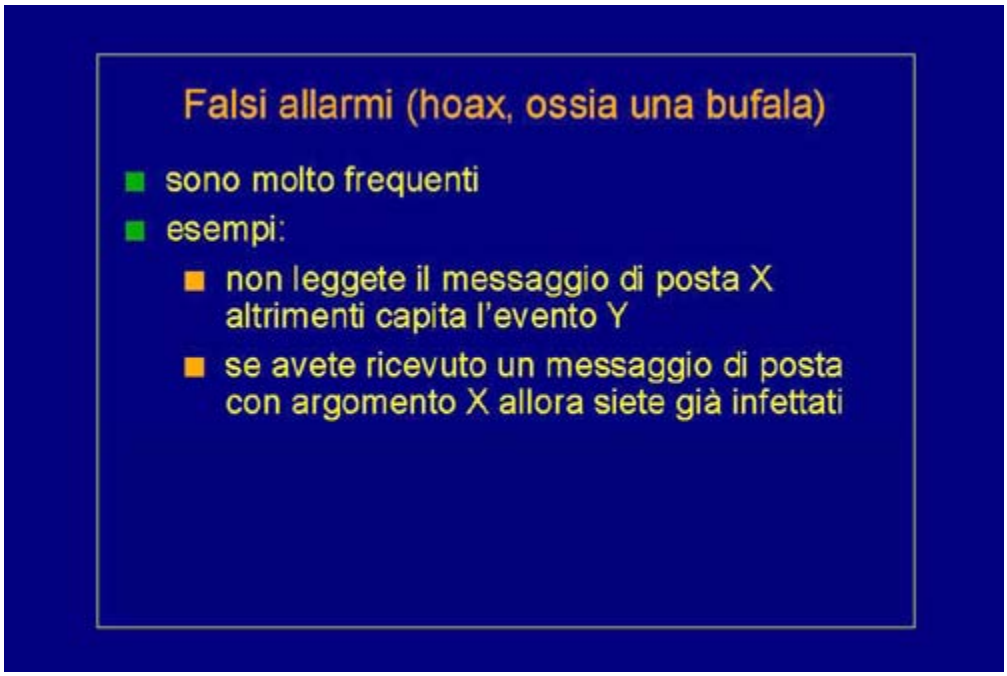
I sistemi della terza generazione sono, invece, dei sistemi attivi *on-line*. Significa che esiste un programma attivo in continuazione sul nostro sistema, il quale controlla le azioni fatte da tutte le applicazioni e cerca di capire se le azioni che vengono svolte sono operazioni normali (lecite) di una applicazione, o sono invece azioni sintomo di un virus. Nel qual caso tende a bloccarle e a segnalare la presenza del virus. I sistemi più recenti, quelli cosiddetti della quarta generazione, ormai mischiamo insieme tutte quante queste tecniche. Anzi, fanno anche di più: molto spesso questi sistemi sono messi all'interno di un insieme di prodotti, diciamo di una *suite* di prodotti, che offre anche degli strumenti per elevare il livello di sicurezza, quindi anche il livello di protezione, di quei sistemi che abbiamo visto essere eccessivamente insicuri e quindi facilmente esposti ai virus.

Aggiornamento dei sistemi antivirus



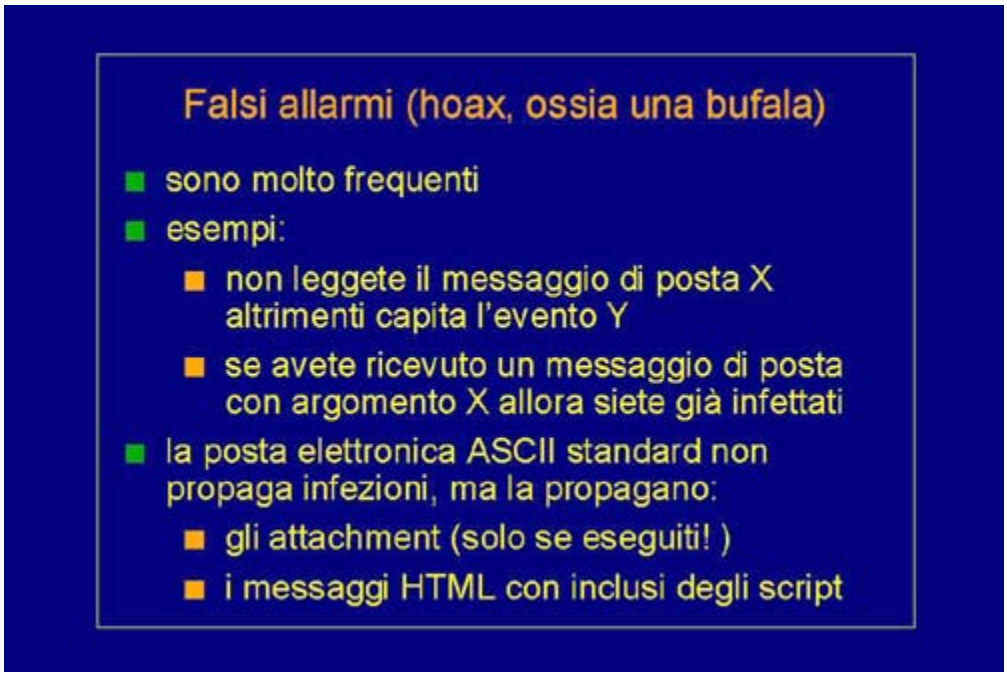
Bisogna ricordare che installare il *software* antivirus su un *computer* è una operazione necessaria, ma non sufficiente a garantirci la protezione da tutti i virus. Bisogna ricordarsi che, in generale, vengono sviluppati in continuazione decine e decine di virus. La stima è che ogni mese vengano introdotti in circolo alcune centinaia di virus di tipo nuovo. Questo significa che se io ho installato il *software* antivirus in una carta data, dopo sei mesi il mio sistema è aperto all'attacco di circa seicento nuovi virus, che non possono essere rivelati dal *software* esistente. È quindi importantissimo aggiornare periodicamente il *software* antivirus su tutte le macchine. Questo può significare due cose: la prima è aggiornare i *file* di informazioni, ad esempio quelli che contengono le stringhe tipiche dei virus. Ma questo può non bastare: a volte è necessario anche aggiornare il motore, ossia il programma che effettua la ricerca dei virus, proprio perché nascono in continuazione nuovi tipi di virus e nuove modalità di infezione. Quindi, anche avendo fornito nuovi dati, se l'applicazione antivirus non è stata pensata per rivelare un certo tipo di infezione, non sarà in grado, anche con nuovi dati, di rivelare i nuovi tipi di virus. Quindi: aggiornamento costante e continuo, sia dei dati che rappresentano i virus, sia del motore di ricerca dei virus.

Falsi allarmi (hoax) 1/2



Indirettamente collegata al virus c'è una tematica, che sarebbe divertente se non fosse in certi casi tragica. Molto spesso vengono diffusi via rete dei falsi allarmi: i cosiddetti *hoax*, o come diremmo noi italiani, delle bufale. Ossia delle cose che non stanno né in cielo né in terra, ma siccome gli utenti dei sistemi informativi non sono necessariamente degli esperti di informatica, riescono ad avere presa su molti utenti e a generare molto spesso delle azioni non appropriate. In particolare, questi falsi allarmi si stanno diffondendo con molta frequenza. Esempi: a me personalmente, e credo anche a molti di coloro che ascoltano questa lezione, è capitato di ricevere dei messaggi di posta in cui si dice: attenzione, non leggete il messaggio di posta X, perché altrimenti vi viene cancellato direttamente tutto quanto il disco. Questo è chiaramente un *hoax*, come vedremo tra poco. Oppure esistono degli altri avvisi che circolano in rete, che dicono: attenzione, se avete ricevuto un messaggio di posta con questo argomento, anche se non lo avete letto, siete già automaticamente infettati. Diciamo che, nell'immaginario collettivo, la posta elettronica diventa uno dei metodi di propagazione di virus più frequenti.

Falsi allarmi (hoax) 2/2



Allora voglio sottolineare che, in generale, la posta elettronica di tipo ASCII, quindi quella che contiene sono ed esclusivamente del testo, non può essere veicolo di propagazione di virus. La posta elettronica può propagare dei virus solo se contiene degli *attachment*, ossia dei pezzi aggiuntivi uniti al testo della posta elettronica, e se questi *attachment* sono di tipo eseguibile e vengono eseguiti. Quindi, è molto importante che sul proprio sistema, il programma di posta elettronica sia configurato in modo tale da non eseguire automaticamente gli *attachment* ricevuti. Prima di eseguire un *attachment* bisognerebbe sempre farlo passare al controllo di un *software* antivirus. Esistono anche degli antivirus che sono in grado di fare automaticamente il controllo degli *attachment* ancor prima che vengano letti. L'unica possibilità per cui un virus si annidi all'interno del testo di posta elettronica, è se il testo non è stato scritto in ASCII, ma in HTML. Questa è una caratteristica che gli *e-mailer* più recenti hanno: permettono la spedizione di messaggi di posta elettronica scritti in HTML. Questo permette di formattare, di abbellire, il nostro messaggio. Ma poiché l'HTML può contenere, al proprio interno, degli *script Java* o degli *script Visual Basic*, come abbiamo già visto poco fa, ecco allora che anche tramite un messaggio di posta elettronica ci può essere inviato un cosiddetto macrovirus (uno *script* virus) che ci infetta. Quindi sarebbe bene, anche in questo caso, disabilitare l'esecuzione di questi *script*, se presenti all'interno del testo del messaggio di posta elettronica che ci viene inviato.

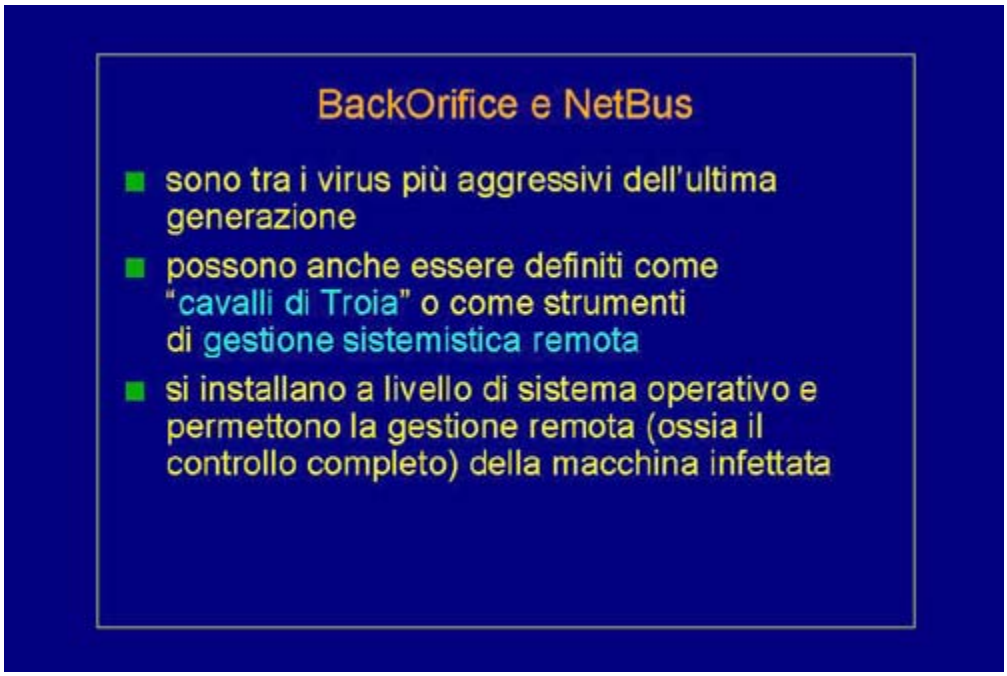
La firma degli applicativi 1/2

Esiste la possibilità di porre fine a questa infinita saga dei virus? Teoricamente la possibilità esiste e si basa sul concetto di firma digitale, che abbiamo già affrontato nell'ambito di questo corso, ma che vogliamo qui ribadire. La firma digitale di un insieme di dati ci permette di garantirne due cose: l'integrità e l'autenticazione. L'integrità ci permette di dire che i dati non sono stati modificati, da quando sono stati creati. L'autenticazione ci permette di identificare in modo univoco chi ha creato questi dati. È ovvio che se noi abbiamo un *file*, che è stato infettato da un virus, questo *file* non è più integro. Quindi se noi abbiamo calcolato la sua firma digitale prima dell'infezione e la ricalcoliamo dopo l'infezione, la firma digitale non sarà più la stessa. Quindi, la semplice verifica della firma digitale di un *file* ci permetterà di scoprire subito che è stato infettato e quindi di ripristinare il *file* originario, o almeno di cancellare il *file* infettato.

La firma degli applicativi 2/2

Questo significa che noi potremmo liberarci completamente dei virus, se all'interno dei nostri calcolatori tutto il codice eseguibile, sia quello direttamente in forma binaria (i programmi, le applicazioni), sia quello annidato dentro ai dati (gli *script*), fosse firmato digitalmente. La firma digitale è un elemento necessario ma non sufficiente. Se anche noi avessimo tutti i dati firmati, questo non ci basterebbe a togliere i virus: bisogna anche che il sistema operativo che esegue il programma binario, o l'applicazione che utilizza i dati che a loro volta contengono lo *script*, prima di eseguire il *file* binario e prima di eseguire lo *script*, ne controllasse la firma. Se la firma indica che i dati sono stati manipolati, allora il sistema operativo e l'applicazione dovrebbero rifiutarsi di eseguirlo. Questa è una pia speranza? No, è una realtà quasi pronta per essere rilasciata. Cominciano ad esserci tracce di importanti ditte di *software* che stanno lavorando in questa direzione, per fornire sistemi operativi in grado di apporre e di verificare la firma digitale a tutti quanti gli eseguibili. Analogamente, ci sono ditte che stanno sviluppando applicazioni che controllano che gli *script* non siano stati modificati; anche in questo caso l'integrità degli *script* viene verificato tramite la loro firma digitale.

BackOrifice e NetBus



Concludiamo questa chiacchierata sui virus parlando di due virus che ultimamente hanno ottenuto una grande notorietà: si tratta di *BackOrifice* e *NetBus*. Questi sono due virus, particolarmente aggressivi, che sono stati sviluppati per gli ambienti di tipo *Windows*. Sono tra i virus più aggressivi tra quelli dell'ultima generazione e possono anche essere classificati come cavalli di Troia, o come strumenti di gestione sistemistica. Cavallo di Troia è normalmente un virus che si inserisce all'interno di un sistema, non per danneggiarlo direttamente, ma per aprirne le porte ai nemici che stanno fuori. Esattamente come il cavallo di Troia non serviva direttamente a danneggiare la città, ma serviva semplicemente ad aprire le porte della città agli attaccanti che stavano all'esterno. Di che cosa si tratta? Entrambi i sistemi, una volta installati su un sistema *Windows*, permettono la gestione remota di tutte le funzionalità della macchina. Allora, in senso benigno, possono essere visti come dei programmi di ausilio a un gestore di sistema che desidera gestire remotamente alcune macchine. Se però il *BackOrifice* o il *NetBus* è stato installato all'insaputa del proprietario della macchina, questo significa avere una porta aperta in cui una qualunque persona di passaggio sulla rete è in grado di controllare completamente la funzionalità del nostro sistema. E questa, ovviamente, è una cosa estremamente negativa e quindi sicuramente equiparabile a un cavallo di Troia o a un virus.

Gestione della posta elettronica su Linux

Franco Callegati
Paolo Presepi
Riccardo Gori

7.3.1 (Installare e configurare applicazioni client/server su un server quali: e-mail, FTP, Web, sistemi di messaggistica, chat, eccetera)

Introduzione

Il sistema di posta elettronica di *UNIX* è formato dall'interazione di più componenti *software* che devono essere opportunamente configurati (sebbene le operazioni più complesse vengono svolte da un unico programma, il *Mail Transport Agent*). Nella trattazione faremo riferimento a ***Sendmail*** , il *Mail Transport Agent* più diffuso in ambiente *UNIX*. Data la complessità dell'argomento, si consiglia di fare riferimento al *Linux Documentation Project* (**www.tldp.org**) e ai testi di base sul sistema operativo *Linux*.

La gestione della posta elettronica in Unix

In questa unità didattica viene descritto il sistema di posta elettronica di *UNIX*. Il sistema deriva dall'interazione di più componenti *software* che devono essere opportunamente configurate (sebbene le operazioni più complesse vengono svolte da un unico programma, il *Mail Transport Agent*). Nella trattazione faremo riferimento a *Sendmail*, il *Mail Transport Agent* più diffuso in ambiente *UNIX*. Data la complessità dell'argomento, si consiglia di fare riferimento ai testi consigliati nella pagina di Risorse.

Generalità sulla posta elettronica in *Unix/Linux*

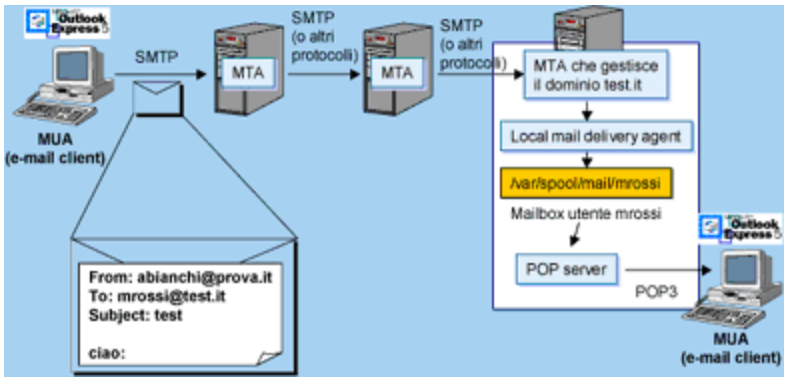
In questo capitolo viene descritto un **sistema di posta elettronica** , soffermandosi sulla funzione dei componenti principali. Viene inoltre descritto il funzionamento del **protocollo SMTP** e vengono introdotti i programmi disponibili in ambiente *UNIX* per la realizzazione di un servizio di posta elettronica.

Struttura e flussi in un sistema di e-mail

Componenti di un sistema di posta elettronica in *UNIX*

Un sistema di posta elettronica per funzionare prevede l'interazione di diversi componenti. I principali sono i seguenti:

- MUA (*Mail User Agent*). Il comune *client* di *e-mail*. È l'interfaccia fra l'utente ed il sistema di posta elettronica. I messaggi in arrivo vengono prelevati dalla *mailbox* dell'utente, residente su un *server*, mentre quelli in uscita vengono passati ad un MTA che si occupa di inoltrarli.
- MTA (*Mail Transport Agent*). Si tratta del componente più importante, il quale svolge diverse funzioni:
  - accetta i messaggi dal MUA o da altri MTA e si occupa di smistarli (*routing*);
  - consegna la posta in uscita verso altri MTA;
  - accetta da altri MTA i messaggi destinati agli indirizzi locali e ne gestisce la consegna nelle *mailbox* degli utenti (*local delivery*);
  - gestisce la posta in transito o in uscita attraverso delle code (*mail queue*).
- MDA (*Mail Delivery Agent*). La spedizione dei messaggi di posta non viene compiuta direttamente dall'MTA ma mediante appositi *Mail Delivery Agent* (o *mailer*), specializzati per i diversi protocolli di trasporto (SMTP, UUCP, consegna locale,...). Anche la consegna dei messaggi per gli utenti locali nelle *mailbox* di sistema avviene utilizzando un apposito *mail delivery agent* (*localmail*, *procmail*).



La figura illustra il tipico percorso di un messaggio di posta elettronica attraverso i componenti descritti.

