Lezione 2 - Introduzione a Unix e GNU/Linux

21 novembre 2006

Alessio Checcucci

Riccardo Aldinucci

Q.It Universita' degli Studi di Siena

1 La migrazione da Windows

Gli utenti provenienti dai sistemi operativi Microsoft e Apple si trovano spesso spaesati dall'apparente complessita' dei sistemi Unix. Questo poteva essere vero qualche anno addietro. Al momento gran parte delle distribuzioni GNU/Linux permettono un'installazione grafica ed interattiva molto comoda. Una serie di GUI (graphical user interface) permettono poi l'amministrazione del sistema mascherando buona parte dei dettagli sottostanti. Chiaramente se si vuole avere un pieno controllo del sistema e procedere al fine tuning dello stesso sara' necessario addentrarsi nei dettagli. Classicamente l'amministrazione dei sistemi Unix e' stata sempre fatta per mezzo di file di testo (in genere residenti nella directory /etc e nelle sue sottodirectory). Questo puo' risultare sconcertante per un amministratore Windows o MacOS, abituato ad avere a disposizione potenti interfacce grafiche con le quali amministrare i dettagli del sistema. Le interfacce grafiche che le distro GNU/Linux forniscono sono si potenti, ma non coprono tutte le possibili sfaccettature della configurazione.

Una volta entrati nel meccanismo, comunque, la configurazione per mezzo di file ascii non comporta eccessivi grattacapi, a patto di sapere dove si mettono le mani!! In ogni caso permette un completo controllo del sistema e non lascia niente mascherato da un'interfaccia grafica, col risultato di perdere un sacco di tempo per capire come mai le cose non funzionano (regedit vi dice niente?).

1.1 L'interazione con il sistema

L'utente al termine del processo di boot del sistema operativo si trovera' di fronte ad una schermata di login, come per qualsiasi altro sistema multiutente. Il login potra' essere fatto su console, oppure su un terminale grafico. Questo a seconda che l'amministratore di sistema abbia configurato il runlevel testuale (in genere il 3) oppure quello grafico (in genere il 5). Nell'ultimo caso quello che compare all'utente e' un client del server grafico X, il cui nome e' xdm (X Display Manager) o uno dei suoi numerosi fratelli, che si occupera' del processo di login.

Come in tutti gli altri sistemi mutiutente occorrera' disporre di un account abilitato sulla macchina per poter accedere. Una volta fornite le corrette credenziali (nome utente e password), un processo si occupera' di validare l'accesso e ci fara' accedere o ad un terminale in cui sia stata lanciata la nostra shell di default, oppure alla sessione grafica.

L'uso della sessione grafica e' di gran lunga meno spiazzante per gli utenti che provengono da ambienti quasi totalmente GUI oriented. La shell nella console puo' sembrare un ambiente povero con cui interagire, ma al contrario presenta notevole potenza, tant'e' che anche nell'ambiente grafico, molto spesso viene lanciato un terminale con una shell per poter impartire i comandi.

Una differenza fondamentale fra altri sistemi operativi e Unix e' che quest'ultimo e' totalmente *case sensitive*, vale a dire che in caratteri minuscoli e maiuscoli NON sono la stessa cosa.

Questo naturalmente vale sia per l'invocazione dei comandi che per i nomi dei

file a cui si fa riferimento. I file system case insensitive (overo quelli provenienti dal mondo Microsoft) vengono trattati di default come case insensitive, ovvero nello stesso modo che nei sistemi operativi originari.

La maggior parte delle utility GNU prevedono un opportuno flag per rendere il comando stesso case insensitive, per esempio l'opzione -i di grep oppure il -iname di find.

1.2 Gli utenti, root e la multiutenza

1.2.1 I sistemi multiutente

Unix nasce originariamente come sistema multiutente, nel senso che piu' utenti possono interagire col sistema in contemporanea e il sistema operativo implementa un livello di astrazione tale da mascherare ad ognuno la presenza degli altri. Nella sostanza ogni processo utente e' convinto di essere solo nel sistema e di interagire in modo prioritario con il sistema operativo, che al contrario si prende carico di distribuire le risorse.