- Un messaggio di *e-mail*, destinato all'utente mrossi@test.it, viene creato dall'utente abianchi@prova.it mediante un MUA, ad esempio *Outlook Express*, e spedito ad un MTA mediante il protocollo SMTP.
- Lo scambio della posta fra MTA avviene utilizzando un meccanismo di *store and forward* (memorizza e passa avanti), in cui ogni MTA riceve e smista verso un altro MTA i messaggi non destinati ad utenti locali. I messaggi vengono all'evenienza memorizzati per un certo periodo di tempo in una coda, in modo da non perderli qualora uno dei nodi sia temporaneamente inattivo.
- Dopo un certo numero di passaggi, il messaggio giunge all'MTA che gestisce la posta del dominio test.it, dove viene consegnato nella *mailbox* dell'utente locale mrossi e ivi conservato.
- L'utente mrossi preleva i messaggi dalla propria *mailbox* utilizzando un MTU, attraverso il protocollo POP3.

Scelta del MTA

Nel seguito di questa *Unit* faremo riferimento al *software Sendmail*, essendo quello più diffuso e disponibile come standard su tutte le versioni di *UNIX*.

La configurazione di *Sendmail* avviene principalmente mediante il *file* /etc/sendmail.cf, dal formato assai criptico. Nelle versioni recenti non è necessario modificare a mano questo *file*, in quanto esso viene creato a partire da un *file* dal formato più semplice utilizzando un processore di macro. Le difficoltà di configurazione sono tuttavia giustificate dalle ampie possibilità di personalizzazione e dalla ottima flessibilità offerte da *Sendmail*.

Sendmail, specialmente nelle vecchie versioni, è conosciuto come il responsabile di molti problemi di sicurezza. Questa fama dipende in parte anche dal fatto che si tratta di un *software* ad ampia diffusione, e perciò con una ampia base di utenza ed un elevato scambio di informazione e conoscenze. Tuttavia è innegabile che vi siano stati in passato alcuni problemi di sicurezza anche significativi, in parte dovuti alle configurazioni fornite di *default* con alcuni sistemi *UNIX*, in parte derivanti da oggettive motivazioni progettuali, come il fatto che si tratta di un programma eseguibile complesso e funzionante con i permessi di *root*.

Le versioni attuali di *Sendmail*, specialmente quelle fornite con *Linux*, offrono un ragionevole grado di sicurezza e contengono dei meccanismi utili ad evitare abusi (SPAM). Tuttavia, come per ogni altro programma che gestisce un servizio di rete, è buona norma seguire con attenzione i vari bollettini di sicurezza e tenere aggiornato il sistema secondo le indicazioni ivi contenute.

Coloro che desiderassero un MTA meno potente di *Sendmail* ma più semplice da gestire, possono prendere in considerazione una delle seguenti alternative:

- *Smail*. Si tratta di un MTA compatibile con *Sendmail* e con caratteristiche simili. È necessario fare attenzione ad utilizzare una versione recente, in quanto quelle antecedenti alla 3.1.29 soffrono di alcuni problemi di sicurezza. È disponibile con licenza GPL.
- ***Exim***. Concettualmente simile a *Sendmail/smail*. Viene fornito assieme ad una ottima documentazione, è efficiente ed offre un notevole controllo contro lo SPAM0.
- ***Qmail***. Si tratta di un *software* intrinsecamente più sicuro rispetto agli MTA tradizionali, essendo formato non da un unico programma eseguibile, bensì da componenti più piccoli e pertanto più facilmente controllabili.

- ***Postfix***. È un *software* piccolo, veloce, efficiente ed estremamente sicuro, che tuttavia non offre alcune delle funzionalità che invece sono disponibili utilizzando altri prodotti.

MUA

Mail User Agents

Per *UNIX* sono disponibili diversi *client* di posta elettronica, sia funzionanti in modalità testuale (*elm*, *pine*) che con interfaccia grafica. Esistono inoltre dei *software* che consentono la lettura e spedizione dei messaggi attraverso un *Web server* (*openwebmail*).

Il tipico *client* di posta elettronica di solito preleva i messaggi ricevuti da una *mailbox* residente su un *server* mediante il protocollo POP3 (o IMAP). Il MUA di solito non è in grado di occuparsi dell'instradamento dei messaggi, pertanto tutta la posta in uscita viene passata, utilizzando il protocollo SMTP, ad un MTA in grado di compiere l'operazione (*smarthost*).

Nel caso il *client* giri sulla stessa macchina *UNIX* dove risiede la casella dell'utente, questa può essere letta anche accedendo direttamente al *file* corrispondente nella *directory* di *spool* della posta. Nei *client* più semplici, che non supportano SMTP, la posta viene spedita all'MTA utilizzando un programma accessorio con funzione di *mail delivery agent* per il protocollo SMTP.

MIME

La gestione degli allegati nei messaggi di posta elettronica è competenza del MUA. Per i sistemi *UNIX* è disponibile il pacchetto *Metamail*, che contiene i programmi necessari alla creazione e all'elaborazione dei messaggi in formato *MIME* e che può facilmente essere interfacciato con altri programmi che non dispongano di tali funzionalità. Esso può essere utilizzato anche per realizzare *script* per elaborare o gestire messaggi *e-mail* contenenti allegati.

SMTP

Cenni sul protocollo SMTP

SMTP (*Simple Mail Transfer Protocol*) è un protocollo, basato su TCP, che permette di trasferire i messaggi di posta elettronica. Viene utilizzato sia per il collegamento fra MUA e MTA che per lo scambio della posta fra MTA. Per quest'ultimo scopo vengono utilizzati anche altri protocolli come UUCP.

Per quanto riguarda SMTP, la macchina che deve spedire il messaggio di posta elettronica implementa la parte *client* del protocollo, mentre quella che lo riceve funge da *server*. Su di essa deve essere attivo un demone SMTP associato alla porta TCP 25.

È importante notare che SMTP funziona solamente in un verso: ad esempio un MTA non può instaurare una connessione verso un altro MTA per richiedergli la propria posta, bensì può solamente rimanere in ascolto fino a quando l'altro non apre una connessione per spedirgliela. In realtà nelle nuove versioni di SMTP (ESMTP) esiste un meccanismo (ETRN) che permette di chiedere all'altro MTA di svuotare una determinata coda di *e-mail* (anche se logicamente le cose vanno come richiesto, formalmente è comunque l'altro MTA ad instaurare la connessione).

Protocollo SMTP	Descrizione
/usr/lib/sendmail -v mrossi@test.it <messaggio.txt mrossi@test.it ... *Connecting to mail.test.it* via esmtp ...	*Sendmail* viene utilizzato come MDA da linea di comando (in modalità test -v). Mediante una *query* al DNS viene trovato l'indirizzo del MTA che funge da MX per il dominio test.it e viene instaurata una connessione verso esso.

220 *mail.test.it ESMTP Sendmail* 8.11.0/8.11.0; Fri, 29 Mar 2002 20:01:34 +0100 >>> *EHLO* pc001.prova.it 250 - *mail*.test.it *Hello* [243.211.173.213], *pleased to meet you* 250 - *ENHANCEDSTATUSCODES* 250 - *EDITMIME* 250 - *SIZE* 250 - DSN 250 - ONEX 250 - ETRN 250 - XUSR 250 - *HELP*	La macchina che funge da *client* ed il *server* si scambiano informazioni sulla versione del protocollo utilizzato e sulle opzioni disponibili. A questo livello è possibile implementare nel MTA delle restrizioni di accesso basate sull'indirizzo del *client*.

>>> *MAIL From*:beppe@prova.it> *SIZE*=S
250 2.1.0 <beppe@prova.it>... *sender ok*
 	
>>> *RCPT To*:<mrossi@test.it>
250 2.1.5 <mrossi@test.it>... *Recepient ok*
	Il *client* passa al *server* le informazioni sul mittente e sul destinatario del messaggio. Generalmente MTA viene configurato per eseguire delle verifiche sugli indirizzi passati (ad esempio che si tratti di indirizzi per i quali è concesso il *relay* oppure che i valori passati mediante SMTP coincidano con quelli presenti nell'*header* del messaggio (*From* e *To*))

>>> DATA
354 *Enter mail, end with . on a line by itself*
>>> .
250 2.0.0 g2TJIZa07454 *Message accepted for delivery*
beppe@prova.it... *Sent* (2.0.0 g2TJIZa07454 *Message accepted for delivery*)
>>> *QUIT*
221 2.0.0 mail.test.it *closing connection*
	Mediante il comando DATA avviene il trasferimento del corpo del messaggio, che viene passato al *server* in formato ASCII, terminando con una linea contenente un singolo punto. A questo punto il *server* restituisce al *client* un numero di protocollo identificativo del messaggio (che viene inserito anche nell'*header*) e il *client* termina la connessione con *QUIT*.

In figura è possibile vedere le diverse fasi del trasferimento di un messaggio di posta elettronica utilizzando il protocollo SMTP. Per la spedizione è stato utilizzato *Sendmail*, il quale può svolgere le funzioni di MDA se viene lanciato specificando come parametro nella linea di comando utilizzando un indirizzo di *e-mail* valido. Nell'esempio è stato passato in *input* a *Sendmail* un *file* (che dovrebbe essere stato in precedenza adeguatamente preformattato in modo da contenere almeno gli *header From*:, *To*: e *Subject*:). È anche possibile scrivere il messaggio direttamente nella linea di comando facendolo terminare con un CTRL-D:

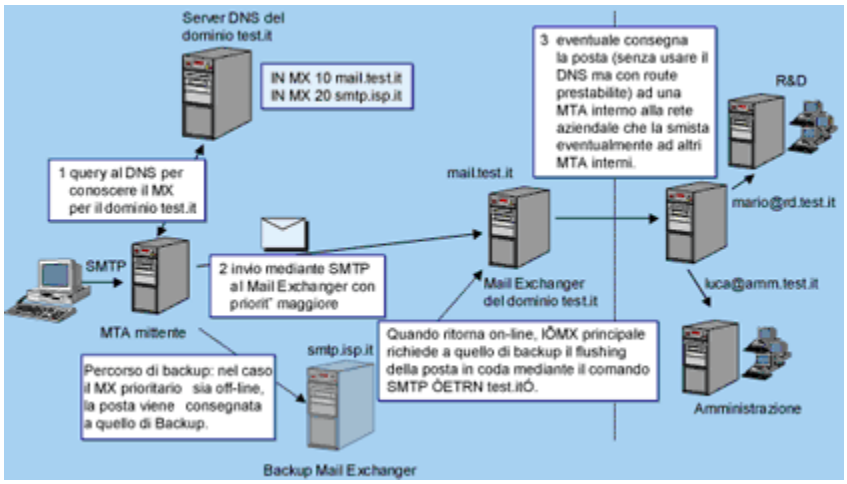
```
# /usr/sbin/sendmail info@test.it
From: Test
To: info@test.it
Subject: prova
Test
^D
```

Il ruolo del DNS nel protocollo SMTP

Rispetto ad alcuni anni or sono, il *routing* della posta elettronica è molto migliorato in efficienza, specialmente per il fatto che attualmente la maggior parte dei *server* sono perennemente connessi ad Internet. Questo ha permesso da una parte di ridurre i tempi di inoltro dei messaggi, dall'altra di sfruttare la presenza del DNS, che consente di non

dover utilizzare (e mantenere aggiornate) complicate tabelle di *routing*.

Il ruolo del DNS non deve essere sottovalutato, in quanto spesso la causa di problemi con la posta elettronica non è dovuta ad una errata configurazione del MTA, bensì a problemi riconducibili al DNS.



Il ruolo del DNS nella consegna della posta è descritto di seguito:

- quando un MTA tenta di spedire un messaggio di *e-mail* ad un determinato destinatario, per prima cosa effettua una *query* al DNS per ricavare l'indirizzo del MTA che funge da *Mail Exchanger* (scambiatore di posta) per l'indirizzo di destinazione. Ad uno stesso dominio possono essere associati più *Mail Exchanger*, per creare una ridondanza oppure per mandare a MTA diversi la posta destinata a sottodomini diversi. Gli indirizzi dei *Mail Exchanger* vengono gestiti mediante dei record di tipo MX nella zona del DNS associata al dominio.
- Una volta ricavati gli indirizzi dei *Mail Exchanger*, l'MTA tenta l'invio del messaggio mediante una connessione SMTP, iniziando dall'MX con priorità maggiore, fino a che la spedizione ha successo. Nel caso la spedizione non vada a buon fine, il messaggio viene tenuto per un certo periodo di tempo in una coda, ritentando l'operazione ad intervalli regolari di tempo.
- Un *Mail Exchanger* può essere il *server* di posta del dominio destinatario (in questo caso il messaggio viene inserito nella *mailbox* dell'utente) oppure può essere configurato in modo da passare la posta ricevuta ad un altro MTA, ad esempio interno alla Intranet aziendale. In questo caso la decisione di *routing* non avviene più mediante una *query* al DNS (che ritornerebbe l'indirizzo del MX stesso), bensì utilizzando un percorso prestabilito.

Quello appena visto è solamente un esempio semplificato che spiega il funzionamento del meccanismo. Nelle pratica si possono trovare diverse soluzioni, dipendenti dalla topologia della rete e dalla politica di gestione del servizio.

Ridondanza e MX di *backup*

Nel caso il *Mail Exchanger* con priorità maggiore sia momentaneamente non disponibile, i messaggi di posta elettronica vengono presi in consegna da un MX secondario, che li tiene in coda e tenta ad intervalli regolari di spedirli al MX principale. Questo avviene utilizzando le informazioni sui record MX prese dal DNS, con alcune cautele, come quelle di non rispedirsi il messaggio e di non creare dei *loop* con altri MX secondari.

In questo modo è possibile realizzare un sistema di *backup*, che, assieme al meccanismo delle code, è in grado di assicurare una certa ridondanza ed affidabilità al sistema.

L'estensione *MIME* ETRN

A questo proposito è importante considerare che se l'MTA di *backup* è configurato in modo da riprovare a spedire la

posta in coda a quello primario ad intervalli di tempo troppo lunghi, possono esservi dei rallentamenti anche significativi nella consegna. Nei casi particolarmente sfortunati può anche accadere che per più volte si tenti di rispedire una coda di messaggi proprio nei momenti in cui l'MTA destinatario non è disponibile.

Una possibile soluzione al problema appena incontrato consiste nell'utilizzare il meccanismo di ETRN (*Extended TuRN*), descritto nell'RFC1985.

Il funzionamento è il seguente: quando il *server* principale è disponibile *on-line*, la posta che arriva viene consegnata direttamente ad esso, essendo l'MX con priorità maggiore. Quando invece il *Mail Exchanger* prioritario non è disponibile, la posta viene consegnata a quello di *backup*, che la tiene in coda. Nel momento in cui il *server* principale torna disponibile, esso esegue un collegamento SMTP verso la porta 25 della macchina di *backup* ed impartisce il comando ETRN nomedominio, dopodiché termina la connessione.

Il comando ETRN causa lo svuotamento della coda relativa al dominio indicato da parte del MTA di *backup* verso il MX prioritario. Tentativo che và a buon fine, dato che l'MTA destinatario risulta sicuramente *on-line*.

L'operazione di solito viene eseguita mediante un apposito programma *client*, ad esempio lo *script* `etrn.pl` fornito assieme a *Sendmail*. A scopo didattico può essere interessante simularla a mano mediante telnet:

```
$ telnet smtp.isp.it 25
Trying 145.23.45.67...
Connected to smtp.isp.it.
Escape character is '^]'.
220 smtp.isp.it ESMTP Sendmail 8.9.3/8.9.3; Fri, 5 Apr 2002 13:07:38 +020
0
ETRN test.it
250 Queuing for node test.it started
quit
221 mail.test.it closing connection
Connection closed by foreign host.
```

Sendmail

Sendmail è l' **MTA** più diffuso, in quanto disponibile come standard su tutte le versioni di *UNIX*. Esso offre ampie possibilità di personalizzazione e dalla ottima flessibilità, a scapito di una certa difficoltà di configurazione. Nel corso del capitolo vengono analizzate le varie componenti del sistema e la funzione dei *file* di configurazione, ponendo in particolare l'attenzione sulle problematiche inerenti la sicurezza.

Sendmail può essere utilizzato in differenti modalità, a seconda dei parametri con cui viene chiamato nella linea di comando. La modalità operativa viene selezionata mediante l'opzione -b:

- *sendmail* -bd: funzionamento come demone;
- *sendmail* -bi: *rebuild* del *database* degli alias (*newaliases*);
- *sendmail* -bp: stampa la coda di posta (*mailq*);
- *sendmail* -bt: *test mode* (risoluzione dell'indirizzo senza effettiva spedizione del messaggio).

Specificando invece nella linea di comando un indirizzo di *e-mail*, *Sendmail* può essere utilizzato come un *delivery agent*:

- *sendmail* info@test.it: utilizzo come *mail delivery agent*.

Nel testo verrà utilizzata la scrittura *Sendmail* per indicare il *software* e la scrittura *sendmail* (minuscolo) per indicare il programma eseguibile, che generalmente si trova in `/usr/lib/sendmail` (per brevità il percorso completo non viene indicato).

Analiziamo ora in dettaglio i due utilizzi più frequenti.

Funzionamento come demone (`/usr/lib/sendmail -bd -q5m`)

Utilizzato come *server* SMTP, *Sendmail* è in grado di gestire sulla porta TCP 25 collegamenti SMTP entranti da parte di MUA o altri MTA. Generalmente esso viene avviato come un *server* a sè stante (parametro `-bd`) utilizzando uno *script* di avvio in `/etc/rc.d`. Ovviamente se il sistema non deve ricevere posta dall'esterno mediante SMTP, non è necessario avviare il servizio.

Il parametro `-q5m` indica l'intervallo di tempo che deve intercorrere fra i diversi tentativi di rispedizione di un messaggio che si trova in coda perché i tentativi precedenti di consegna sono falliti.

In *Red Hat Linux* per controllare il servizio si utilizza lo *script* di avvio `/etc/rc.d/rc3/S80sendmail`, richiamandolo con gli opportuni parametri:

- *`/etc/rc.d/rc3/S80sendmail start`*;
- *`/etc/rc.d/rc3/S80sendmail stop`*;
- *`/etc/rc.d/rc3/S80sendmail restart`*.

Funzionamento come *Mail Delivery Agent* (`/usr/lib/sendmail mrossi@test.it`)

Sendmail può essere utilizzato anche come *mail delivery agent* per inviare posta mediante il protocollo SMTP. Di fatto nei sistemi in cui *Sendmail* viene utilizzato come *server* SMTP, esso viene utilizzato anche come MDA per il protocollo SMTP. Quando la copia di *sendmail* in attesa sulla porta 25 riceve un nuovo messaggio non destinato ad un utente locale, essa esegue una nuova copia di *sendmail*, questa volta utilizzato come **MDA** , per il *delivery* via SMTP. Entrambe le copie utilizzano i medesimi *file* di configurazione.

Configurazione di Sendmail

Il *file* di configurazione base di *Sendmail* è:

- `/etc/sendmail.cf`.

Oltre ad esso vengono utilizzati altri *file* per mantenere le informazioni configurabili dall'utente ed i *database* (su molte versioni di *UNIX* sono collocati nella *directory* `/etc/mail`). Il nome esatto dei *file* può dipendere dalla particolare installazione e non è detto che nella propria configurazione di *Sendmail* tutte le funzionalità (*features*) che analizzeremo siano attivate.

Nel caso *Sendmail* venga fornito già parzialmente configurato, come avviene per esempio in *Red Hat Linux*, le operazioni di personalizzazione possono nella maggioranza dei casi limitarsi alla gestione dei *file* e dei *database* contenuti nella *directory* `/etc/mail`. Le modifiche più complesse, come l'aggiunta di nuove funzionalità, richiedono invece la modifica del *file* di configurazione globale di *Sendmail*, `/etc/sendmail.cf`, che verrà descritta nei prossimi paragrafi.

I principali *file* di configurazione sono i seguenti:

- `/etc/mail/access` (`/etc/mail/access.db`). *Database* contenente le *access list*. L'argomento verrà discusso nel seguito con maggiore dettaglio.
- `/etc/mail/domaintable` (`/etc/mail/domaintable.db`). *Database* di supporto alla funzione *domaintable*, che facilita il passaggio da un vecchio ad un nuovo nome di dominio, permettendo di utilizzare contemporaneamente più nomi.
- `/etc/mail/local-host-names`. Permette di definire i nomi dei domini che devono essere considerati

locali al sistema.
- `/etc/mail/mailertable` (`/etc/mail/mailertable.db`). *Database* di supporto alla funzione *mailertable*, la quale permette di associare ad un nome di dominio un determinato metodo di trasporto. Questo permette ad esempio di reindirizzare tutta la posta di un determinato dominio ad un altro MTA (prova.it smtp:server.pippo.it).
- `/etc/mail/trusted-users`. Lista di utenti che possono spedire messaggi assumendo l'identità altrui senza generare messaggi di errore (*root*, uucp, ...).
- `/etc/mail/virtusertable` (`/etc/mail/virtusertable.db`). *Database* di supporto alla funzione *virtusertable*. Essa permette di mappare domini virtuali in nuovi indirizzi. Si noti che gli indirizzi presenti nell'*header* del messaggio non vengono modificati.
- `/etc/aliases` (`/etc/aliases.db`). *Database* degli alias per gli indirizzi locali (viene ricreato da `/etc/aliases` utilizzando il comando *newaliases*).

Una descrizione più dettagliata delle *features* corrispondenti ai *file* appena descritti verrà effettuata nel seguito.

Database **in formato** ***hash***

I *file* con estensione `.db` contengono *database* in formato *hash*. Essi vengono generati mediante il comando *makemap* a partire da un corrispondente *file* in formato testo. Nella lista precedente sono stati indicati sia i nomi dei *file* di testo sorgente, sia, fra parentesi, i nomi dei *database* risultanti.

Volendo modificare il contenuto del *database mailertable*, si dovranno compiere le seguenti operazioni:

- modifica mediante un *editor* del *file* `/etc/mail/mailertable`.
- Rigenerazione del *database* `/etc/mail/mailertable.db` mediante il comando *`makemap hash /etc/mail/mailertable < /etc/mail/mailertable`*.

Nella *directory* `/etc/mail` è presente un *Makefile* che permette di semplificare tali operazione mediante il comando *make*.

Domini virtuali

Mediante *virtusertable* è possibile definire dei domini virtuali, operazione non possibile semplicemente inserendo più nome in `/etc/mail/local-domain-names`, in quanto così facendo gli utenti info@test.it e info@prova.com verrebbero mappati sul medesimo utente info.

Mediante *virtusertable* è possibile associare i due indirizzi ad utenti *UNIX* differenti:

info@test.it a.verdi
info@prova.com m.rossi

La ridirezione può avvenire anche verso indirizzi esterni:

info@test.it c.gialli@tin.it
info@prova.com n.verdi@hotmail.com

È inoltre possibile mandare tutta la posta di un dominio nella *mailbox* di un singolo utente:

info@test.it infotest
@test.it d.arancio
sales@prova.com m.verdi
info@prova.com r.rosa
@prova.com s.viola@mail.it

@www.prova.com webmaster
@www.test.it webmaster

Funzionamento di Sendmail

Il funzionamento di *Sendmail* è basato su un *rewriting engine*, il quale, in base all'indirizzo del destinatario, si occupa del *routing* dei messaggi di posta elettronica, restituendo come risultato il *mailer delivery agent* più opportuno per la spedizione del messaggio.

Quando si spedisce un messaggio, l'indirizzo del destinatario viene innanzitutto tokenizzato, ovvero spezzato in parti utilizzando alcuni caratteri speciali come separatori. Ad esempio l'indirizzo mrossi@test.it diventa

- mrossi;
- @;
- test;
- .;
- it;

Successivamente l'indirizzo viene elaborato mediante delle *subroutine* (*rule sets*) secondo il flusso schematizzato in figura. Le *subroutine* da S0 a S5 hanno i seguenti scopi predefiniti:

- S0: restituisce il *mail delivery agent* con cui spedire il messaggio;
- S1: elabora l'indirizzo del mittente;
- S2: elabora l'indirizzo del destinatario;
- S3: preelabora tutti gli indirizzi;
- S4: postelabora gli indirizzi;
- S5: riscrive gli indirizzi degli utenti locali (*unaliased*);
- Snn: *rule sets* specifici per i vari *mailer*;

Ogni *subroutine* è costituita da un insieme di regole di riscrittura (*rules*), descritte nel *file* di configurazione `/etc/sendmail.cf`, come una lista di espressioni regolari con cui di volta in volta viene confrontato l'indirizzo destinatario:

- Rlhs <tab> rhs <tab> commento.

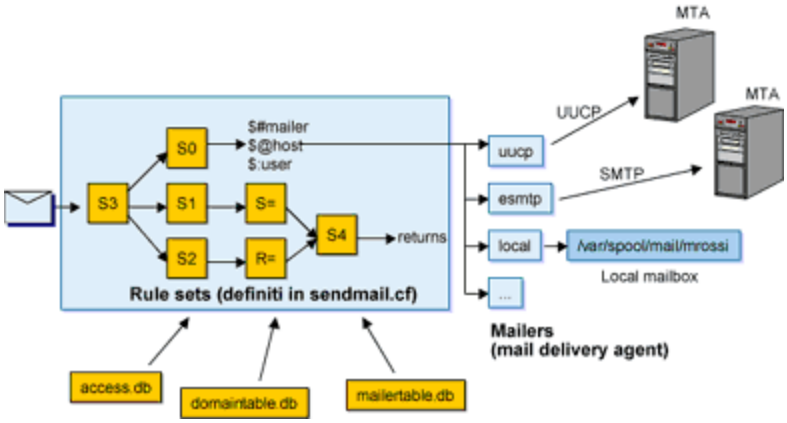
Se l'indirizzo verifica espressione lhs (*Left hand side*), esso viene riscritto secondo quanto specificato in rhs (*Right hand side*).

Oltre alla riscrittura dell'indirizzo, rhs può contenere la chiamata ad una *subroutine* oppure la restituzione di una variabile. In uscita dalla *subroutine* S0 viene restituita una terna che contiene:

- $#*mailer* il nome del *mail delivery agent* da utilizzare per la spedizione (esmtp, uucp, *smarthost*, ...);
- $@*host* l'indirizzo dell'MTA a cui spedire il messaggio;
- $:*user* indirizzo riscritto dell'utente;

Il *file* di configurazione `/etc/sendmail.cf`

La maggior parte delle informazioni necessarie al funzionamento di *Sendmail* sono definite all'interno del *file* di configurazione `/etc/sendmail.cf`. La sintassi utilizzata è abbastanza complessa, perché all'interno di esso sono presenti anche le regole di riscrittura.



In questa sede non entreremo troppo in dettaglio sulla configurazione di `sendmail.cf`, in quanto esiste un metodo più semplice di configurazione, che ne prevede la generazione automatica a partire da un formato più comprensibile. Vale tuttavia la pena conoscerne almeno il formato generale:

Ogni linea di *sendmail* ha uno scopo preciso, identificato dal primo carattere:

- D: definisce una macro;
- C: definisce una *class* (macro contenente una lista di valori);
- F: definisce una *class* leggendo i valori da un *file* oppure dall'*output* di un programma;
- K: esegue una ricerca su un *database*;
- V: indica la versione del *file* di configurazione;
- T: definisce i *trusted users* (utenti fidati);
- P: definisce una priorità di *delivery*;
- O: definisce una opzione;
  - *Character Options*: OL9
  - *String Options*: OLogLevel=9
- H: definisce un *header*;
- M: definisce un *mail delivery agent*;
- S: indica l'inizio di una *subroutine* (*rule set*);
- R: definisce una regola di riscrittura.

Esempi:

Per offrire una idea delle diverse possibilità, di seguito sono descritti alcune linee di configurazione in `/etc/sendmail.cf`. Per maggiori dettagli o per conoscere le altre opzioni si possono studiare direttamente i commenti contenuti all'interno del *file*.

- `Cwlocalhost test.it www.test.it dns.test.it linux.test.it`. Definisce la lista degli *hostname* o nomi di dominio che devono essere considerati locali, ovvero se viene ricevuta una *mail* destinata ad un utente del tipo mrossi@www.test.it, essa viene consegnata all'utente locale mrossi. Nelle versioni di *Sendmail* successive alla V8.7 è possibile utilizzare una scrittura del tipo C{*hosts*} *localhost* test.it www.test.it dns.test.it linux.test.it.
- `Fw/etc/mail/local-host-names`. Ha la stessa funzione della linea precedente, con la differenza che la lista viene letta dal *file* `/etc/mail/local-host-names`.
- `DSmail.tin.it`. Definisce lo *smarthost*. Viene utilizzata per passare in blocco tutta la posta in uscita ad un MTA che si occupi di smistarla (ad esempio al *server* SMTP del *provider* Internet oppure all'MTA centrale dell'azienda).
- `Dmtest.it`. Permette di mascherare il nome dell'*host* nel campo mittente dei messaggi in uscita (utente@pc001.test.it diventa utente@test.it).
- `Kaccess hash -o /etc/mail/access`. Definisce il *database* in formato *hash* da utilizzare per il controllo

della *access list*.

Alcune *Options* utili sono le seguenti:

- OLogLevel=9 (oppure OL9): definisce il livello di *logging*;
- Ox12: definisce il carico della CPU oltre il quale i messaggi non devono essere spediti bensì messi in coda;
- OX8: definisce il carico della CPU oltre il quale non accettare ulteriori connessioni;
- OT24h1m: definisce il *timeout*, scaduto il quale un messaggio in coda viene rispedito al mittente;
- OT2d/1h: definisce il *timeout*, scaduto il quale viene notificato al mittente che un suo messaggio è ancora in coda;
- `OQ/usr/spool/mqueue`: definisce la *directory* di *spool* dei messaggi.

Le seguenti macro risultano predefinite in `/etc/sendmail.cf`:

- Dw: nome dell'*host* (pc001);
- Dm: *domain name* (test.it);
- Dj: nome canonico dell'*host* (pc0011.test.it);
- Dv: versione di *sendmail*;
- Dn: nome del mittente da utilizzare nella spedizione dei messaggi di errore (esempio: *Mailer-Daemon*);

Configurazione mediante M4

Un metodo più semplice per configurare *sendmail* consiste nella generazione automatica del *file* `sendmail.cf` a partire da un formato più comprensibile, basato su delle macro che vengono convertite nel linguaggio di `sendmail.cf` utilizzando il processore di macro M4.

Le operazioni da compiere per ottenere un *file* di configurazione di *sendmail* utilizzando m4 sono le seguenti:

- generazione di un *file* (esempio: `file.mc`) in cui siano descritte, sotto forma di macro, le caratteristiche (*features*) desiderate per la propria installazione di *sendmail*;
- trasformazione di `file.mc` nel corrispondente *file* .cf mediante m4 (il percorso del *file* cf.m4 dipende dalla versione di *sendmail* e dal sistema operativo utilizzati): `m4 /usr/share/sendmail-cf/m4/cf.m4 <file.mc> > file.cf`
- eventuale test del nuovo *file* di configurazione risultante (`sendmail -Cfile.cf -v -bt info@test.it`);
- installazione del *file* .cf come `/etc/sendmail.cf`;
- riavvio del servizio *sendmail*.

Poiché il *file* sorgente in formato .mc è necessario nel caso si volessero apportare delle modifiche successive, è buona norma tenerlo in una *directory* che non venga cancellata durante un eventuale aggiornamento del sistema. Una buona soluzione potrebbe essere quella di copiarlo in `/etc/sendmail.mc`.

Seppur possibile, è buona norma evitare di apportare modifiche direttamente a `/etc/sendmail.cf`, ma è preferibile imparare come compiere l'operazione attraverso il *file* .mc, essendo esso più indipendente dalla versione di *Sendmail* utilizzata. In caso di aggiornamento del sistema si potrà utilizzare tale *file* per ricreare `/etc/sendmai.cf` utilizzando la versione di cf.m4 fornita col nuovo sistema.

Esempio di configurazione

Di seguito è mostrato il file di configurazione di sendmail tratto da *Red Hat Linux* 9 (redhat.mc).

```
divert(-1)dnl
dnl #
dnl # This is the sendmail macro config file for m4. If you make changes to
dnl # /etc/mail/sendmail.mc, you will need to regenerate the
dnl # /etc/mail/sendmail.cf file by confirming that the sendmail-cf package is
dnl # installed and then performing a
dnl #
dnl # make -C /etc/mail
dnl #
include('/usr/share/sendmail-cf/m4/cf.m4')dnl
VERSIONID('setup for Red Hat Linux')dnl
OSTYPE('linux')dnl
dnl #
dnl # Uncomment and edit the following line if your outgoing mail needs to
dnl # be sent out through an external mail server:
dnl #
dnl define('SMART_HOST','smtp.your.provider')
dnl #
define('confDEF_USER_ID',''8:12'')dnl
define('confTRUSTED_USER', 'smmsp')dnl
dnl define('confAUTO_REBUILD')dnl
define('confTO_CONNECT', '1m')dnl
define('confTRY_NULL_MX_LIST',true)dnl
define('confDONT_PROBE_INTERFACES',true)dnl
define('PROCMAIL_MAILER_PATH','/usr/bin/procmail')dnl
define('ALIAS_FILE', '/etc/aliases')dnl
dnl define('STATUS_FILE', '/etc/mail/statistics')dnl
define('UUCP_MAILER_MAX', '2000000')dnl
define('confUSERDB_SPEC', '/etc/mail/userdb.db')dnl
define('confPRIVACY_FLAGS', 'authwarnings,novrfy,noexpn,restrictqrun')dnl
define('confAUTH_OPTIONS', 'A')dnl
dnl #
dnl # The following allows relaying if the user authenticates, and disallows
dnl # plaintext authentication (PLAIN/LOGIN) on non-TLS links
dnl #
dnl define('confAUTH_OPTIONS', 'A p')dnl
dnl #
dnl # PLAIN is the preferred plaintext authentication method and used by
dnl # Mozilla Mail and Evolution, though Outlook Express and other MUAs do
dnl # use LOGIN. Other mechanisms should be used if the connection is not
dnl # guaranteed secure.
dnl #
dnl TRUST_AUTH_MECH('EXTERNAL DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
dnl define('confAUTH_MECHANISMS', 'EXTERNAL GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')
dnl
dnl #
dnl # Rudimentary information on creating certificates for sendmail TLS:
dnl # make -C /usr/share/ssl/certs usage
dnl #
dnl define('confCACERT_PATH','/usr/share/ssl/certs')
dnl define('confCACERT','/usr/share/ssl/certs/ca-bundle.crt')
dnl define('confSERVER_CERT','/usr/share/ssl/certs/sendmail.pem')
dnl define('confSERVER_KEY','/usr/share/ssl/certs/sendmail.pem')
dnl #
dnl # This allows sendmail to use a keyfile that is shared with OpenLDAP's
dnl # slapd, which requires the file to be readble by group ldap
dnl #
dnl define('confDONT_BLAME_SENDMAIL','groupreadablekeyfile')dnl
dnl #
dnl define('confTO_QUEUEWARN', '4h')dnl
dnl define('confTO_QUEUERETURN', '5d')dnl
dnl define('confQUEUE_LA', '12')dnl
```

```
dnl define('confREFUSE_LA', '18')dnl
define('confTO_IDENT', '0')dnl
dnl FEATURE(delay_checks)dnl
FEATURE('no_default_msa','dnl')dnl
FEATURE('smrsh','/usr/sbin/smrsh')dnl
FEATURE('mailertable','hash -o /etc/mail/mailertable.db')dnl
FEATURE('virtusertable','hash -o /etc/mail/virtusertable.db')dnl
FEATURE(redirect)dnl
FEATURE(always_add_domain)dnl
FEATURE(use_cw_file)dnl
FEATURE(use_ct_file)dnl
dnl #
dnl # The -t option will retry delivery if e.g. the user runs over his quota.
dnl #
FEATURE(local_procmail,'','procmail -t -Y -a $h -d $u')dnl
FEATURE('access_db','hash -T<TMPF> -o /etc/mail/access.db')dnl
FEATURE('blacklist_recipients')dnl
EXPOSED_USER('root')dnl
dnl #
dnl # The following causes sendmail to only listen on the IPv4 loopback address
dnl # 127.0.0.1 and not on any other network devices. Remove the loopback
dnl # address restriction to accept email from the internet or intranet.
dnl #
DAEMON_OPTIONS('Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
dnl #
dnl # The following causes sendmail to additionally listen to port 587 for
dnl # mail from MUAs that authenticate. Roaming users who can't reach their
dnl # preferred sendmail daemon due to port 25 being blocked or redirected find
dnl # this useful.
dnl #
dnl DAEMON_OPTIONS('Port=submission, Name=MSA, M=Ea')dnl
dnl #
dnl # The following causes sendmail to additionally listen to port 465, but
dnl # starting immediately in TLS mode upon connecting. Port 25 or 587 followed
dnl # by STARTTLS is preferred, but roaming clients using Outlook Express can't
dnl # do STARTTLS on ports other than 25. Mozilla Mail can ONLY use STARTTLS
dnl # and doesn't support the deprecated smtps; Evolution <1.1.1 uses smtps
dnl # when SSL is enabled-- STARTTLS support is available in version 1.1.1.
dnl #
dnl # For this to work your OpenSSL certificates must be configured.
dnl #
dnl DAEMON_OPTIONS('Port=smtps, Name=TLSMTA, M=s')dnl
dnl #
dnl # The following causes sendmail to additionally listen on the IPv6 loopback
dnl # device. Remove the loopback address restriction listen to the network.
dnl #
dnl # NOTE: binding both IPv4 and IPv6 daemon to the same port requires
dnl # a kernel patch
dnl #
dnl DAEMON_OPTIONS('port=smtp,Addr=::1, Name=MTA-v6, Family=inet6')dnl
dnl #
dnl # We strongly recommend not accepting unresolvable domains if you want to
dnl # protect yourself from spam. However, the laptop and users on computers
dnl # that do not have 24x7 DNS do need this.
dnl #
FEATURE('accept_unresolvable_domains')dnl
dnl #
dnl FEATURE('relay_based_on_MX')dnl
dnl #
dnl # Also accept email sent to "localhost.localdomain" as local email.
dnl #
LOCAL_DOMAIN('localhost.localdomain')dnl
```

```
dnl #
dnl # The following example makes mail from this host and any additional
dnl # specified domains appear to be sent from mydomain.com
dnl #
dnl MASQUERADE_AS('mydomain.com')dnl
dnl #
dnl # masquerade not just the headers, but the envelope as well
dnl #
dnl FEATURE(masquerade_envelope)dnl
dnl #
dnl # masquerade not just @mydomainalias.com, but @*.mydomainalias.com as well
dnl #
dnl FEATURE(masquerade_entire_domain)dnl
dnl #
dnl MASQUERADE_DOMAIN(localhost)dnl
dnl MASQUERADE_DOMAIN(localhost.localdomain)dnl
dnl MASQUERADE_DOMAIN(mydomainalias.com)dnl
dnl MASQUERADE_DOMAIN(mydomain.lan)dnl
MAILER(smtp)dnl
MAILER(procmail)dnl
```

I principali tipi di macro presenti nel *file* sono i seguenti:

- le righe inizianti con dnl introducono dei commenti;
- *define*('variabile', 'valore'): definisce il valore di una variabile;
- *FEATURE*('nome','argomenti'): permette di includere in `sendmail.cf` una determinata funzionalità;
- *MAILER*(nome): permette di includere in `sendmail.cf` il supporto per un determinato *mail delivery agent*.

Ogni linea presente nel *file* .mc, quando viene elaborata mediante M4, genera delle linee corrispondenti nel *file* .cf risultante. La trasformazione viene effettuata secondo le regole contenute nel *file* cf.m4 e usando come prototipo il *file* `proto.cf` presente nella stessa *directory*.