Il sistema e' in questo senso *multiprocesso* nel senso che permette l'esecuzione "contemporanea" di piu' processi, tramite un sistema a divisione di tempo gestito da un elemento del sistema operativo detto *scheduler*. Esistono anche sistemi operativi multiprocesso in cui non e' il sistema operativo a decidere l'assegnazione delle risorse e lo scheduling dei processi, ma sono le applicazioni stesse a rilasciare le risorse. Questi sistemi operativi si dicono a multitasking cooperativo. Esistono infine sistemi multiprocesso che non sono multiutente, un esempio per tutti e' Microsoft Windows 98.

1.2.2 Gli utenti

Gli utenti sono quelle entita' che possono vantare credenziali riconosciute dal sistema e possono interagire con il sistema stesso in vario modo. Gli utenti che il sistema puo' riconoscere sono (in genere, salvo meccanismi esterni di autenticazione) registrati nel file /etc/passwd. Storicamente questo file conteneva anche le password degli utenti. Naturalmente questo era in contrasto con i piu' elementari criteri di sicurezza, per cui e' stato introdotto il concetto delle shadow password, ovvero hash crittografiche (attualmente MD5) della password affiancate da altri campi:

- Password: detta anche *key* nell'uso della funzione crypt().
- Salt: Una coppia di caratteri scelti dal set [a-zA-Z0-9./]. Questo permette di dare casualita' all'algoritmo, avendo a disposizione 4096 valori.
- Pad: Campo di riempimento.

Le shadow password sono salvate in un file separato detto shadow, a cui solo il superutente (root) puo' accedere. Per poter cambiare la propria password l'utente dovra' interagire con questo file, e lo fara' grazie al fatto che il comando passwd e' SUID a root.

I comuni utenti avranno la possibilita' di collegarsi al sistema su un terminale virtuale e poter navigare fra le directory e lanciare comandi, ovvero avranno associato al loro account un programma detto *login shell*.

Esistono poi tutta una serie di utenti, detti di sistema, che non avranno associata alcuna login shell e quindi non potranno aprire sessioni interattive col sistema. Essi sono utilizzati per la maggior parte per far girare dei servizi (come per esempio il server http/https) che non necessitano di privilegi elevati oppure allo scopo di limitarne i privilegi e quindi i rischi per la compromissione del sistema.

Ogni utente oltre ad un nome unico avra' associato uno User IDentifier (o UID, in genere a 16 bit) che lo identifica al sistema. Tutti i meccanismi di accesso al sistema sono infatti basati su questo numero. Nulla vieta peraltro che due utenti condividano lo stesso UID, anche se la cosa non e' molto "consigliata".

1.2.3 Root

Nel normale modello di Unix (ovvero se non sono presenti meccanismi Mandatory Access Control o extended capabilities) tutti gli utenti sono equivalenti nei confronti del sistema operativo e le limitazioni di accesso sono basati sui permessi dei file (ricordate in Unix "tutto e' un file"). Esiste pero' un utente privilegiato detto *superuser* o *root* (root of all evil) a cui non si applica alcuna restrizione di accesso. E' per molti versi l'equivalente di quello che in Windows e' l'amministratore (administrator). Esso viene identificato per mezzo dello UID 0.

Non tutti gli Unix presentano un modello cosi' debole di protezione, nel senso che l'utente root non ha limitazioni ai propri privilegi ed e' il grantor dei privilegi per gli altri utenti. I sistemi di derivazione BSD per esempio prevedono un utente root e un secondo utente detto *toor* che puo' essere usato per compiti amministrativi senza avere il potere illimitato di root.

La morale della favola e': utilizzare l'utente root SOLO quando strettamente necessario e dare agli utenti i permessi necessari e sufficienti a poter lavorare senza compromettere la sicurezza del sistema.

1.2.4 Creazione degli utenti

Sebbene la creazione di un utente possa avvenire anche "a mano", le distribuzioni provvedono una serie di utility per la creazione degli utenti. Generalmente l'utente root viene creato durante l'installazione e qualche distro permette anche la creazione di ulteriori utenti non privilegiati. Altrimenti il superutente puo' crearli dopo aver fatto il login nel sistema.