Ad esempio la definizione della variabile *SMART_HOST*

```
define('SMART_HOST', 'mail.tin.it')dnl
```

genera nel *file* .cf la linea

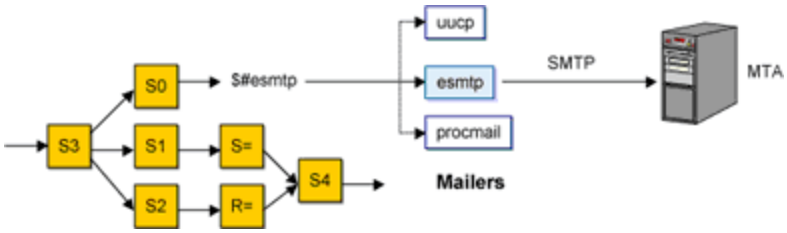
```
Dsmail.tin.it
```

Altre funzioni, ad esempio l'inclusione di un *MAILER* o di una *FEATURE*, possono generare risultati molto più complessi, ad esempio delle regole di riscrittura degli indirizzi.

Per una descrizione dettagliata del formato mc e delle funzionalità che possono essere incluse in *Sendmail* si faccia riferimento al documento *New Sendmail Configuration Files* (`README.cf`), incluso nella documentazione di *Sendmail*.

MAILERS (Mail Delivery Agent)

Per *mail delivery agent* (*mailer*) si intende un programma che si occupa del trasporto di un messaggio di posta elettronica mediante determinato protocollo (SMTP, consegna locale, UUCP, ...). Il nome del *mail delivery agent* da utilizzare per spedire un determinato messaggio viene ritornato come parametro $# in uscita dal *rule set* S0.



È possibile configurare *Sendmail* in modo da utilizzare più *mail delivery agent*, mediante la macro *MAILER*(nome, parametri) nel *file* di configurazione in formato mc:

```
MAILER(procmail)dnl
```

Essa viene convertita nel *file* .cf in linee simili alle seguente, che definiscono il programma e i parametri da utilizzare per il *delivery* della posta locale:

```
Mlocal,
P=/usr/bin/procmail,
F=lsDFMAw5:/|@qSPfhn9,
S=EnvFromL/HdrFromL,
R=EnvToL/HdrToL,
T=DNS/RFC822/X-Unix,
A=procmail -Y -a $h -d $u
```

In una configurazione di *sendmail* possono essere definiti diversi *mail delivery agent*, a seconda delle funzioni e del tipo di protocolli di trasporto della posta che si vogliono supportare. Ad esempio in un MTA di transito in cui si voglia solamente scambiare posta mediante il protocollo SMTP, sarà sufficiente definire:

```
MAILER(smtp)dnl
```

Se si desidera anche consegnare i messaggi per gli utenti locali nelle *mailbox* di sistema, occorrerà definire anche un *local mail delivery agent*:

```
MAILER(local)dnl
```

I *mail delivery agent* più comunemente utilizzati sono i seguenti:

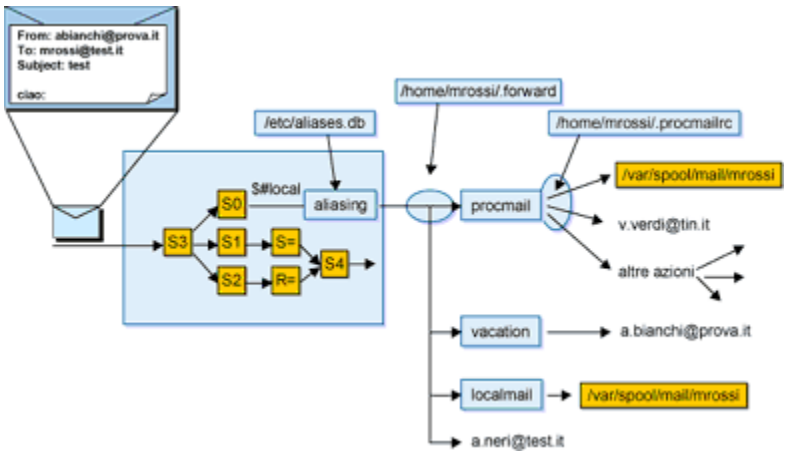
- *MAILER*(*local*).
- *MAILER*(prog). I *mailer local* e prog consentono rispettivamente il *delivery* dei messaggi nelle *mailbox* degli utenti locali e l'esecuzione di programmi alla ricezione di un messaggio. Generalmente sono presenti in tutte le installazioni, salvo il caso in cui tutta la posta venga consegnata ad uno *smarthost*.
- *MAILER*(*procmail*). Si tratta di una interfaccia verso il programma *procmail* (che non viene fornito con *sendmail*), utilizzabile come alternativa per il *delivery* locale della posta. *Procmail* viene utilizzato come *default* da molte versioni di *Linux*.
- *MAILER*(smtp). Permette il trasferimento dei messaggi mediante il protocollo SMTP. Se questa macro viene definita, vengono aggiunti quattro *mail delivery agent* con funzioni leggermente diverse: smtp, esmtp (*Extended* SMTP), smtp8 e *relay*. Se si utilizza uno *smarthost* (variabile *SMART_HOST*) viene definito anche il *mailer relay* associato ad esso.
- *MAILER*(uucp). Definisce il *mailer* relativo allo scambio della posta mediante il protocollo UUCP.
- *MAILER*(fax). Permette di spedire messaggi di *e-mail* sotto forma di FAX. Si tratta di una interfaccia verso il programma **HylaFAX**.

Delivery locale

Un MTA considera come locali, gli utenti con indirizzo del tipo utente@dominio, in cui dominio corrisponda al nome della macchina stessa oppure sia contenuto nella classe Cw. I nomi dei domini locali vengono generalmente letti da un *file* di testo (in *Red Hat Linux* viene utilizzato allo scopo `/etc/mail/local-host-names`):

- miodominio.com;
- miodominio2.com;
- www.miodominio.com;
- mbox.miodominio.com;
- server.miodominio.com;

Quando il destinatario viene riconosciuto come un utente locale, il messaggio viene consegnato all'utente *UNIX* corrispondente alla parte dell'indirizzo di *e-mail* che si trova a sinistra del simbolo @ (è possibile creare degli indirizzi fittizi utilizzando le funzioni di *aliasing*).



La consegna avviene utilizzando il *local mail delivery agent* e, a meno che non siano stati attivati degli alias o *forwarding* su quell'indirizzo, consiste nell'accodare i messaggi nel *file* corrispondente alla *mailbox* dell'utente.

- a.bianchi@miodominio2.com -> a.bianchi -> `/var/spool/mail/a.bianchi`;
- mrossi@miodominio.com -> mrossi -> `/var/spool/mail/mrossi`;
- mrossi@mbox.miodominio.com -> mrossi -> `/var/spool/mail/mrossi`;

Nonostante questa possa sembrare una operazione banale, in realtà vi sono delle complicazioni, come la necessità di gestire il *locking* del *file* per evitare che più programmi (o più copie del *local mailer*) vi possano scrivere contemporaneamente.

Alle volte risulta desiderabile collocare le *mailbox* in un'altra posizione all'interno del *filesystem*, ad esempio all'interno della *home directory* dell'utente *UNIX* corrispondente, allo scopo di poterne fare il *backup* assieme agli altri *file* dell'utente o per gestire un meccanismo di quote su disco. Per far ciò occorre configurare opportunamente il programma associato al *mailer local*. È tuttavia necessario aggiornare anche tutti gli eventuali programmi che gestiscono la posta elettronica, ad esempio il *server* POP3.

Aliasing

Mediante il *file* `/etc/aliases` è possibile definire degli indirizzi locali diversi dai nomi degli utenti *UNIX*.

Il formato di una linea `/etc/aliases` è il seguente:

nome alias: indirizzo/i reale a cui inviare i messaggi ricevuti oppure azione

Mediante il meccanismo degli alias è possibile ad esempio:

- associare più indirizzi al medesimo utente *UNIX* (non viene fatta differenza fra minuscole e maiuscole): Mario_Rossi: mrossi; Mario.Rossi: mrossi; m.rossi: mrossi;
- ridirigere un messaggio ricevuto da un determinato utente ad un altro indirizzo (*forwarding*): *postmaster*: *root*; *sales*: marco@test.it;
- distribuire copia della posta ricevuta su un indirizzo a più utenti: info: beppe, c.verdi@azienda.com, supporto;
- creare delle semplici *mailing list* (per utilizzi più complessi è conveniente utilizzare un *software* opportuno come *Majordomo*): cda: mrossi, averdi, s.bianchi@mail.it;
- scrivere i messaggi di posta elettronica ricevuti in un *file*: *news*: /tmp/news.txt;
- lanciare un programma alla ricezione di un messaggio (passando il messaggio come *input*): mail2www: |/usr/local/bin/mail2www; sms: |/usr/local/bin/smsgw 333123456;
- caricare una lista di alias da un *file*: clienti: :*include*:`/home/averdi/clienti.txt`;

Si noti che è possibile creare un alias che ridiriga verso un altro alias.

Per rendere attive le modifiche apportate al *file* `/etc/aliases` è necessario ricreare il *database* degli alias `/etc/aliases.db` utilizzando il comando *newaliases* (*sendmail* -bi).

Forwarding

Successivamente all'*aliasing*, *Sendmail* verifica se nella *home directory* dell'utente *UNIX* a cui deve essere inoltrato il messaggio di posta elettronica è presente un *file* di nome '`.forward`' (esempio: `/home/mrossi/.forward`). Se questo esiste, vengono eseguite le operazioni in esso descritte, che sono analoghe a quelle effettuabili mediante `/etc/aliases`. Il *file* '`.forward`' può essere gestito autonomamente dai singoli utenti, facendo attenzione ai permessi di scrittura.

Per spedire tutta la posta ricevuta ad un altro indirizzo, è sufficiente scriverlo all'interno del *file* '`.forward`':

```
mario.rossi@prova.it
```

Volendo mantenere una copia dei messaggi anche sulla casella locale si dovrà invece scrivere:

```
\mrossi, mario.rossi@prova.it
```

Il carattere di *backslash* \ davanti al nome dell'utente locale indica a *Sendmail* di non applicare ulteriormente il meccanismo degli alias (per evitare *loop*).

Nel caso si desideri abilitare un risponditore automatico o fare elaborare da un programma il messaggio ricevuto, si dovrà invece indicare all'interno di '`.forward`' il percorso del programma da eseguire, a cui il messaggio verrà passato in *input*:

```
\mrossi, |/usr/ucb/vacation mrossi
```

Protezione dallo SPAM

Protezione da abusi

Un MTA è un programma particolarmente complesso e pertanto soggetto a diversi problemi di sicurezza. Un

malintenzionato potrebbe utilizzare un MTA non correttamente configurato per:

- accedere ai dati appartenenti ad altri utenti del sistema;
- produrre malfunzionamenti nel sistema (eccessivo carico, consumo di disco o di banda);
- spedire messaggi con un falso mittente;
- spedire messaggi indesiderati (SPAM) agli utenti del nostro dominio;
- spedire messaggi indesiderati (SPAM) ad altri utenti.

In particolare è bene aver cura di evitare che il proprio *server* venga utilizzato come un *open relay*, ovvero come un MTA mediante cui chiunque in Internet possa spedire messaggi di *e-mail*. Questo col duplice scopo di non lasciare che altri consumino nostre risorse e per evitare di risultare i mittenti di messaggi indesiderati (SPAM), con la possibile conseguenza di avere il nostro MTA inserito in qualche *black list*.

Nelle nuove versioni di *Sendmail* sono presenti diverse funzionalità che permettono di tenere sotto controllo la sicurezza del proprio sistema. Esse sono descritte nel *file New Sendmail Configuration Files* (`README.cf`), incluso nella documentazione di *Sendmail*.

La principali funzioni di *Sendmail* relative alla sicurezza sono le seguenti:

- il *relay* è negato per *default*;
- presenza di un *Access Database* (`/etc/access`);
- possibilità di rifiutare messaggi in base al contenuto degli *header* o del corpo di un messaggio;
- possibilità di fare delle verifiche durante la ricezione di un messaggio mediante SMTP (*check_mail*, *check_rcpt*);
- utilizzo facoltativo del TCP *wrapper* (*libtcpwrap*);
- utilizzo di *procmail* come *local mailer*;
- possibilità di utilizzare una *blacklist* pubblica (RBL) per la verifica del mittente di un messaggio (**http://maps.vix.com/rbl/**).

Controllo del *relay* della posta

Generalmente un MTA che non sia di transito viene configurato in modo da accettare solamente le seguenti tipologie di messaggi:

- messaggi generati all'interno della propria rete e diretti verso qualunque indirizzo;
- messaggi generati dall'esterno e indirizzati verso utenti locali interni alla propria rete;

Il *relaying*, ovvero la trasmissione di messaggi da una macchina esterna al proprio dominio ad un'altra macchina esterna al proprio dominio, è vietato per *default* nelle versioni di *Sendmail* a partire dalla 8.9. Per tornare al comportamento precedente si deve utilizzare *FEATURE*('*_promiscuousrelay*').

Le versioni recenti di *Sendmail*, per *default* vietano il *relay* anche nel caso l'indirizzo del mittente ricevuto mediante il comando *MAIL FROM*: durante la ricezione di un messaggio usando SMTP non sia corretto o non sia possibile risalire alla macchina mittente mediante una *query* inversa al DNS.

Access Database

È possibile abilitare il *relay* da parte di determinati utenti, indirizzi IP o domini utilizzando le funzioni di *Access Database* presenti in *Sendmail*. Esse vengono abilitate mediante la funzionalità *FEATURE*('`access_db`'). La macro può accettare un secondo parametro che permette di specificare il tipo e la posizione del *database*: *FEATURE* ('`access_db`', '*`hash -o /etc/mail/access`*').

Il *file* `/etc/access` è una tabella contenente una lista di indirizzi con associate le relative azioni da compiere

quando si riceve un messaggio con quel mittente. Le possibili azioni sono le seguenti:

- *OK*: accetta il messaggio con il mittente indicato, anche se esso sarebbe stato vietato da altre regole (ad esempio se il nome del dominio non è risolvibile);
- *RELAY*: permette il *relay* di messaggi destinati verso il dominio indicato o ricevuti da esso. Serve come un *OK* implicito per eventuali ulteriori controlli;
- *REJECT*: rifiuta i messaggi il cui mittente o destinatario sia l'indirizzo indicato, emettendo un generico messaggio di errore;
- *DISCARD*: scarta il messaggio utilizzando il *mailer* `$#discard`;
- nnn messaggio: restituisce al MTA che sta effettuando la connessione SMTP il codice di errore e il testo specificati. Il codice d'errore nnn deve essere concorde allo standard RFC 821.

Trattandosi di un *database* in formato *hash* è necessario rigenerare il corrispondente *database* quando si apportano delle modifiche al *file* di testo `/etc/access`:

```
makemap hash /etc/mail/access < /etc/mail/access
```

Esempi

```
127.0.0.1 RELAY
192.168 RELAY
192.168.12 REJECT
129.79.6.220 RELAY
test.it RELAY
spammer.com 550 non accettiamo messaggi dal vostro dominio
good.spammer.com OK
john@hotmail.com REJECT
baduser@ REJECT
```

Utilizzando la *feature FEATURE*('*blacklist_recipients*') è possibile verificare anche gli indirizzi del destinatario (utenti o *host* locali):

a.verdi 550 questo utente ha cambiato *mail* in andrea.verdi@test.it
pc001.miodominio.com 550 questo *host* non accetta *mail*
e.neri@server02.miodominio.com 550 utente sospeso

Inoltre tale *feature* rende possibile anche il blocco della spedizione di messaggi da parte di utenti locali che abbiano come destinatari determinati indirizzi:

subscribe@barzellette.com 550 non potete spedire *mail* a questo destinatario

TCP *wrapper*

Le versioni di *Sendmail* distribuite nelle versioni recenti di *Linux* sono configurate di serie in modo da utilizzare il TCP *wrapper*. Questo permette di concedere l'accesso al servizio solamente agli *host* abilitati in `/etc/hosts.allow` e costituisce una ulteriore protezione da usare in associazione a quelle già presenti in *Sendmail*.

sendmail: 127.0.0.1, 192.168., mail.mioisp.it

Per il funzionamento del TCP *wrapper* si faccia riferimento alla *Unit* relativa ai servizi di rete.

Gestione della coda, logging e debugging

Mail queue	Gestione coda
/var/spool/mqueue/dfg32HULj12622 */var/spool/mqueue/qfg32HULj12622* */var/spool/mqueue/dfg32HULj346f4* */var/spool/mqueue/qfg32HULj346f4* */var/spool/mqueue/dfg32Hdl439dkk* */var/spool/mqueue/qfg32Hdl439dkk*	*# mailq* *var/spool/mqueue (3 requests)* ----Q-ID---- - --*Size*-- -----Q-*Time*----- ----*Sender/Recipient*---- g32HULj12622 *(l'asterisco indica un messaggio che sta venendo processato) 12498 *Tue Apr* 2 19:30 iltelefonofisso (*Deferred*: mail@virigilio.it.: *No route to host*) test@virigilio.it g32Hdl439dkk 6 *Fri Apr* 5 10:38 *root* (*host map: lookup (profuso.com): deferred*) beppe@profuso.com g32HULj346f4 55462 *Tue Apr* 2 19:28 beppe (*Deferred*: test.it.: *No route to host*) info@test.it

Gestione della coda di *sendmail*

La coda dei messaggi in transito viene gestita da *sendmail* mediante una coppia di *file* nella *directory* `/var/spool/mqueue`:

- dfXXXXXXXX contiene il corpo del messaggio;
- qfXXXXXXXX contiene le informazioni necessarie alla spedizione, gli *header* e l'esito dell'ultimo tentativo di spedizione.

Quando viene ricevuto un messaggio, *Sendmail* tenta di spedirlo immediatamente a destinazione utilizzando il *mail delivery agent* più adatto. Se la spedizione non và a buon fine, il messaggio viene posto in una coda e la spedizione viene ritentata dopo un certo intervallo di tempo.

L'intervallo fra i tentativi successivi può essere modificato agendo sul parametro `-q[time]` (esempio: `sendmail -bd -q10m`).

Esistono alcuni casi in cui il messaggio viene inserito direttamente nella coda, ad esempio se il *server* è troppo carico (opzione OX), oppure se il *mailer* con cui deve essere spedito è marcato come *expensive* in `sendmail.cf` (utile per evitare che la spedizione immediata di ogni messaggio causi eccessive chiamate nel caso di un MTA su una connessione *dial-up*).

La gestione della coda avviene richiamando *sendmail* con uno dei seguenti parametri nella linea di comando:

- `sendmail -bp`: mostra il contenuto della coda di posta (alias: *mailq*);
- `sendmail -q`: forza la spedizione dei messaggi in coda (*flushing*).

È possibile eseguire un *flushing* selettivo solamente di determinati messaggi della coda, utilizzando i seguenti parametri:

- qRstringa: forza la spedizione dei soli messaggi che contengono stringa nell'indirizzo del destinatario;
- qSstringa: forza la spedizione dei soli messaggi che contengono stringa nell'indirizzo del mittente;
- qIstringa: forza la spedizione dei soli messaggi che contengono stringa nell'identificativo del messaggio.

Ad esempio il comando *sendmail* -qRtest.it forza la spedizione dei soli messaggi il cui indirizzo di destinazione contiene la stringa test.it.

Logging

Il log di tutti i programmi che costituiscono il sottosistema di posta elettronica (MTA, *server* POP3, *server* IMAP, ...)

avviene utilizzando la *facility mail* del *logger* sistema (*syslog*). Nei sistemi *Linux* i messaggi relativi alla posta elettronica vengono indirizzati nel *file* `/var/log/maillog`, mediante la seguente linea in `/etc/syslogd.conf`:

```
# Log all the mail messages in one place
mail.* /var/log/maillog
```

Test e *debugging*

Uno degli strumenti di test più utili consiste nell'utilizzare *Sendmail* come *mail delivery agent* sfruttando l'opzione -v. Questo permette una semplice e veloce verifica sulla raggiungibilità di un dato indirizzo:

```
$ /usr/lib/sendmail -v mrossi@test.it
Test
^D
mrossi@test.it... Connecting to mail.test.it. via esmtp...
220 mail.test.it ESMTP Sendmail 8.11.0/8.11.0; Fri, 29 Mar 2002 20:01:34 +0100
>>> EHLO pc001.prova.it
250-mail.test.it Hello [243.211.173.213], pleased to meet you
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-SIZE
250-DSN
250-ONEX
250-ETRN
250-XUSR
250 HELP
>>> MAIL From:<beppe@prova.it> SIZE=5
250 2.1.0 <beppe@prova.it>... Sender ok
>>> RCPT To:<mrossi@test.it>
250 2.1.5 <mrossi@test.it>... Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> .
250 2.0.0 g2TJ1Za07454 Message accepted for delivery
beppe@prova.it... Sent (2.0.0 g2TJ1Za07454 Message accepted for delivery)
>>> QUIT
221 2.0.0 mail.test.it closing connection
```

L'opzione -d nella linea di comando di *sendmail* permette invece di gestire il *debugging* delle regole (*rewriting rules*) presenti nel *file* di configurazione.

- /usr/lib/sendmail -v -bt -dcategoria.livello,categoria.livello;
  - -d0-99.127 massimo livello di *debugging* (-d0-99.99 massimo raccomandato);
  - -d: equivalente a -d0-99.1;
  - -d0.1: mostra solo le informazioni generali;
  - -d21.15: *debug* delle regole di riscrittura degli indirizzi.

Il comando presenta un *prompt* in cui deve essere specificata la regola di partenza e l'indirizzo del destinatario da testare. Vengono restituiti il *mailer* utilizzato per la spedizione ($#), l'indirizzo dell'MTA a cui spedire il messaggio ($@) e l'indirizzo (riscritto) del destinatario ($:).

```
/usr/lib/sendmail -v -bt -d21.15
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 beppe@profuso.com
> rewrite: ruleset 3 input: beppe @ profuso . com
-----trying rule: $@
----- rule fails
```

```
-----trying rule: $*
-----rule matches: $: $1 < @ >
...
$3:
rewritten as: $# esmtp $@ profuso . com . $: beppe < @ profuso . com . >
rewrite: ruleset 198 returns: $# esmtp $@ profuso . com . $: beppe < @ profuso . com .
>
rewritten as: $# esmtp $@ profuso . com . $: beppe < @ profuso . com . >
rewrite: ruleset 0 returns: $# esmtp $@ profuso . com . $: beppe < @ profuso . com . >
```

In un sistema di produzione dove non sia consigliabile eseguire modifiche al *file* `/etc/sendmail.cf`, in quanto in uso, si può fare il *debugging* specificando un diverso *file* di configurazione, mediante l'opzione -Cfile.

Servizio POP3/IMAP

I servizi POP3 e IMAP permettono la ricezione dei messaggi in arrivo da parte di un *client* di posta elettronica. L'installazione del *software* che fornisce detti servizi non presenta particolari difficoltà rispetto all'installazione di un comune *server* lanciato mediante `inetd`, tuttavia è bene prestare comunque particolare attenzione, in quanto nelle versioni standard le *password* viaggiano in chiaro.

Server POP3/IMAP4

POP3 (*Postal Office Protocol*, RFC1939) permette lo scaricamento da parte di un MUA dei messaggi di posta elettronica dalla *mailbox* di un utente *UNIX* su un *server*. In alternativa è possibile utilizzare il protocollo IMAP, che offre maggiori possibilità, anche dal punto di vista della sicurezza, e permette la gestione delle cartelle di posta direttamente sul *server*.

Protocollo POP	Descrizione
telnet pop3.test.it 110 *Trying* 192.34.33.22... *Connected to localhost.localdomain.* *Escape character is ^].* *+ OK POP3 pop3.mail.it v7.59 server ready*	Per mostrare il funzionamento del protocollo POP3, le operazioni compiute dal *client* possono essere simulate utilizzando telnet per effettuare un collegamento alla porta TCP 110 del *server*. Già a questo livello è possibile introdurre delle limitazioni d'accesso nel *server* utilizzando il meccanismo del *TCP wrapper* (*/etc/hosts.allow*).
user mrossi *+ OK User name accepted, password please* *pass* pippo *+ OK Mailbox open, 58 messages*	Una volta instaurato il collegamento, viene effettuata una autenticazione sull'utente utilizzando il classico meccanismo di *username/password*. Si noti che la *password* viene passata in chiaro.
Retr 1 *+ OK 482 octets* *Date: Fri*, 24 Mar 2002 17:19:18 +0100 CET *From*: Andrea Verdi <a.verdi@test.it> *Subject*: Prova *Content-Length*: 260 Prova di invio messaggio. . *dele 58*	Mediante un semplice protocollo, il *client* può scaricare, uno ad uno, i messaggi contenuti nella *mailbox* dell'utente. Dopo avere scaricato un messaggio si può cancellarlo dalla *mailbox* nel *server* utilizzando il comando *DELE*

+ OK Message deleted	
quit *+ OK Sayonara* *Connection closed by foreign host*.	Il *client* termina la connessione con *QUIT*.

In figura è rappresentato un esempio di utilizzo del protocollo mediante il comando telnet, simulando l'attività di un *client* di posta. POP3 è mappato sulla porta TCP 110.

I *server* per i protocolli POP3 e IMAP forniti di serie con *Linux* (ipop3d, imapd) vengono attivati mediante il supervisore inetd (o xinetd) e non necessitano per funzionare di particolari configurazioni oltre a quelle tipiche dei *server* eseguiti in questa modalità.

Nelle versioni standard del protocollo POP3 la posta e le *password* viaggiano in chiaro, il che può rappresentare un rischio, specialmente se le medesime *password* sono utilizzate anche per altri servizi, ad esempio l'accesso alla *shell* sul *server* come utente *UNIX*. Pertanto è buona norma limitare mediante il TCP *wrapper* l'utilizzo del servizio in modo da evitarne l'accesso a *client* il cui traffico possa passare per reti su cui non si ha il controllo della sicurezza. Nel caso tale limitazione non fosse attuabile, esistono delle estensioni al protocollo che permettono la codifica delle *password* (APOP), che tuttavia non sono supportate da tutti i *client*. Come alternativa è possibile sfruttare un *tunneling* SSL mediante il programma **stunnel** per realizzare una versione criptata del protocollo.

L'utilizzo di inetd per l'avvio dei *server* POP3 e IMAP può rappresentare un problema di prestazioni nel caso si debba gestire un numero elevato di *mailbox*. Purtroppo il *software* standard fornito con *Linux* non permette il funzionamento in modalità standalone. Esistono altri *server* POP3/IMAP che possono essere utilizzati anche come demoni:

- ***Qpopper Qualcomm* 4.0 con supporto TLS/SSL**
- ***Cyrus IMAP Server***
- ***GNUpop***
- ***Cubic Circle Cucipop***
- ...

MTA su dial-up

In questo capitolo viene presentato un caso abbastanza frequente, ovvero l'utilizzo di un *server* di posta in una rete non perennemente connessa ad Internet, ad esempio una rete con connessione *dial-up*. Vengono presentate le diverse soluzioni possibili e le problematiche che possono derivare da tale configurazione.

MTA su connessione dial-up

Utilizzo di un MTA non perennemente connesso ad Internet

Nel caso la propria rete non risulti perennemente connessa ad Internet, ad esempio perché si utilizza una connessione *dial-up*, è possibile comunque gestire la posta elettronica del proprio dominio mediante un MTA interno.

La posta in uscita generalmente non rappresenta un problema, in quanto è sufficiente che l'MTA interno abbia la possibilità di eseguire un collegamento SMTP verso l'esterno. La posta eventualmente spedita dagli utenti locali se il *server* non è collegato viene tenuta in coda e può essere spedita utilizzando il comando *sendmail* -q. Per minimizzare la durata delle connessioni *dial-up*, generalmente la posta in uscita non viene smistata direttamente verso i destinatari ma viene spedita in blocco ad uno *smarthost*, di solito l'MTA del *provider*.

Per quanto riguarda la posta in ingresso è possibile appoggiarsi ad un MTA esterno perennemente connesso, ad esempio quello del *provider* o di un fornitore di servizi, configurato come *mail exchanger* in modo da raccogliere la posta per il dominio.

Essa verrà poi trasferita al *server* interno quando esso si collega ad Internet richiedendo il *flushing* della coda di posta al *server* del *provider* mediante ETRN. Nel *server* del *provider* l'instradamento della posta verso il *server* aziendale può essere fornito da una *entry* nella *mailertable*:

- test.it;
- smtp:smtpinterno.test.it.

Questo metodo può essere utilizzato solamente se è possibile rendere visibile all'esterno la porta TCP 25 del *server* interno e se si dispone di un indirizzo IP statico. Si noti che non è necessario rendere visibile il *server* interno a tutto il mondo, bensì solamente al MTA del *provider*. Questo può essere fatto, a seconda della topologia della rete, agendo sul *firewall* aziendale oppure utilizzando le funzioni di *access list* di *Sendmail* (*Access Database* oppure TCP *Wrapper*).

Se il *server* non è visibile dall'esterno oppure non utilizza un indirizzo statico, non è possibile utilizzare SMTP, pertanto è necessario trovare una soluzione alternativa. Alcuni metodi che possono essere utilizzati allo scopo sono i seguenti:

- utilizzo del protocollo UUCP. Il protocollo UUCP permette a *server* interno di instaurare la connessione di scambiare in un'unica operazione sia la posta in ingresso che quella in uscita. Anche se attualmente viene poco utilizzato, si tratta probabilmente della alternativa migliore, in quanto efficiente, affidabile e perfettamente integrata in *Sendmail*. Inoltre permette di non dover esporre il *server* interno;
- *hosting* di una o più caselle *e-mail* POP3 presso il *provider*. Consiste nel creare gli utenti locali del proprio dominio direttamente nel MTA del *provider* e nello scaricare mediante POP3 le singole caselle di posta elettronica. Si tratta di una soluzione limitata e che non consente una gestione autonoma degli utenti;
- *hosting* di una casella POP3 *catchall* presso il *provider*. Consiste in una casella POP3, sul MTA del *provider*, in cui vengono inseriti tutti i messaggi destinati a qualunque utente del dominio. Sull'MTA del *provider* questo si ottiene inserendo nella *virtusertable* una linea del tipo @dominio.it nomecasella. Una volta prelevata la casella è necessario filtrarla in modo da riconsegnare la posta ai diversi utenti. Tale metodo non è consigliato, in quanto durante le operazioni possono venir perse alcune informazioni contenute nell'*header* dei messaggi;

Fetchmail

Fetchmail (**http://www.tuxedo.org/~esr/fetchmail/**) permette il prelievo di messaggi di posta elettronica da un *server* utilizzando i protocolli più diffusi POP2, POP3, RPOP, APOP, KPOP, IMAP, e ESMTP ETRN. I messaggi così prelevati possono essere salvati in formato *mailbox* oppure passati ad un MTA per essere inoltrati ad indirizzi locali o remoti. Trattandosi di un programma gestibile mediante linea di comando, *fetchmail* può essere utilizzato per automatizzare il prelievo dei messaggi.

La tipica applicazione di *Fetchmail* consiste nel prelievo di caselle di posta elettronica da un *server* remoto e nella consegna ad utenti locali. Poiché la consegna avviene mediante SMTP all'MTA locale, *fetchmail* si integra perfettamente con il sistema di posta di *UNIX* e permette di sfruttarne tutte le funzionalità (*forwarding*, *aliasing*, *procmail*, ...).

Fetchmail può essere eseguito da un utente creando nella propria *home directory* un *file* di configurazione .fetchmailrc contenente i dati necessari all'accesso alle caselle remote ed il nome dell'utente a cui consegnare la posta. Un esempio di *file* .fetchmailrc è il seguente:

```
poll pop.provider.net proto pop3 port 110
user andrea.verdi with pass pippo is andrea.verdi here
user mario.rossi with pass pluto is mrossi here
```

Il *file* di configurazione può essere creato mediante un *editor* oppure utilizzando il programma fetchmailconf, fornito col pacchetto o mediante una versione recente di linuxconf.

Fetchmail può anche essere utilizzato come un demone che, una volta lanciato mediante uno *script* di avvio in `/etc/rc.d`, rimane attivo nel sistema e si occupa di scaricare le caselle ad intervalli di tempo prestabiliti. È inoltre possibile, dopo aver prelevato una casella di tipo *catchall*, fare in modo che *fetchmail* si occupi di suddividerla opportunamente fra i vari utenti.

Problematiche derivanti dall'utilizzo di una connessione *dial-up*

Le operazioni da compiere per lo scambio della posta, ovvero

- lo scaricamento posta mediante *fetchmail* o UUCP o (*fetchmail* -a);
- lo svuotamento coda di spedizione verso lo *smarthost* (*sendmail* -q);

possono essere eseguite automaticamente ad intervalli prescelti mediante lo schedulatore di sistema (crontab). Questa tecnica prende il nome di *polling*. In un sistema *dial-up* bisogna tuttavia fare attenzione a non utilizzare un intervallo inferiore al *timeout* della connessione ad Internet, per evitare che questa rimanga perennemente attiva. Al contrario, un intervallo di tempo troppo lungo causerà dei problemi di lentezza nella ricezione dei messaggi.

Un altro aspetto a cui è necessario porre attenzione consiste nell'evitare che *Sendmail* faccia traffico indesiderato verso l'esterno, che potrebbe causare connessioni indesiderate da parte del *router*.

Per evitare problemi è necessario adottare due cautele:

- fare in modo che i messaggi destinati all'esterno non vengano spediti immediatamente ma vengano invece tenuti in coda. Per far ciò si devono marcare i *mailer* smtp, esmtp e *relay* come "*expensive*" (costosi);
- evitare che *Sendmail* effettui delle *query* al DNS. Queste avvengono ad esempio in occasione delle verifiche (*check_mail*, *check_rcpt*) effettuate durante la ricezione mediante SMTP dei messaggi generati dai MUA interni.

Le precauzioni da adottare per eliminare queste cause di traffico indesiderato sono spiegate nella FAQ di *Sendmail*.

Nota: un'altra possibile causa di richieste inutili al DNS è l'impossibilità da parte del *server* di risolvere localmente (mediante `/etc/hosts` oppure mediante un DNS sulla rete interna) gli indirizzi dei *client* locali. Questo problema è stato affrontato in forma generale nella *Unit* relativa ai servizi di rete.

Accesso remoto con Windows 2000

Franco Callegati
Paolo Presepi
Riccardo Gori
7.3.1 (Installare e configurare applicazioni client/server su un server quali: e-mail, FTP, Web, sistemi di messaggistica, chat, eccetera)

Default e Politiche Multiple

Default

Affinché sia possibile avere accesso remoto deve esistere sempre almeno una Politica di Accesso Remoto. Per garantire ciò in un *server* di accesso remoto è sempre presente Politica di Accesso Remoto di *Default* che è sufficiente per gestire la maggior parte dei casi.

Tale Politica di Accesso Remoto di *Default* viene applicata a tutti quei tentativi di connessione che non soddisfano le condizioni di nessun altra politica definita.

La Politica di Accesso Remoto di *Default* è denominata *Allow access if dial-in permission is enabled* e viene creata automaticamente quando viene installato *Routing and Remote Access*.

Di seguito sono specificate le impostazonii di tale politica:

- Condizione: Qualsiasi Giorno e Qualsiasi Ora.
- Permesso: *Deny access*.
- Profilo: Nessuno.

Modalità Nativa (nega tutti gli accessi a meno che l'*account* sia impostato ad *Allow*)

In un dominio in 'Modalità Nativa', settando i permessi di *dial-in* dell'*account* utente come '*Control access through Remote Access Policy*' la Politica di Accesso Remoto di *Default* ha come effetto quello di rifiutare qualsiasi connessione. Tuttavia se i permessi di *dial-in* dell'*account* utente sono settati come *Allow access* allora il tentativo di connessione di tale utente sarà accettato.

Modalità *Mixed* (la politica di *Default* viene sovrascritta)

Invece, in un dominio in 'Modalità Mista' la 'Politica di Accesso Remoto di *Default*' viene sempre sovrascritta dalle proprietà di *dial-in* specificate per l'*account* utente, poiché l'opzione '*Control access through Remote Access Policy*' non è disponibile sui controllori di dominio di un dominio che viene eseguito in 'Modalità Mista'.

Durante la conversione da 'Modalità Mista' a 'Modalità Nativa' il nelle proprietà di *dial-in* dell'*account* utente il '*Deny access*' diventa '*Control access through Remote Access Policy*' ed il '*Allow access*' resta '*Allow access*'.

Dunque, per quanto detto nel caso siano presenti più Politiche di Accesso Remoto ogni tentativo di connessione viene giudicato in base alla prima politica di cui sono soddisfatte le condizioni. Nel caso in cui la richiesta di connessione non soddisfi le condizioni di nessuna politica, resta la Politica di Accesso Remoto di *Default* le cui condizioni sono sempre soddisfatte e dunque la richiesta viene accettata solo se nelle proprietà di *dial-in* dell'utente che sta richiedendo la connessione sta specificato *Allow access*.

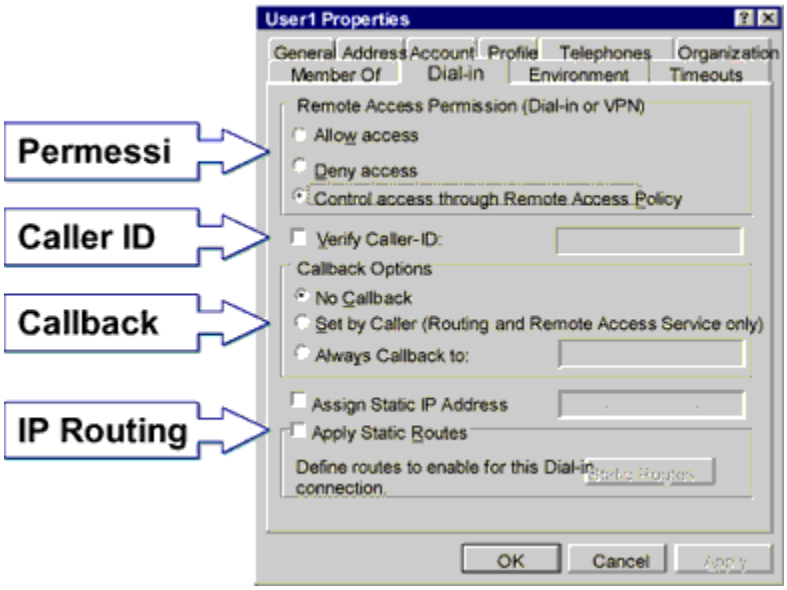
Creazione delle Politiche di Accesso Remoto

Dunque le Politiche di Accesso Remoto permettono di gestire le richieste di accesso remoto tramite la definizione di regole che possono essere molto semplici o estremamente complesse, in base a quelle che sono le esigenze della nostra organizzazione.

Gli elementi di base nella definizione delle Politiche di Accesso Remoto sono dunque:

- I permessi di *dial-in* nelle proprietà dell'*account* utente in *Active Directory*.
- La creazione di una Politica di Accesso Remoto in *Routing e Remote Access* con le relative condizioni e permesso.
- Il Profilo associato ad ogni Politica di Accesso Remoto.

Proprietà di Accesso Remoto dell'Account



Per configurare le proprietà di *dial-in* di un utente su un *server stand-alone* utilizzare lo strumento '*Computer Management*' presente in *Start|Programs|Administrative Tools* e nel contenitore '*Local Users and Groups*' cercare l'utente desiderato e facendo clic con il tasto destro selezionare '*Properties*'.

Invece, per configurare le proprietà di *dial-in* di un utente in un dominio *Active Directory* utilizzare lo strumento '*Active Directory Users and Computers*' presente in *Start|Programs|Administrative Tools*, cercare l'utente desiderato e facendo clic con il tasto destro selezionare '*Properties*'e scegliere la scheda '*Dial-In*'.

Utilizzando la sezione '*Remote Access Permission (Dial-In or VPN)*' è possibile negare o permettere l'accesso remoto esplicitamente o, selezionando '*Control access through Remote Access Policy*', fare riferimento al permesso associato alla politica di accesso remoto di cui tale utente soddisfa le condizioni al momento dell'accesso. Quest'ultima opzione è disponibile solamente in un dominio in 'Modalità Nativa'.