Tutte le distibuzioni forniscono un comando (facente parte del pacchetto shadow-utils) dal nome *adduser* che permette la creazione ordinata di un utente con tutte le sue caratteristiche. Ogni distribuzione ha la propria GUI personalizzata che usa come backend questo programma. Esse possono essere totalmente grafiche, a curses, oppure testuali.

Quando viene creato un utente, verra' chiesto anche il default group. Oltre ad esso l'utente potra' appartenere ad ulteriori gruppi e cambiare la propria appartenenza a runtime con il comando *newgrp*.

Al momento della creazione dell'utente viene richiesta anche la shell che questo utilizzera'; per gli utenti che non devono poter fare login interattivi vengono utilizzate particolari shell come /bin/nologin, /bin/false oppure /dev/null.

All'atto di creazione degli utenti vengono create anche le loro *home directory*, ovvero le directory a cui l'utente accede appena fatto il login e che rappresentano le singole aree di lavoro. Nella home directory vengono copiati una serie di file dalla directory /etc/skel. In genere si tratta di file di inizializzazione dell'ambiente che poi l'utente potra' modificare a proprio piacimento.

Gli account utente possono avere o meno una scadenza della password e possono essere bloccati. Fare questo e' piuttosto banale. Basta mettere un carattere non valido (per esempio *!*) nel secondo campo (la password criptata) del file /etc/shadow.

1.3 I gruppi

I gruppi consentono agli utenti di condividere le risorse con altri utenti. Il modello di controllo sui file che Unix tradizionalemente implementa non e' molto granulare, ma puo' essere rafforzato con meccanismi come ACL.

In generale tutti gli appartenenti ad un gruppo avranno accesso, con qualche livello di privilegio, ai file di quel gruppo.

Un utente puo' naturalmente appartenere a piu' gruppi.

I gruppi sono elencati nel file /etc/group. In genere il default group di un utente viene indicato nel file /etc/passwd, mentre per i gruppi aggiuntivi si accoda lo username alla riga realtiva al gruppo in questione nell'ultimo campo del file /etc/group.

Cosi' come per gli utenti anche i gruppi sono identificati per mezzo di un numero (in genere a 16 bit) detto Group IDentifier (o GID) e tutti i meccanismi di enforcement dei permessi si basano su di esso.

Anche per i gruppi potrebbe essere definita una password. Ma non e' una pratica comune. Essa rappresenterebbe il secondo campo (in forma criptata) nei record del file /etc/gshadow, che classicamente costituito da un carattere non valido (!).

1.3.1 Creare un gruppo

In genere i gruppi vanno creati prima di fargli appartenere qualche utente. Questo puo' essere fatto con una delle moltiplici GUI messe a disposizione dalle distribuzioni, oppure in modo molto semplice con l'obiquo comando *groupadd*, che richiede come paramentro in pratica solo il nome del gruppo.

1.4 I permessi di accesso

I file system Unix e il sistema stesso si basano sui permessi con cui gli utenti possono accedere a file e directory. Se trascuriamo per il momento gli extended attributes (come le ACL), vediamo il modello classico dei permessi.

A ogni oggetto del filesystem sono associati una serie di flag. Che costituiscono il primo campo restituito dal comando ls -l (a meno che non siano stati definiti particolari alias). Questi flag sono 10. E vengono salvati all'interno dell'inode associato all'oggetto.