Utilizzando il *check Verify Caller-ID* è possibile fare in modo che il *server* verifichi che il numero telefonico del chiamante coincida con quello ivi specificato.

La sezione *Callback Options* permette di definire se il *Callback* venga o meno abilitato (*No Callback*) e nel caso venga abilitato se il numero a cui il *server* effettua la richiamata è stabilito dall'utente (*Set By Caller*) o una volta per tutte dall'amministratore (*Always Callback To*).

Infine è possibile assegnare all'utente remoto un indirizzo IP statico (*Assign Static IP Address*) e/o configurargli delle *routes* statiche (*Apply Static Routes*).

Condizioni di una Politica di Accesso Remoto

Se la richiesta...

- Avviene tra le 8 a.m. - 5 p.m. di Lunedì-Venerdì.
- Proviene da un indirizzo IP che corrisponde a: 192.168.*.*.
- È di un utente del gruppo Vendite.

La Condizione di una Politica di accesso remoto è costituita da tutta una serie di attributi con relativi valori che

verranno comparati con i corrispondenti attributi della richiesta di accesso remoto. Affinché una richiesta soddisfi una condizione, devono esserci corrispondenza con i valori di tutti gli attributi che costituiscono la condizione.

Di seguito elenchiamo quelle che sono le condizioni che è possibile specificare:

- *NAS IP Address*. Una stringa che identifica l'indirizzo IP del *network access server* (NAS).
- *Service Type*. Il tipo di servizio RADIUS richiesto.
- *Framed Protocol*. Il protocollo dei pacchetti in entrata (PPP, *AppleTalk*, SLIP, *Frame Relay*, e X.25).
- *Called Station ID*. Una stringa che identifica il numero telefonico del NAS.
- *Calling Station ID*. Una stringa che identifica il numero di telefono del chiamante.
- *NAS Identifier*. Una stringa che identifica il NAS da cui proviene la richiesta.
- *NAS Port Type*. Il tipo di media utilizzato dal chiamante (analogico, ISDN, e VPN).
- *Day and Time Restrictions*. Il giorno della settimana e l'ora del giorno in cui viene effettuata la connessione.
- *Client IP Address*. Una stringa di caratteri che identifica l'indirizzo del *client* RADIUS.
- *Client Vendor*. Il produttore del NAS che richiede l'autenticazione.
- *Client Friendly Name*. Una stringa di caratteri che identifica il nome del *client* RADIUS che richiede l'autenticazione.

Windows Groups. Il nome del gruppo di *Windows* 2000 cui l'utente che effettua la chiamata appartiene. Non esiste nessuna condizione per controllare il nome dell'utente.

- Per creare una Politica di Accesso Remoto con le relative condizioni ed il profilo associato utilizzare il *tool Routing and Remote Access* presente in *Start|Programs|Administrative Tools* e:
- Cliccare con il tasto destro su *Remote Access Policies* e selezionare *New Remote Access Policy*.
- Utilizzando il *wizard Add Remote Access Policy* inserire il nome in *Policy friendly name* selezionare *Next*.
- Per configurare le condizioni, per ognuna di esse, cliccare *Add*.
- In *Select Attribute* selezionare l'attributo da inserire e cliccare *OK*. Nella corrispondente finestra di dialogo inserire le informazioni richieste da quell'attributo e cliccare su *OK*.
- Una volta inserite tutte le condizioni cliccare *Next*.
- Per garantire l'accesso a chi soddisfa le condizioni selezionare *Grant remote access permission*, mentre per negarlo selezionare *Deny remote access permission*.
- A questo punto è possibile creare un Profilo o, cliccando *Finish* avere una Politica di Accesso Remoto senza specificare nessun Profilo.

Profilo di una Politica di Accesso Remoto

Il Profilo è...

- Tempo massimo di connessione 90 minuti.
- 4 linee multilink.
- Richiesta criptazione IP Sec.

Il Profilo relativo ad una Politica di Accesso Remoto permette di specificare quali sono le caratteristiche che devono essere soddisfatte dalla connessione per tutta la sua durata, pena la disconnessione immediata.

Per configurare il Profilo per una Politica di Accesso Remoto utilizzare il *tool Routing and Remote Access* presente in *Start|Programs|Administrative Tools* e:

- Fare doppio click sul contenitore *Remote Access Policies*.
- Selezionare la Politica di Accesso Remoto desiderata.
- Cliccare con il tasto destro e scegliere *Properties*.
- Cliccare su *Edit Profile* ed utilizzare la finestra di dialogo *Edit Dial-in Profile* come di seguito descritto:

- *Dial-in Constraints*. Permette di determinare la durata massima della connessione, la durata massima del periodo di inattività, il giorno, l'ora, il tipo di media (ISDN, VPN o altro) che sono permessi.
- IP. Permette di filtrare pacchetti IP in ingresso ed in uscita in base alle loro caratteristiche.
- Multilink. Permette di configurare le modalità del Multilink e di BAP (*Bandwith Allocation Protocol*).
- *Authentication*. Permette di definire i protocolli di autenticazione autorizzati.
- *Encryption*. Permette di definire il livello di cifratura richiesto e le modalità di utilizzo.
- *Advanced*. Permette di configurare parametri addizionali tipo quelli inerenti RADIUS.

Server di Accesso Remoto

La modalità più semplice per permettere l'accesso alla rete ad utenti remoti è quella di definire su un *server* un *server Dial-Up* di accesso remoto.

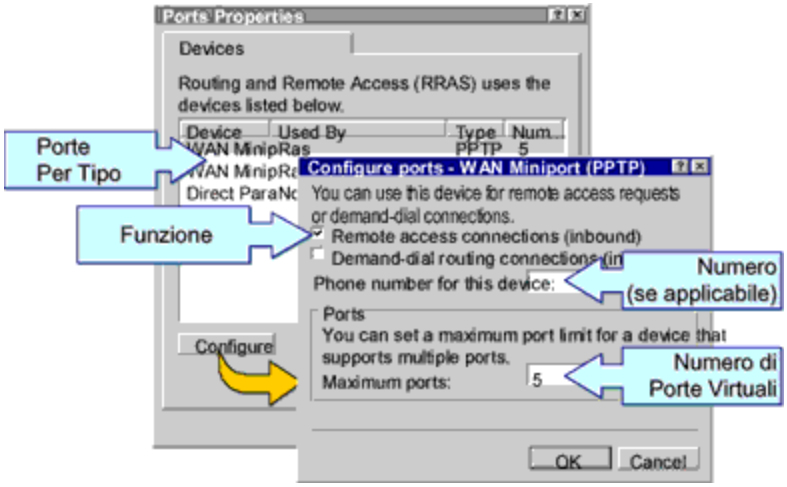
Per configurare ed attivare tale *server* eseguire i seguenti passi:

- Verificare la compatibilità dei dispositivi *hardware* (modem, adattatore ISDN o quant'altro) che si intendono utilizzare con *Windows* 2000, usando la *Hardware Compatibility List* (HCL) e, in caso di riscontro positivo, procedere alla loro installazione.
- Utilizzare lo strumento *Routing and Remote Access* contenuto in *Start|Programs|Administrative Tools* e cliccando con il tasto destro sul nome del *server* selezionare *Configure and Enable Routing and Remote Access*.
- Completare il *wizard* secondo quanto segue:
  - *Common Configurations*: *Remote access server*.
  - *Remote Client Protocols*: Se vi sono elencati tutti i protocolli che si intende utilizzare selezionare *Next*, altrimenti procedere alla loro installazione.
  - *IP Address Assignment*: Selezionare un metodo di installazione degli indirizzi IP ai *client*. Se si decide di definire sul *server* RRAS un *range* di indirizzi, piuttosto che appoggiarsi ad un *server* DHCP, occorre specificare tale *range* di indirizzi.
  - *Managing Multiple Remote Access Servers*: Selezionare se intende o meno utilizzare un *server* RADIUS. In caso di risposta positiva bisogna specificare il nome del *server* RADIUS, il nome di un eventuale *server* RADIUS di *backup* e la *password* da utilizzare per dialogare con tali *server*.
- Configurare eventualmente politiche di accesso remoto, e tutti i settaggi inerenti le modalità di autenticazione e cifratura.

Quando il *server* di accesso remoto *dial-up* parte la prima volta, *Windows* 2000 rileva automaticamente tutti i dispositivi *hardware* (modem, adattatori ISDN ...) installati e configura una porta modem per ognuno di essi.

Windows 2000 provvede anche a creare automaticamente una porta per ogni connessione ad una porta seriale o parallela che dovesse rilevare.

È possibile, da questo momento in poi, modificare la configurazione di tali porte utlizzando il contenitore *Ports* presente nel lo strumento *Routing and Remote Access* contenuto in *Start|Programs|Administrative Tools*:



- Accedere alle proprietà del contenitore *Ports*.
- In *Ports Properties* selezionare una device e quindi *Configure*.
- In *Configure Device* selezionare *Remote access (inbound)* per abilitare le connessioni in ingresso.
- Nel caso in cui si stia configurando una porta modem inserire in numero di telefono in *Phone number of this device*.
- Selezionare *OK* in *Configure Device* e *Ports Properties*.

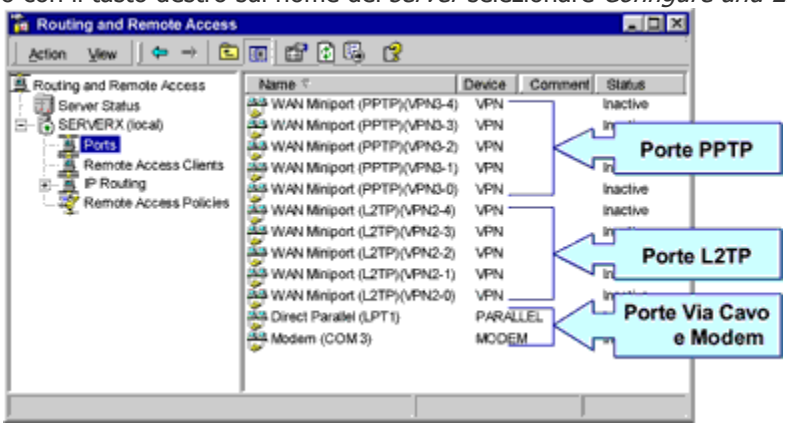
Server VPN

Tra gli svantaggi di un *server* di accesso remoto di tipo *dial-up* ricordiamo essenzialmente il problema dei costi dovuto alla necessità di effettuare chiamate spesso extraurbane ed alla necessità di disporre di tante connessioni fisiche quanti sono i *client* che si vuole accedano contemporaneamente al *server* da remoto.

Utilizzare lo strumento *Routing and Remote Access* contenuto in *Start|Programs|Administrative Tools* e cliccando con il tasto destro sul nome del *server* selezionare *Configure and Enable Routing and Remote Access*.

Completare il *wizard* secondo quanto segue:

- Verificare la compatibilità dei dispositivi *hardware* (modem, adattatore ISDN o quant'altro) che si intendono utilizzare con *Windows* 2000, usando la ***Hardware Compatibility List*** (HCL) e, in caso di riscontro positivo, procedere alla loro installazione.
- Utilizzare lo strumento *Routing and Remote Access* contenuto in *Start|Programs|Administrative Tools* e cliccando con il tasto destro sul nome del *server* selezionare *Configure and Enable Routing and Remote Access*.



- Completare il *wizard* secondo quanto segue:

    - *Common Configurations*: *Virtual private network (VPN) server*.
    - *Remote Client Protocols*: Se vi sono elencati tutti i protocolli che si intende utilizzare selezionare *Next*, altrimenti procedere alla loro installazione.
    - *IP Address Assignment*: Selezionare un metodo di installazione degli indirizzi IP ai *client*. Se si decide di definire sul *server* RRAS (Routing and Remote Access Services) un *range* di indirizzi, piuttosto che appoggiarsi ad un *server* DHCP (Dynamic Host Configuration Protocol), occorre specificare tale *range* di indirizzi.
    - *Managing Multiple Remote Access Servers*: Selezionare se intende o meno utilizzare un *server* RADIUS. In caso di risposta positiva bisogna specificare il nome del *server* RADIUS, il nome di un eventuale *server* RADIUS di *backup* e la *password* da utilizzare per dialogare con tali *server*.
- Configurare eventualmente politiche di accesso remoto, e tutti i settaggi inerenti le modalità di autenticazione e cifratura.

Quando il *server* di accesso remoto VPN parte per la prima volta, *Windows* 2000 crea e configura automaticamente 128 porte PPTP (*Point to Point Tunneling Protocol*) *ports* e 128 porte L2TP (*Layer Two Tunneling Protocol*) *ports*. Il numero di porte virtuali disponibili non è limitato da considerazioni legate al numero di dispositivi *hardware* bensì semplicemente da considerazioni relative alle *performance*.

È possibile, da questo momento in poi, modificare la configurazione di tali porte utlizzando il contenitore *Ports* presente nel lo strumento *Routing and Remote Access* contenuto in *Start|Programs|Administrative Tools*:

- Accedere alle proprietà del contenitore *Ports*.
- In *Ports Properties* selezionare una device VPN, *WAN Miniport* (PPTP) o *WAN Miniport* (L2TP), quindi *Configure*.
- In *Configure Device* selezionare *Remote access (inbound)* per abilitare le connessioni in ingresso.
- Opzionalmente, è possibile aumentare o diminuire il numero di porte virtuali disponibili.
- Selezionare *OK* in *Configure Device* e *Ports Properties*.

Politiche di Accesso Remoto

Le **Politiche di Accesso Remoto** ci permettono di controllare in maniera molto dettagliata chi si connette, come si connette, quando si connette e, una volta connesso, quali sono le caratteristiche della connessione.

Conoscere come sono fatte e quali sono i criteri che regolamentano la loro applicazione, ci permette di utilizzare tale strumento in maniera molto proficua.

La prima osservazione da fare riguarda il fatto che le informazioni relative alla configurazione delle Politiche di Accesso Remoto non sono memorizzate in *Active Directory*, come ci si potrebbe aspettare, bensì localmente al *server* per il quale vengono definite.

Una politica di accesso remoto consiste di tre componenti:

- **Condizioni**. Le condizioni sono un insieme di parametri, come ad esempio la data e l'ora, l'appartenenza a gruppi, il numero di telefono o l'indirizzo IP del chiamante, che devono coincidere con i parametri del *client* che sta effettuando la chiamata. La prima politica di accesso remoto le cui condizioni coincidono con quelle del *client* è quella che viene applicata. Dunque deve esistere almeno una politica di accesso remoto, vale la prima di cui il *client* soddisfa le condizioni (per cui il loro ordine è significativo) e se non sono soddisfatte le condizioni di nessuna politica la connessione viene rifiutata.
- **Permesso**. Nel caso in cui vengano soddisfatte le condizioni relative ad una politica allora la connessione viene permessa o negata in base ad una combinazione del permesso specificato nelle proprietà dell'*account* utente e del permesso specificato nella politica.Questo permette di avere una certa flessibilità specificando una politiche che permette o nega l'accesso ad un certo gruppo di utenti con certe caratteristiche e

specificando poi l'eccezione per particolari utenti che pure appartengono a quel gruppo.
- **Profilo**. Quando le condizioni di una politica sono soddisfatte ed il permesso concede la connessione, viene applicato a tale connessione un profilo. Nel profilo sono contenute quelle che devono essere le caratteristiche della connessione (per esempio la durata) e la loro violazione provoca la terminazione della connessione.

Logica di Valutazione

Vediamo innanzitutto qual'è la logica seguita nella valutazione delle Politiche di Accesso Remoto. Tale logica giudica le richieste di connessione mixando le condizioni, i permessi della politica e dell'utente che richiede la connessione e le impostazionii specificate nel profilo.

La logica di valutazione delle Politiche di Accesso Remoto è la seguente:

- I parametri del *client* che sta effettuando la chiamata vengono confrontati con le condizioni specificate nelle Politiche di Accesso Remoto
  - Se essi non coincidono con le condizioni di nessuna Politiche di Accesso Remoto allora la chiamata viene rifiutata.
  - Se essi coincidono con le coindizioni di una Politica di Accesso Remoto, allora tale politica viene usata per determinare l'accesso.
- Innanzitutto vengono controllate la proprietà di *dial-in* relative all'*account* utente che sta richiedendo la connessione:
  - Se essa è settata a *Deny access*, la connessione viene negata.
  - Se essa è settata a *Allow access*, la connessione viene accettata e viene applicato il permesso della politica.
  - Se essa è settata a *Control access through Remote Access Policy* (opzione disponibile solo nei domini in Modalità Nativa), viene applicato il permesso relativo alla politica. Se esso permette l'accesso la connessione viene accettata e viene applicato il permesso della politica, altrimenti viene rifiutata.
- Nel caso in cui la connessione sia stata accettata i parametri del richiedente vengono continuamente confrontati con quelli del profilo e non appena non dovesse essere più riscontrata corrispondenza, la connessione viene terminata.

Samba

Franco Callegati
Paolo Presepi
Riccardo Gori
7.1.7 (Installare e connettere più server (anche basati su piattaforme diverse)), 7.3.1 (Installare e configurare applicazioni client/server su un server quali: e-mail, FTP, Web, sistemi di messaggistica, chat, eccetera)

Integrazione UNIX-NT

Questa unità didattica descrive le modalità di integrazione di un sistema *UNIX* all'interno di una rete in tecnologia *Windows*. In particolare viene presentato il *software* Samba, che implementa in *UNIX* la parte *server* del protocollo **SMB** (*Session Message Block*).

SMB, (utilizzato anche con il significato di *Server Message Block*) è un protocollo utilizzato per condividere risorse *hardware* (*file*, stampanti, periferiche in genere) e risorse *software* di un sistema di elaborazione (*pipe*, *mailbox*, eccetera).

L'idea del protocollo SMB nasce in ambito IBM verso la metà degli anni '80 e successivamente è stata ripresa da *Microsoft* fin dal 1987. Il protocollo è stato successivamente sviluppato ulteriormente da *Microsoft* e altri.

I dati SMB possono essere incapsulati e trasportati in datagrammi TCP/IP, oppure NetBEUI e IPX/SPX.

Al fine di collegare quanto analizzato nei moduli didattici in cui sono state affrontate le architetture e le infrastrutture per trasmissione dati ed i protocolli della famiglia TCP/IP, si include qui uno schema grafico che evidenzia la posizione del protocollo SMB nell'ambito dell'architettura di comunicazione.

Samba

Samba è una implementazione del protocollo *Session Message Block*, utilizzato dalle reti *Windows*. Samba è disponibile come *software Open Source* per molti sistemi *UNIX*. Oltre alla distribuzione ufficiale, esistono anche alcuni prodotti commerciali derivati da Samba.

Attualmente il pacchetto supporta solamente il protocollo SMB incapsulato all'interno di pacchetti IP (in *Windows* è infatti possibile utilizzare altri protocolli di trasporto come NetBEUI o IPX/SPX). Rispetto ad un *server* NT vi sono inoltre alcune differenze nei meccanismi di *locking* dei *file*, di gestione della autenticazione, una diversa gestione dei permessi ed un diverso formato nei *file* di testo.

Samba implementa in modo efficiente e sufficientemente completo la parte *server* del protocollo, e può essere utilizzato per fornire da un *server UNIX* i seguenti servizi ad una rete di *client Windows*:

- Condivisione di *directory* da *UNIX* verso PC.
- Condivisione di servizi di stampa verso PC.
- *Primary Domain Controller*.
- *Nameserver* compatibile NetBIOS utilizzabile come *master browser* e *server WINS*.
- Supporto per *Common Internet Filesystem* (CIFS).
- Gestione grafica mediante *browser Web* delle principali funzionalità (*Swat*).