- Il primo carattere individua il tipo di oggetto:
  - - indica un comune file
  - d indica una directory
  - c indica un character oriented device file
  - b indica un block oriented device file
  - s indica una socket
  - l indica un link simbolico, ricordiamo che un hard link non e' altro che un nome alternativo per un file e quindi viene trattato in tutto e per tutto come un file (e' possibile comunque accorgersene andando a vedere il link count associato al file).
  - p indica una named pipe

A riguardo delle named pipes vale la pena di spendere qualche parola. Esse sono definite anche FIFO, e servono ad estendere il concetto di comandi semplici collegati tramite pipe che sta alla base della gestione di Unix. Una named pipe e' un oggetto del filesystem che restituisce in uscita cio' che gli e' stato passato in ingresso e nello stesso ordine. L'utilita' di una named pipe entra in gioco quando l'output di un comando non sia possibile redirigerlo sullo standard output, in maniera da mettere in moto il classico meccanismo di pipe. In questo caso e' possibile specificare come output file la pipe stessa e poi prendere l'uscita della pipe per concaternarvi altri comandi. Un tipico caso e' l'esecuzione di un full export di un database Oracle. Il comando di export prevede come paramentro il file di dump. Siccome il full export di un database puo' essere di dimensioni davvero poderose e quindi potrebbe non entrare nel filesystem, occorre comprimerlo mentre viene effettuato (i dump di database hanno compression rate in genere superiori all'80%). Per fare questo si definisce una named pipe:

mknod /tmp/dump_file.dmp p

Dopo di che si puo' lanciare il comando di compressione con in ingresso la pipe e in uscita il file compresso:

gzip -9 </tmp/dump_file.dmp >export.dmp.gz

e infine il comando di export:

exp system/password full=y file=/tmp/dump_file.dmp

- I seguenti 9 caratteri sono relativi ai permessi di accesso all'oggetto. Sono divisi in tre gruppi: permessi per l'utente a cui appartiene l'oggetto, permessi per il gruppo a cui appartiene l'oggetto e permessi per tutti gli altri (others). Quando viene applicato il criterio di accesso all'oggetto, la valutazione avviene esattamente in quest'ordine.

1.4.1 I permessi dell'utente

I seguenti tre flag sono relativi ai permessi dell'utente proprietario del file. Essi sono nella forma di una tripletta:

rwx

- Il campo *r* indica il permesso di lettura per l'oggetto
- Il campo *w* indica il permesso di scrittura per l'oggetto
- Il campo *x* indica il permesso di esecuzione per l'oggetto

Ogni configurazione di questi 3 flag e' valida. Qualora uno o piu' di essi non sia attivo, viene sostituito dal carattere -.

Vale la pena di far notare che il solo flag x possa essere applicato a file eseguibili ai quali non si voglia concedere la lettura. Naturalmente l'assenza del flag r rende gli script (che per loro narura devono essere letti ed interpretati) del tutto inservibili.

Notiamo anche il fatto che questi flag sono in realta' degli switch. Ovvero ad ognuno di essi puo' essere associato il valore binario 1 o 0. Il fatto di essere raggruppati a gruppi di 3 bit, ha portato a associare alla configurazione dei flag un valore ottale. In particolare il flag r avra' peso 4 se attivo, il flag w il peso 2 se attivo e il flag x il peso 1 se attivo. Quindi un oggetto per cui tutti i flag siano attivi, avra' come permessi (espressi in ottale) 7.

Questa notazione e' molto comoda quando si definiscono i permessi dei file (per es. con chmod).

1.4.2 I pemessi del gruppo

Similarmente all'utente, il gruppo a cui appartiene l'oggetto ha una tripletta di permessi:

rwx

Il loro significato e' del tutto equivalente a quelli dell'utente a cui appartiene l'oggetto. Essi naturalmente si applicano ai membri del gruppo a cui appartiene l'oggetto escluso il prorpietario.

1.4.3 I permessi degli altri

Se un utente non e' proprietario dell'oggetto, ne' fa parte del gruppo a cui apartiene l'oggetto, allora fa parte di tutti gli altri, definiti others o world. Anche in questo caso avremo a che fare con una tripletta di permessi:

rwx

Il significato e' del tutto similare ai casi precedenti, va pero' fatto notare che l'impostazione dei permessi per gli others deve essere fatta con estrema cautela, in quanto tutti gli utenti del sistema, anche quelli che non appartengono al gruppo avranno i permessi specificati. In genere non e' conveniente assegnare il permesso di scrittura ai file a questa categoria di utenti, se non in specifiche situazioni.

Da questa breve disamina si evince come il sistema di permessi di Unix sia potente, ma poco flessibile. Per oltrepassare questo ostacolo, i filesystem hanno introdotto degli extended attributes, dei quali fanno parte le ACL.

1.4.4 I permessi delle directory

I flag appena visti si applicano a tutti gli oggetti dei filesystem, ma hanno un significato particolare per le directory. In questo caso e' chiaro che il flag x non avrebbe senso se gli attribuiamo il significato di permesso di esecuzione. Il flag x in questo caso identifica il permesso di accedere alla directory. In sua presenza sara' possibile fare un cd nella directory ed riferirisi a file al suo interno anche in assenza del flag r. naturalmente in questo caso non sara' possibile leggere il contenuto della directory (che ricordiamo, in Unix e' in tutto e per tutto un file che contiene i nomi degli oggetti al suo interno).

1.4.5 I bit set user id, set group id e sticky

Questi flag hanno un uso particolare, per cui vengono esposti a parte. Per prima cosa, nella notazione ottale essi vengono espressi prima dei restanti flag, finendo per essere una quarta cifra in tutto e per tutto. Essi sono:

- Set User ID (SUID). Nel caso questo bit sia applicato ad un file eseguibile, quando esso viene eseguito (da qualsiasi utente) esso avra' il permessi del proprietario del file. Nel caso questo fosse root, il processo girera' con i privilegi del superutente. Quando questo bit e' attivo una s va a sostituire il flag x.
- Set Group ID. Nel caso questo bit sia applicato ad un file eseguibile, quando esso viene eseguito (da qualsiasi utente) esso avra' il permessi del gruppo proprietario del file. Nel caso di una directory invece questo flag indica che un file che venga creato in essa dovra' avere come gruppo quello della directory e non quello del processo che l'ha creato (BSD style). Quando questo bit e' attivo una s va a sostituire il flag x.
- Sticky bit. Lo sticky bit rappresenta per i file un retaggio del passato, quando la quantita' di memoria dei sistemi era scarsa e le periferiche molto lente. In questo caso corrispondeva ad una richiesta di mantere un programma terminato nella memoria virtuale, in particolare nello swap. La vera utilita' di questo bit si ha per le directory. In una directory in cui sia attivo lo sticky bit, i file presenti possono essere cancellati solo dall'utente che li ha creati o dalla root. La tipica applicazione e' ad una directory condivisa da tutti gli utenti come /tmp.

1.4.6 Umask

La umask permette di assegnare default permissions per gli oggetti creati da un utente. Essa e' tradizionalmente espressa come il valore ottale complementare a quello dei totali permessi. Ovvero il valore

777-umask

definisce i permessi di default per gli oggetti creati dall'utente.

la definizione della umask viene fatta attraverso l'internal di shell

umask

Le flag di interesse sono -S, che fa assumere un aspetto simbolico (simile a chmod) e -p che restituisce in output il comando da lanciare per impostare la umask al valore attuale.

1.5 L'autenticazione nel sistema

I sistemi Unix tradizionali hanno sempre autenticato gli utenti per mezzo di programmi (come login, su e similia) che gestivano in proprio il meccanismo di autenticazione stessa. Gli Unix moderni e Linux in particolare, si avvalgono di una struttura comune a cui i vari programmi si appoggiano quando hanno esigenze di autenticazione e autorizzazione. Quest'infrastruttura (costituita da una serie di librerie e la definizione di un API) si chiama Pluggable Authentication Modules o PAM.

1.5.1 PAM

Linux-PAM e' un sistema di librerie che gestiscono i task di autenticazione delle applicazioni nel sistema. La libreria fornisce una API a cui i programmi che concedono privilegi (come su oppure login) si appoggiano per i comuni task di autenticazione.

La caratteristica principale dell'approccio PAM e' che la natura dell'autenticazione e' dinamicamente configurabile, ovvero l'amministratore di sistema e' libero di scegliere come le singole applicazioni autenticano gli utenti. La configurazione dinamica viene controllata per mezzo del file */etc/pam.conf* oppure molto piu' frequentemente da una serie di singoli file di configurazione contenuti nella directory */etc/pam.d*.