La parte *client* invece comprende:

- un *client* a linea di comando per l'accesso a volumi condivisi (*smbclient*) o la stampa su stampanti *Windows*; la maggior parte delle distribuzioni di *Linux* è già predisposta per stampare attraverso *smbclient* su una stampante remota residente su un *server* di rete *Windows*.
- Un comando (*smbtar*) per eseguire il *backup* di PC via rete.
- I programmi per montare su *Linux* volumi condivisi da un *server Windows*; il supporto necessario da parte del *kernel* è compreso nella distribuzione standard di *Linux*.

I principali componenti di Samba sono i seguenti:

- *smbd*: *server* SMB;
- *nmbd*: *nameserver* NetBIOS (rfc1001/1002);
- *smbclient*, *smbprint*, *smbmount*, *smbtar*: *client* SMB;
- *smbpasswd*, *convert_smbpasswd*, *smbadduser*, *mksmbpasswd.sh*, *testparm*, *smbstat*, *nmbdlookup*: programmi di amministrazione;
- *swat*: interfaccia grafica (*web based*).

SAMBA versione 2.2

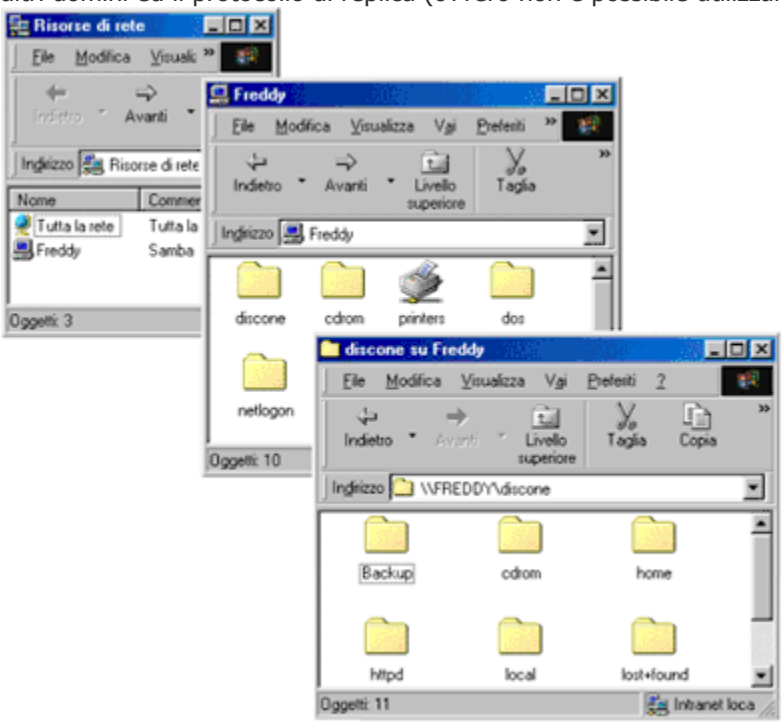
La versione 2.2 di Samba, a cui faremo riferimento, ha introdotto le seguenti novità:

- **Stampanti**. È stata introdotta la possibilità di fornire automaticamente i *driver* delle stampanti ai *client*. I *driver* devono essere prima installati ed assegnati alle stampanti sul *server* Samba mediante l'*Add Printer Wizard* (APW) di *Windows* oppure mediante il *software Imprints* (*Installation Manager of Printer driver*

Retreival and Installation for Samba), disponibile su **http://imprints.sourceforge.net/**.

- ***Access Control Lists* (ACL)**. Viene effettuato una mappatura fra le ACL di NT e le ACL fornite dal *filesystem UNIX*. Tale funzione viene supportata limitatatmente ad alcuni sistemi operativi e *filesystem* (esempio: *Linux* + ext2). Le ACL sono modificabili direttamente tramite la *shell* grafica di *Windows WinNT Explorer Security Tab*.
- ***Locking***. Vengono risolte le limitazioni delle versioni precedenti, offrendo diversi meccanismi di *locking* dei *file*. Tali funzioni dovrebbero essere utilizzati con attenzione, in quanto potrebbero avere conseguenze significative sia sull'integrità dei *file* che sulle prestazioni del sistema.
- **Domini *Windows***. È possibile utilizzare Samba per gestire le funzionalità tipiche di un *server* di rete NT:
  - liste utenti per *Windows* 9x;
  - supporto completo dei *client* NT 4.0 SP3+;
  - supporto completo dei *client Windows* 2000 in modalità *legacy*.

  Le *System Policy* funzionano in modo analogo a quanto avviene nei domini NT 4.0, eccetto i gruppi, che verranno supportati in una prossima versione di Samba. Non sono inoltre supportate le *Trust Relationship* con altri domini ed il protocollo di replica (ovvero non è possibile utilizzare Samba come BDC).



Installazione di Samba

Il programma è già presente di serie in molte distribuzioni di *Linux* ed in alcuni sistemi *UNIX*. Nel caso non fosse presente, è possibile prelevare i sorgenti oppure i binari precompilati da **http://www.samba.org/**.

È possibile avviare Samba sia come *server* a sé stante, che mediante il supervisore `inetd/xinetd`. Nel primo caso è necessario creare uno *script* `rc` di avvio che faccia partire come demoni (opzione -D) i due programmi *smbd*, che implementa il protocollo vero e proprio, e *nmbd*, che si occupa della risoluzione dei nomi:

```
smbd -D
nmbd -D
```

Volendo utilizzare `inetd` si devono invece inserire in `/etc/inetd.conf` due linee simili alle seguenti:

```
netbios-ssn stream tcp nowait root /usr/sbin/smbd smbd
netbios-ns dgram udp wait root /usr/sbin/nmbd nmbd
```

L'utilizzo di `inetd` permette eventualmente di proteggere ulteriormente l'accesso ai servizi rispetto alle *access list* di Samba utilizzando un *TCP wrapper*.

Il protocollo NetBIOS sopra TCP/IP utilizza le seguenti porte

- UDP/137: risoluzione dei nomi e registrazione (*nmbd*);
- UDP/138: *browsing* e annuncio (*nmbd*);
- TCP/139: condivisione *file* e stampanti (*smbd*).

che devono essere opportunamente definite in `/etc/services`:

```
netbios-ns 137/tcp nbns # NetBIOS Name Service
netbios-ns 137/udp nbns
netbios-dgm 138/tcp nbdgm # NetBIOS Datagram Service
netbios-dgm 138/udp nbdgm
netbios-ssn 139/tcp nbssn # NetBIOS session service
```

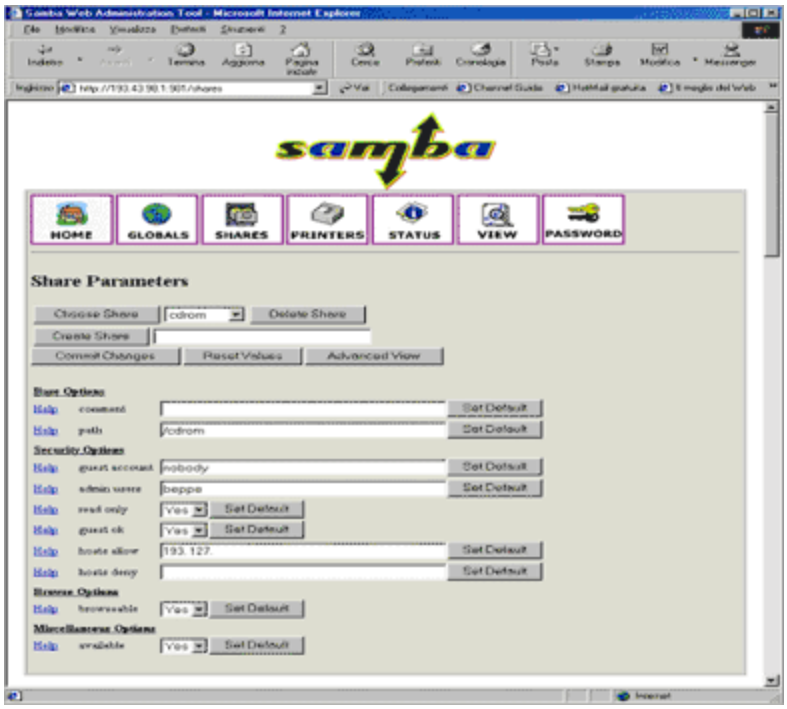
La configurazione di Samba avviene mediante il *file* `smb.conf` (generalmente in `/etc`) e può essere effettuata mediante la modifica diretta del *file*, che ha formato simile ad un *file* INI di *Windows*. Una volta modificato il *file* è necessario far ripartire il servizio oppure inviare un segnale di tipo HUP sia a *smbd* che a *nmbd*.

È buona norma, quando si apportano delle modifiche a `smb.conf`, verificare la correttezza della configurazione utilizzando il comando *testparm*.

Una volta attivato il servizio, è possibile tenerne sotto controllo il funzionamento mediante i *file* di log (generalmente in `/var/log/samba/`) e mediante il programma `smbstatus`.

Swat

Se è stato installato il programma *Swat*, è possibile accedere ad una interfaccia grafica collegandosi con un *browser* alla porta 901 del *server* (http://indirizzo:901/). L'utilizzo di *Swat* non è incompatibile con la modifica diretta di `smb.conf`.



Per installare il programma è sufficiente aggiungere a `/etc/services` la linea

```
swat 901/tcp
```

e configurare opportunamente `inetd`

```
swat stream tcp nowait.400 root /usr/sbin/swat swat
```

Anche in questo caso è consigliabile utilizzare il *TCP wrapper* per limitare l'accesso al servizio.

Oltre a *swat* esistono anche altre interfacce grafiche che aiutano la configurazione di Samba. Volendo effettuarla direttamente da un *client Windows* si può utilizzare il programma **SmbEdit**.

Schema di autenticazione

Quando l'utente di un *client* accede ad una condivisione, Samba tenta di associarlo ad un utente *UNIX*, con i cui permessi verranno poi effettuati gli accessi alle risorse *UNIX* legate alla condivisione.

Samba utilizza diverse tecniche per determinare l'utente associato alla condivisione in base al nome della stessa. Nel caso non fosse possibile determinare l'utente, è possibile permettere comunque l'accesso utilizzando i permessi di un utente fittizio (*guest*). Questo è utile ad esempio per creare delle condivisioni pubbliche.

Come questa associazione venga effettuata dipende dallo schema di autenticazione prescelto. In Samba sono possibili i seguenti metodi di autenticazione:

- ***Share* (*security* = *share*)**. È lo schema di autenticazione utilizzato nelle versioni di Samba precedenti la 2.0. L'utente *UNIX* viene associato alla condivisione in fase di definizione della stessa.
- ***User* (*security* = *user*)**. È il metodo di autenticazione predefinito a partire dalla versione 2.0 di Samba. L'utente deve effettuare il *logon* per venire associato ad un corrispondente utente *UNIX* ed ottenere accesso alla risorsa condivisa.

- ***Server* (*security = server*)**. In questo schema si utilizza un altro *server* SMB (ad esempio un *server* NT) per l'autenticazione. Se questa fallisce, allora viene utilizzato il metodo "*security = user*".
- ***Domain* (*security = domain*)**. Valida gli utenti allo stesso modo di *Windows* NT, usando un PDC o un BDC. Richiede che i *client* vengano aggiunti al *domain* mediante il comando *smbpasswd*. Accetta solamente *encrypted password*.

Encrypted password

Allo scopo di non fare transitare *password* in chiaro attraverso la rete, a partire dalla versione 2.0 di Samba vengono utilizzate per *default* delle *password* codificate. Esse vengono mantenute in un *file* separato e sono codificate in modo diverso rispetto a quelle degli utenti *UNIX* in `/etc/passwd`. Per facilitare la transizione è prevista la creazione automatica della *password* codificata nel caso l'utente esegua il *logon* in chiaro.

Nel caso non si desideri utilizzare le *password* codificate, si deve specificare in `smb.conf` l'opzione *encrypt passwords* = No. Tuttavia le *password* in chiaro non sono supportate dai *client Windows* recenti e probabilmente non verranno supportate nelle prossime versioni di Samba.

Nota: volendo, è possibile far accettare ai nuovi *client* le *password* in chiaro utilizzando una *entry* speciale nel *registry*. I *file* `.reg` necessari sono inclusi per convenienza nella *directory* `doc` di Samba.

File di configurazione

Sezione	Descrizione
[*global*] *security=share* *workgroup=WORKGROUP* *netbios name=SERVER*	Parametri globali e valori di *default*
[*homes*] *comment=Home Directories* *read only*=No *create mask*=0755 *browseable*=No	*Home director* degli utenti *UNIX*: *//SERVER/USERNAME*
[*printers*] *comment*=lista stampanti *path*=/var/*spool*/samba *guest ok=YES* *print ok=YES*	Condivisione stampanti di sistema: *//SERVER/PRINTERNAME*
[disco2] *comment*=Disco2 *path=/disk2* *read only*=No *guest ok*=No [cdrom] *comment*=CD-ROM *path*=/mnt/cdrom *read only=Yes* *guest ok=Yes* [tmp]	Definizione esplicita di servizi e condivisioni

path=/tmp *read only*=No *create mask*=0755 *guest ok*= *Yes*	

Il *file* di configurazione `/etc/smb.conf` ha un formato simile ad un *file* INI di *Windows* ed è suddiviso in due tipologie di sezioni:

- sezioni speciali
  - [*global*];
  - [*homes*];
  - [*printers*];
- sezioni che definiscono servizi
  - [nomeservizio];

Sezioni speciali

- **[*global*]**. Le opzioni definite in questa sezione si applicano al *server* nella sua interezza. Generalmente viene utilizzato per il *tuning* del *server* (*dead time*, *keep alive*, *widelinks*, *getwd cache*, *socket options*, ...) e per definire i valori di *default* per i servizi.
- **[*homes*]**. È una sezione opzionale, che, se definita, permette di associare automaticamente una condivisione del tipo `//nomeserver/utente` con la *home directory* e con i permessi dell'utente *UNIX* corrispondente. Ad esempio, tentando di aprire `//nomeserver/rossi`, se non è stato definito esplicitamente un servizio di nome [rossi], viene condivisa la *home directory* per l'utente *UNIX* rossi. Questo permette di non dover definire e mantenere aggiornate in `smb.conf` tutte le condivisioni corrispondenti ai singoli utenti. Specificando nella definizione del servizio il valore *browseable* = *Yes*, è possibile rendere sfogliabile la lista degli utenti.
- **[*printers*]**. Definendo questa sezione, è possibile condividere automaticamente tutte le stampanti presenti nel *server UNIX* senza dover definire le singole condivisioni. È possibile rendere sfogliabile la lista delle stampanti specificando nella definizione del servizio *browseable* = *Yes*.

La regola di ricerca per associare il nome di una condivisione (esempio: `//nomeserver/nomevol`) con la corrispondente risorsa nel sistema è la seguente:

- viene cercato fra i servizi definiti se ne esiste uno con etichetta nomevol;
- se non viene trovato, ed è definita la sezione [*homes*], allora viene verificato se esiste l'utente *UNIX* nomevol;
- se non viene trovato, ed è definita la sezione [*printers*], allora viene verificato se nel sistema esiste una stampante di nome nomevol.

Esempio di definizione di un servizio

```
[nomeservizio]
comment = sample share
path = /disk2/test
writable = yes
printable = no
public = no
valid users = name1, name 2, @group2
invalid users = name3, name5, @group1
write list = name1, @group2
force create mode = 0660
force directory mode = 0775
```

Questo esempio di definizione di servizio mostra come rendere la *directory* `/disk2/test` del *server UNIX* accessibile dai *client Windows* come `//nameserver/nomeservizio`. Per far ciò è necessario creare una nuova sezione in `smb.ini` col nome che si vuole assegnare alla condivisione. Seguono un commento opzionale e il percorso della *directory* che si desidera condividere.

La riga *printable*=no indica che non si tratta di una stampante, mentre *writable* = *yes* identifica un servizio a cui è possibile accedere anche in scrittura.

La riga *public*=no restringe la leggibilità della condivisione ai soli utenti definiti come *valid users*, ad eccezione di quelli definiti come *invalid users*, mentre *write list* regolamenta l'accesso in scrittura.

Le liste possono essere costituite da un insieme di *username* di utenti *UNIX* oppure da scritture del tipo @gruppo, che identificano tutti gli utenti appartenenti ad un determinato gruppo *UNIX*.

Le due opzioni *force create mode* e *force directory mode* controllano i permessi assegnati dal sistema ai nuovi *file* ed alle *directory* che vengono create da Samba. La prima riga forza il permesso 0660 per ogni nuovo *file* che viene creato, in modo da permettere agli altri utenti dello stesso gruppo *UNIX* del proprietario di modificare il *file*. La seconda controlla invece il permesso da assegnare alle *directory* (775 permette la lettura e scrittura a tutti gli utenti dello stesso gruppo del proprietario e l'accesso a tutti gli utenti). La gestione dei permessi da parte di Samba comprende anche una maschera, con cui viene eseguita una operazione di *and* binario con i permessi DOS/*Windows*. I valori predefiniti sono 0744 per i *file* e 0755 per le "*directory*".

Questo metodo permette comunque una mappatura abbastanza limitata dei permessi, in quanto non esiste una corrispondenza biunivoca fra come vengono interpretati dal mondo *Windows* e da quello *UNIX*. Ad esempio in *UNIX* non esiste un meccanismo standard di gestione delle ACL, ma i diversi sistemi operativi lo implementano ciascuno a proprio modo. La versione 2.2 di Samba introduce dei notevoli miglioramenti in questo campo, che tuttavia sono disponibili solamente per certe combinazioni particolari di sistema operativo/*filesystem*.

Esempio di utilizzo di Samba come PDC

```
[global]
security = user
workgroup = GRUPPO
netbios name = NOME
server string = Samba
encrypt passwords = Yes
passwd program = /usr/bin/passwd %u
passwd chat = *password* %n\n *password* %n\n *successfull*
unix password sync = Yes
log level = 2
logon script = scripts\%U.bat
logon path = \\netlogon\%U\profiles
logon home = \\netlogon\%U
domain logons = Yes
preferred master = Yes
domain master = Yes
wins support = Yes
admin users = beppe
hosts allow = 10.10.1. 213.215.88.138
[Profiles]
comment = Windows-User-Profiles
path = /home/%U/profiles
read only = No
guest ok = Yes
browseable = No
[homes]
comment = Home Directories
```

```
read only = No
create mask = 0755
browseable = No
[netlogon]
path = /home/samba/netlogon
create mask = 0755
[cartelle]
comment = Cartelle Condivise
path = /disk2/cartelle
force group = utenti
read only = No
create mask = 0777
directory mask = 0777
guest ok = Yes
hosts allow = 10.10.
[tmp]
path = /tmp
read only = No
create mask = 0755
guest ok = Yes
[printers]
comment = All Printers
path = /var/spool/samba
guest ok = Yes
print ok = Yes
```

Questo esempio, da intendersi come punto di partenza per realizzare una propria configurazione, illustra come Samba possa essere utilizzato per implementare un *server* di rete con funzione di *Primary Domain Controller*. Le diverse opzioni utilizzate sono descritte in dettaglio nella documentazione di Samba.

Creazione *account* per i *client*

È necessario creare un *account* per ognuno dei *client* a cui si desidera offrire accesso al sistema. Per far ciò si deve creare un utente *UNIX* corrispondente, avendo cura di far terminare il nome con il carattere $. Per far questo si può utilizzare il comando

```
# useradd pc001$
```

Non è necessario assegnare una *password*, così come non sono importanti la *home directory* e gli altri campi in `/etc/passwd`, in quanto il controllo sugli accessi viene gestito da Samba mediante il *file* separato `/etc/smbpasswd`. Occorre pertanto aggiungere un record relativo al *client* anche in questo *file*, utilizzando il comando

```
# smbpasswd -a -m pc001
Added interface ip=193.43.98.1 bcast=193.43.98.255 nmask=255.255.255.0
Added user pc001$.
Password changed for user pc001$.
```

Non occorre specificare il $ alla fine del nome, in quanto viene aggiunto direttamente dal programma.