Per l'amministratore di sistema l'aspetto importante e' che questi file di configurazione definiscono la connessione fra le applicazioni (services) e i pluggable authentication modules (PAMs) che effettuano la vera autenticazione.

Linux-PAM divide i task di autenticazione in 4 quattro gruppi amministrativi indipendenti: *account* management, *auth*entication management, *password* management, *session* management.

Questi gruppi si occupano di aspetti diversi di una tipica richiesta utente per un servizio con restrizioni:

- **Account**: fornisce il tipo di servizio di verifica degli account: la password e' scaduta? L'utente puo' accedere al servizio richiesto?

- **Auth**entication: stabilisce che l'utente sia veramente chi dichiara di essere, in genere attraverso una qualche challenge che l'utente deve soddisfare. Non tutte le autenticazioni sono di questo tipo, alcune si basano su particolari device hardware. Per questi tipi di autenticazione esistono specifici moduli che possono sostituire quelli standard.

- **Password**: il task responsabilita' di questo gruppo e' quello di aggiornare i meccanismi di autenticazione. In genere sono servizi strettamente legati a quelli del gruppo auth. Il tipico esempio e' il rinnovo dell'accesso basato su password degli Unix standard.

- **Session**: questo gruppo di task si occupa di cose che dovrebbero essere fatte prima che un servizio sia concesso e dopo che esso e' terminato. Per esempio il mantenimento della parte audit e il mounting della home directory dell'utente. Il gruppo amministrativo session e' importante perche' fornisce un anello di apertura e chiusura per i moduli per influenzare i servizi a disposizione degli utenti.

Quando una applicazione che concede autorizzazioni e' stata compilata con il supporto PAM e viene avviata, essa attiva il proprio collegamento alla PAM API. Questa attivazione comporta una serie di task, il piu' importante del quali e' la lettura dei file di configurazione.

Questi file elencano i moduli PAM che si occuperanno del task di autenticazione richiesti dal servizio e il comportamento della PAM API nel caso uno di questi moduli fallisse.

Il formato dei file nella directory /etc/pam.d e' del tipo:

type control module-path module-arguments

Il servizio a cui si riferisce il file di configurazione e' dato dal nome del file stesso, che deve essere in caratteri minuscoli. Il nome di servizio other e' riservato per dare una lista di regole di default.

Una caratteristica importante di Linux-PAM e' che un certo numero di regole possono essere accodate (stacked) per combinare i servizi di una serie di moduli PAM per un certo task di autenticazione.

- Type: e' il management group a cui corrisponde la regola. Viene utilizzato per specificare a quale management group la seguente regola debba essere associata. (account, auth, password, session).

- Control: indica il comportamento della PAM-API se il modulo dovesse fallire nel suo task di autenticazione. La sintassi piu' usata per questo paramentro e' quello di una singola parola:

  - *requisite*: il fallimento del modulo PAM risulta nella fine immediata del processo di autenticazione
  - *required*: il fallimento del modulo PAM condurra' la PAM API a ritornare un messaggio di fallimento solo dopo che gli altri moduli in stack (per questo type) sono stati invocati

- *sufficient*: il successo di questo modulo e' sufficiente per soddisfare le esigenze di autenticazione dello stack di moduli. Se, pero', un precedente modulo required e' fallito, il successo del modulo sufficient viene ignorato.
- *optional*: Il successo o fallimento di questo modulo e' importante solo se esso e' l'unico modulo nello stack per quel type

1.5.2 L'autenticazione centralizzata: NIS

Il NIS, o Network information service, è un sistema di gestione di dati amministrativi concentrati in una sola fonte, rendendoli disponibili a tutta una rete in modo uniforme.

Questo tipo di servizio è stato ideato e sviluppato originariamente dalla Sun Microsystems denominandolo Yellow pages. Molti programmi di servizio che riguardano questa gestione hanno il prefisso yp, spesso si parla di «servizi YP» invece che di «servizi NIS». Quando si introduce il NIS, si inserisce un livello di intermediazione tra l'utente e il sistema di amministrazione preesistente.

Lo scopo del NIS è quello di concentrare in un solo elaboratore la gestione di una serie di file amministrativi:

/etc/passwd
Informazioni sugli utenti.
/etc/group
Gruppi di utenti.
/etc/shadow
Parole d'ordine oscurate
/etc/aliases
Alias di posta elettronica.
/etc/hosts
Traduzione degli indirizzi IP dei nodi della rete locale.
/etc/networks
Traduzione degli indirizzi IP delle sottoreti (locali).
/etc/protocols
Nomi e numeri dei protocolli di rete.
/etc/rpc
Numeri delle chiamate RPC.
/etc/services
Abbinamento dei servizi di rete ai numeri di porta corrispondenti.

La concentrazione amministrativa si attua facendo in modo che le informazioni dei file che interessano siano gestite a partire da un solo nodo. Generalmente, l'utilità del NIS sta nella possibilità di amministrare gli utenti da un'unica origine, facendo in modo che questi vengano riconosciuti in tutti gli elaboratori di un certo «dominio», senza dover essere inseriti effettivamente in ognuno di questi.

Il NIS non utilizza i file amministrativi così come sono, ne crea una copia; queste copie sono denominate «mappe». I file di mappa sono in formato DBM, dove si memorizzano solo coppie di dati: chiave-valore. Per questo motivo, a

seconda della struttura dei file amministrativi originali, si possono generare più mappe differenti.

Quando si attiva il NIS, non si possono più utilizzare i vecchi comandi amministrativi (come passwd, chsh, ecc.), o quantomeno non conviene, perché il NIS non si accorge (autonomamente) dei cambiamenti apportati ai file amministrativi tradizionali. Bisogna utilizzare i comandi specifici del NIS, in modo che i cambiamenti siano annotati immediatamente nelle mappe e poi siano propagati nei file amministrativi normali del server NIS.

passwd.byname
Utenti per nome.
passwd.byuid
Utenti per numero UID.
group.byname
Gruppi per nome.
group.bygid
Gruppi per numero GID.
shadow.byname
Utenti per nome (dal file /etc/shadow).
mail.aliases
Alias di posta elettronica.
hosts.byname
Nodi per nome.
hosts.byaddr
Nodi per indirizzo.
networks.byname
Reti locali per nome.
networks.byaddr
Reti locali per indirizzo.
protocols.byname
Protocolli di rete per nome.
protocols.bynumber
Protocolli di rete per numero.
rpc.byname
Chiamate RPC per nome.
rpc.bynumber
Chiamate RPC per numero.
services.byname
Servizi di rete per nome.

Quando si attiva un servizio NIS in un nodo, in modo che questo renda disponibili le informazioni relative a un gruppo di elaboratori, si deve definire un dominio NIS corrispondente. Questo non ha niente a che fare con i domini utilizzati dal servizio DNS, ma generalmente, anche se potrebbe sovrapporsi perfettamente a un dominio di questo tipo, conviene utilizzare nomi distinti, che non abbiano un nesso logico o intuitivo.

Finora si è fatto riferimento a un server NIS unico per tutto il suo dominio di competenza. Quando si attiva un servizio di questo tipo, tutti gli elaboratori client di questo dominio dipendono completamente dal server per tutte quelle informazioni che sono state concentrate sotto la sua amministrazione. Se l'elaboratore che offre questo servizio dovesse venire a mancare per qualsiasi motivo, come un guasto, tutti i suoi clienti sarebbero in grave difficoltà. Per risolvere il problema, si possono predisporre dei server NIS secondari, o slave, che riproducono le informazioni del server principale, o master.

Il client NIS è un programma demone che si occupa di fornire al sistema in cui è in funzione le informazioni che altrimenti verrebbero ottenute dai soliti file di configurazione. La situazione tipica è quella della procedura di accesso: se il nome dell'utente non viene trovato nel file /etc/passwd locale, il client NIS cerca di ottenerlo dal server NIS. I client cercano le informazioni, riferite al loro dominio, dal server che risponde più rapidamente. Ciò viene determinato generalmente attraverso una richiesta circolare (broadcast). Questo, tra le altre cose, è uno dei punti deboli del NIS: dal momento che qualunque elaboratore può rispondere a una chiamata circolare, chiunque è in grado di intromettersi per cercare di catturare delle informazioni.