Creazione degli *account* per gli utenti

A questo punto si deve creare gli *account* per gli utenti che possono effettuare il *logon* sui *client*. Anche in questo caso deve essere creato sia l'*account UNIX* che quello nel *file smbpasswd*:

```
# adduser utente1
# smbpasswd -a utente1
```

```
Added interface ip=193.43.98.1 bcast=193.43.98.255 nmask=255.255.255.0
New SMB password:
Retype new SMB password:
Added user utente1.
Password changed for user utente1.
```

Per la creazione degli utenti può essere utile ricorrere allo *script mksmbpasswd.sh*, che permette di popolare il *file* `/etc/smbpasswd` con i record degli utenti definiti in *UNIX*.

Una volta creato l'utente, la modifica di una *password* può essere effettuata sia da *UNIX* che direttamente dal *client windows*. Non è necessario che le *password* usate da *UNIX* e da Samba coincidano.

Una volta configurati in modo opportuno i *client*, sarà possibile accedere alle risorse condivise utilizzando lo *username* appena creato.

Smbclient

Consiste in un *client* a linea di comando, che può essere utilizzato anche all'interno di *script*, e che permette di accedere a volumi condivisi da un *server* SMB. Permette di trasferire *file* da e verso *server Windows*, di spedire messaggi *WinPopUp* e di stampare su *server Windows*. Risulta inoltre un utile strumento diagnostico.

Lo *script smbtar* permette di eseguire il *backup* automatizzato di una rete di PC su un *server UNIX*.

Smbfs

Linux consente di accedere al volume condiviso da un *server Windows* o Samba come se si trattasse di un *filesystem UNIX*. Il *software* necessario è costituito da un modulo del *kernel*, compreso di serie in *Linux*, e dai programmi di gestione (*smbmount*), i quali si trovano all'interno della distribuzione di Samba.

Smbfs sfrutta solamente un sottoinsieme delle possibilità di SMB.

Il *mounting* del *filesystem* avviene mediante il comando *smbmount* specificando direttamente sulla linea di comando le *password* necessarie

```
# smbmount //ntserver/test /mnt4/ -U test
Added interface ip=193.43.98.1 bcast=193.43.98.255 nmask=255.255.255.0
Password:
```

oppure utilizzando il classico *mount* con l'opzione -t smbfs

```
# mount -t smbfs -o
username=test,password=test123,netbiosname=linuxserv //ntserver/JAZZ /backupdir/
```

È disponibile, come pacchetto a parte, l'interfaccia grafica Gnomba, che permette di sfogliare una rete e montare *filesystem* in modo simile a quanto avviene nei *client Windows*.

Maggiori informazioni sono disponibili all'interno della documentazione del *kernel* di *Linux* nel *file* `/usr/src/linux/Documentation/filesystems/smbfs.txt`.

Modulo PAM per autenticazione mediante server SMB

Si tratta di un modulo di autenticazione che permette l'integrazione di *workstation UNIX* in reti *Windows* NT. È utilizzabile anche come metodo di autenticazione per offrire servizi Intranet a reti *Windows*.

Esiste anche un modulo che permette autenticazione attraverso un *server* SMB per il *Web server Apache*.

Servizi di rete in Linux

Franco Callegati
Paolo Presepi
Riccardo Gori
7.1.3. (Installare e configurare un server di rete locale), 7.1.6. (Descrivere aspetti di gestione e procedure per gestire più server su una rete)

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).

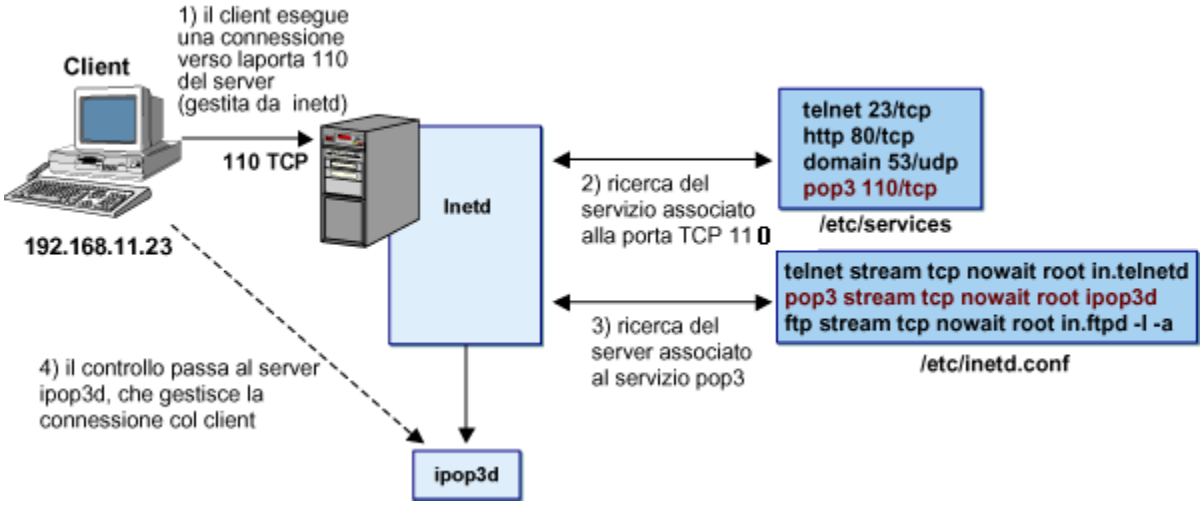
I demoni che sovrintendono ai servizi di rete vengono normalmente lanciati mediante uno *script* di avvio in `/etc/rc.d`. Ad esempio il *Web server* `httpd` viene avviato su *Red Hat Linux* mediante `/etc/rc.d/init.d/httpd` (o meglio, mediante un link simbolico posto nella *directory* `rc.d` corrispondente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

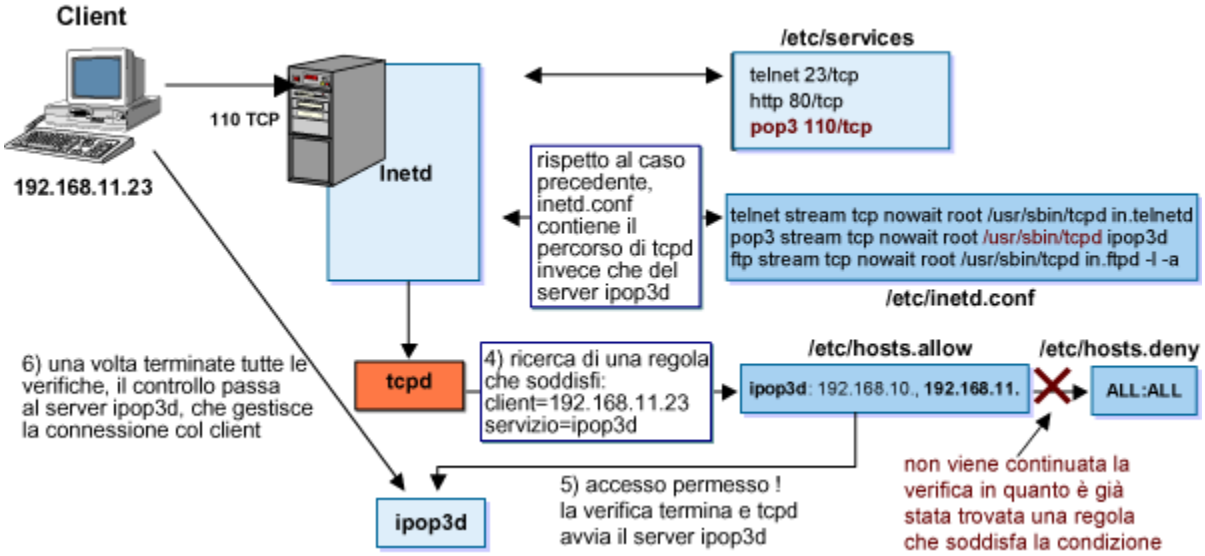
Aggiunta di un servizio ad inetd.conf

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*
only_from	*access list* sulle reti o macchine da cui accettare le connessioni verso questo 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
```

Bibliografia

Linux

Documentazione su Linux
The Linux Documentation Project; **http://www.tldp.org**
PLUTO Free Software Users Group; **http://www.pluto.linux.it**

Documentazione su Linux
The Linux Documentation Project; **http://www.tldp.org**
PLUTO Free Software Users Group; **http://www.pluto.linux.it**

Libri elettronici su Linux
O'Reilly Book List; **http://www.oreilly.com/catalog**

Servizi di rete integrati Windows/Unix e Linux
Samba Web Pages; **http://www.samba.org**

Windows

Documentazione stampata su Windows
Microsoft Corporation; *MCSE Training Kit - Microsoft Windows 2000 Professional;* 2001Microsoft Press
Microsoft Corporation; *MCSE Training Kit - Microsoft Windows 2000 Server;* 2002Microsoft Press
Microsoft Corporation; *Microsoft Windows 2000 Server Resource Kit;* 2000Microsoft Press
Microsoft Corporation; *Microsoft Windows 2000 Server Administrator's Companion;* 2002Microsoft Press

Documentazione on line sui sistemi Windows
Microsoft Help and Support; **http://support.microsoft.com/**

Glossario

Apache: Rappresenta un HTTP *server* multipiattaforma.

BGP (Border Gateway Protocol): Protocollo di *routing*, standardizzato da IETF, usato da un *exterior router* in un *autonomous system* per annunciare gli indirizzi delle reti appartenenti all'*autonomous system* stesso.

DNS (Domain Name System): Sistema/servizio per l'associazione e la traduzione di indirizzi numerici IP in nomi logici alfanumerici mnemonici. È basato su una struttura gerarchica ad albero, ad ogni ramo del quale corrisponde un dominio.

Dominio di routing: Termine generico che indica una partizione gerarchica della rete contenente un insieme di nodi e di *router*; i *router* condividono le stesse informazioni di *routing*, calcolano le tabelle utilizzando lo stesso algoritmo, e sono gestiti da un amministratore comune.

Dominio Windows: Insieme di *computer* forniti di sistema operativo *Microsoft Windows* (95, 98, NT, 2000, XP) collegati in rete. I servizi di rete forniti dalle macchine *server* sono tipicamente sotto il controllo di un'unica entità amministrativa (supervisore del dominio).

How-TO: Documento che spiega il funzionamento ed elenca le linee guida di comandi, pacchetti *software*, apparati *hardware* legati al sistema operativo *Linux/Unix*. Generalmente in formato digitale, può essere scritto in ogni formato: testo puro, HTML, PDF, eccetera. Una raccolta degli *HOW-TO* più diffusi la si può trovare all'indirizzo

http://www.tldp.org (*The Linux Documentation Project*).

IMAP (Internet Message Access Protocol): Protocollo per l'accesso al servizio di posta elettronica e per la gestione della casella postale. Consente la manipolazione e la gestione dei messaggi fermo restando che questi restino sul *server*, senza che debbano essere trasferiti sul *client* di posta.

ISP (Internet Service Provider): Fornitore di servizio di accesso ad Internet. Generalmente, per gli utenti residenziali l'accesso è fornito mediante collegamento telefonico al POP del *provider*, mentre per categorie di utenti di tipo affari, il collegamento può essere su linea dedicata e collegamento diretto numerico fra la sede dell'utente ed il *router* del *provider*.

LDAP (Lightweight Directory Access Protocol): Protocollo Internet per i servizi di *directory*.

LDIF (LDAP File Interchange Format): Rappresenta un formato standard per la descrizione in formato testo, di una *directory* (elenco di informazioni) e delle *entry* in essa contenute.

LILO (LInux LOader): Si tratta di un componente *software* che consente di effettuare il caricamento del sistema operativo *Linux* su una macchina. Non ha dipendenze rispetto al *file system* e consente di effettuare il *boot* del *kernel* di *Linux* (e anche di altri sistemi operativi) da diverse periferiche di *computer* (*floppy*, CD, *hard disk*).

Multicasting: Trasmissione di informazioni da una sorgente ad un gruppo precostituito ed indirizzabile di ricevitori.

Multipiattaforma: Si dice di programma o codice che può essere eseguito su diversi sistemi operativi (piattaforme).

NAT (Network Address Translation): Funzionalità di *mapping* fra indirizzi interni ad una rete (privati, tipicamente di classi 10.x.y.z, 192.168,x,y, eccetera) e indirizzi esterni (pubblici e quindi univoci in ambito Internet). Un NAT *server* (tipicamente implementato mediante un *router* di frontiera fra una rete privata e la Internet Pubblica) gestisce un *pool* di indirizzi pubblici assegnandoli ai *client* della rete privata, man mano che richiedono connettività verso l'esterno.

NETBEUI (NETBios Extended User Interface): Protocollo per la realizzazione di reti di PC, appartenente all'architettura SNA ed usato come protocollo standard nelle reti *Microsoft*.

NetBIOS (Network Basic Input Output System): API standard per le reti di *Personal Computer*.

NFS (Network File System): Protocollo sviluppato da *SUN Microsystems* che si appoggia sull'architettura di rete TCP/IP e consente ad un insieme di elaboratori di condividere i *file system*. È spesso utilizzato dai *client* in ambienti di reti locali di *computer* per utilizzare porzioni del disco di un *server* come disco di rete. Prevede una parte *client* ed una parte *server*.

NIS (Network Information Service): Servizio Internet mediante il quale è possibile l'amministrazione centralizzata di un numero qualsiasi di *file* di qualsiasi tipo in rete.

NMS (Network Management System): Centro di gestione di rete.

NNTP (Network News Transfer Protocol): Protocollo applicativo utilizzato in ambito Internet per il trasferimento di *news*. Utilizza il servizio di trasporto TCP.

NTFS (Windows NT File System): Rappresenta l'architettura del *file system* nel sistema operativo NT di *Microsoft*. Per approfondire l'argomento si suggerisce **http://www.microsoft.com/msj/1198/ntfs/ntfstop.htm**.

OSPF (Open Shortest Path First): Protocollo di tipo *LINK STATE*, utilizzato fra *router* nell'architettura TCP/IP per il calcolo delle tabelle di instradamento.

PAP (Password Authentication Protocol): È un protocollo utilizzato per l'autenticazione degli utenti che si connettono attraverso modem e linea telefonica commutata, nei confronti di un *server* di accesso ad una rete (esempio: accesso ad Internet).

PPP (Point to Point Protocol): Protocollo di livello 2 utilizzato per la trasmissione dati seriali in ambito Internet fra un *client* (tipicamente PC) ed un apparato di *networking* (*router*). È tipicamente impiegato per l'accesso ad Internet fra l'utente ed il *service provider*; viene anche impiegato per collegare fra loro due *router* attraverso rete pubblica geografica, utilizzando per esempio una connessione commutata telefonica, una connessione commutata numerica (ISDN) oppure una connessione numerica permanente.

Proxy: Componente *software* e/o *hardware* utlizzato per varie funzioni: per esempio per controllare da un punto unico l'accesso ad Internet e per analizzare i pacchetti IP che Internet accedono all'interno di una rete privata. Un *proxy* può avere anche le funzioni di traduzione di indirizzi IP privati in indirizzi IP pubblici e viceversa.

QoS (Quality of Service): È definita, per tradizione nelle reti OSI e ATM, su base *end to end* per una connessione dati, in termini di attributi generici, quali: tasso di perdita di pacchetti, ritardo di trasferimento, varianza del ritardo.

RIP (Routing Information Protocol): È un protocollo di *routing* tra i più semplici, utilizzato in ambiente Internet, soprattutto in reti di piccole dimensioni. È il più implementato sulle piattaforme *hardware* più disparate (anche su *UNIX* e *Windows* NT). Utilizza il numero di *router* attraversati per confrontare i diversi percorsi alternativi di instradamento.

Route: Percorso di instradamento; nei *router* IP esiste una *route* per ogni *subnet* raggiungibile.

Routing: Funzione di instradamento dei pacchetti a livello *Network*.

SMTP (Simple Mail Transfer Protocol): Protocollo definito nella RFC 821, ed utilizzato per il trasferimento della posta elettronica tra *computer*. È un protocollo utilizzato per la trasmissione della posta. Per la ricezione dei messaggi invece vengono impiegati altri protocolli (POP, IMAP).

SNMP (Simple Network Management Protocol): Protocollo per lo scambio di informazioni di gestione in ambiente TCP/IP. Ad oggi molte implementazioni di SNMP consentono la gestione di ambiti anche diversi da reti TCP/IP.

SSL (Secure Socket Layer): È uno strato di *software* che si posiziona fra TCP ed una applicazione e consente di gestire un canale sicuro di comunicazione fra *client* e *server*. La cifratura dei dati avviene mediante algoritmi crittografici asimmetrici a chiave pubblica.

Subnet: Nell'architettura di rete TCP/IP una rete può essere suddivisa in un insieme di sottoreti mediante la definizione di una *netmask*.

Tabella di routing: Tabella contenente le informazioni utili per gli algoritmi di instradamento quali, per ogni destinazione, la linea da utilizzare, il costo e il numero di *hop*.

Telnet: Applicazione Internet basata sul protocollo TCP, che consente di remotizzare attraverso una rete IP, l'accesso a un *host* remoto. Il *client* si comporta come terminale remoto del *server* telnet ed accede alle risorse dell'*host* mediante l'autenticazione con una *username* ed una *password*.

Unicasting: Trasmissione di informazioni da una sorgente verso una singola destinazione.

VPN (Virtual Private Network): Indica, in diversi ambiti, la possibilità di utilizzare una rete fisicamente condivisa con altri insiemi di utenti, come se fosse privata e inaccessibile (ovvero accessibile con restrizioni) rispetto ad utilizzatori che non ne fanno parte.

Autori

Hanno realizzato il materiale di questo modulo:

Prof. Franco Callegati
Franco Callegati è; professore associato di Reti di Telecomunicazioni presso il Dipartimento di Elettronica, Informatica e Sistemistica (D.E.I.S.) dell'Università di Bologna. Presso la Facoltà di Ingegneria di Bologna prima ed ora presso la Facoltà di Ingegneria di Cesena ha tenuto e tiene corsi di base di Reti di Telecomunicazioni e corsi avanzati su teoria del traffico e progettazione di reti. Si interessa di problematiche di dimensionamento e progettazioni di reti di telecomunicazione a larga banda e la sua attività di ricerca più recente ha come oggetto le reti ottiche ad altissima velocità, argomento sul quale ha pubblicato numerosi lavori, partecipando a progetti di ricerca nazionali ed internazionali con ruoli di coordinamento.

Dott. Ing. Paolo Presepi
Paolo Presepi è laureato in Ingegneria Elettronica, si interessa di reti di telecomunicazioni, sistemi operativi e sistemi di sviluppo a microcontrollore. Ha effettuato docenze presso enti di formazione professionali e ha collaborato alla realilzzazione di vari progetti dell'Università degli Studi di Bologna nella sede di Cesena, tra cui la progettazione ed il *set-up* di una rete *wireless* sperimentale. Ha ottenuto una borsa di ricerca nell'ambito del progetto Spinner della Regione Emilia Romagna per lo studio di Soluzioni di rete *embedded* nell'*information technology* che ha portato alla creazione della società Net-IT, operante nel campo delle tecnologie di *networking*, di cui è uno dei soci fondatori.

Dott. Ing. Riccardo Gori
Riccardo Gori è laureato in Ingegneria delle Telecomunicazioni, si interessa di reti di telecomunicazioni e sistemi operativi. Ha collaborato con l'Università degli Studi di Bologna, fornendo consulenze, progettazioni di reti dati e attività di supporto alla didattica per gli insegnamenti di Reti di Telecomunicazioni. Ha effettuato ed effettua docenze presso enti di formazione professionali e aziende che operano nel campo dell'informazione. Ha ottenuto una borsa di ricerca nell'ambito del progetto Spinner della Regione Emilia Romagna per lo studio di Soluzioni di rete *embedded* nell'*information technology* che ha portato alla creazione della società Net-IT, operante nel campo delle tecnologie di *networking*, di cui è uno dei soci fondatori.

Modulo realizzato sulla base di materiali prodotti nell'ambito di un piano di formazione di 12.000 tecnici delle pubbliche amministrazioni e messi a disposizione del MIUR dall'Autorità per l'Informatica nella Pubblica Amministrazione (AIPA).