Gli elaboratori che devono condividere le informazioni amministrate con il NIS, devono utilizzare il demone ypbind, configurato opportunamente. In tal modo, su tali elaboratori, invece di utilizzare le informazioni amministrative locali, vengono usate quelle concentrate dal NIS.

La configurazione di ypbind avviene attraverso i file /etc/yp.conf e /etc/nsswitch.conf. Il primo serve a definire come raggiungere i server; il secondo definisce l'ordine di utilizzo dei servizi (Name service switch).

Come nel caso dei server, anche i client richiedono la definizione del dominio NIS, attraverso domainname. Se il dominio non viene predisposto ypbind non può funzionare.

Il client richiede la presenza della directory /var/yp/. Al suo interno viene creata la directory binding/. Il client richiede l'attivazione del Portmapper RPC.

A seconda delle caratteristiche particolari del client, sono possibili delle configurazioni speciali per ciò che riguarda l'accesso da parte degli utenti. Quando la loro gestione è compito del NIS, si può configurare il client in modo da definire una graduatoria nella ricerca dei dati che identificano l'utente al momento dell'accesso. Di solito si cerca prima l'utente nel file /etc/passwd locale, quindi si prova con il NIS.

A parte questo particolare abbastanza semplice, si può porre il problema di voler concedere l'accesso su un certo elaboratore solo ad alcuni utenti definiti attraverso il NIS, oppure, più semplicemente, si può volere escludere l'accesso da parte di qualcuno. Per ottenere questo occorre intervenire sul file /etc/passwd utilizzando record con notazioni particolari.

In generale, per fare in modo che gli utenti NIS del dominio a cui si fa riferimento possano accedere da un certo client, occorre aggiungere in coda un record speciale nei file /etc/passwd, /etc/group e /etc/shadow:

/etc/passwd

+::::::

/etc/group
+:::
/etc/shadow
+::::::
Questo record viene interpretato come il punto in cui si vogliono inserire virtualmente gli utenti NIS.

1.5.3 Autenticazione in un dominio AD Microsoft

Una macchina Linux puo' essere integrata in un dominio Microsoft Active directory per mezzo di Samba e agire sia come shared server che come Domain Controller.

Qui pero' prendiamo in considerazione l'autenticazione di utenti locali della macchina, effettuata per mezzo dei servizi centralizzati dell'Actice Directory. L'autenticazione in un dominio Microsoft avviene per mezzo del protocollo kerberos, basandosi sui dati di una directory LDAP.

I sistemi piu' moderni (per es. RedHat e derivati) permettono la configurazione in modo estramamente semplice per mezzo dell'utilita' *authconfig*.

La procedura che vedremo comporta invece la modifica manuale di una certa quantita' di file di configurazione. L'elemento centrale a cui viene affidata l'opera di autenticazione e' il demone winbindd della suite Samba, che assieme ad una libreria per nsswitch e ad un modulo PAM permettono di far interagire i due mondi:

- Creare un file di configurazione di Samba (/etc/smb.conf) per far entrare la nostra macchina a far parte del dominio come shared server:

```
[global]
workgroup name = UNISIENA
encrypt password = yes
security = domain
password server = *
winbind separator = +
winbind uid = 10000-20000
winbind gid = 10000-20000
winbind enum users = yes
winbind enum groups = yes
template shell = /bin/bash
template homedir = /home/%D/%U
```

Da notare che lo winbind separator viene impostato al valore + invece del classico \. Le due direttive template pemettono di definire la shell e la home directory degli utenti (i globals Samba %D e %U indicano rispettivamente il dominio e il nome utente).

- Modificare il file /etc/nsswitch.conf. Esso contiene la configurazione del nsswitch, un servizio (collegato alla glibc) per la risoluzione di tutta una

serie di nomi nel sistema, in genere affidato a file statici. Occorre aggiungere un campo alle linee passwd e group:

passwd: files winbind
group: files winbind

- Modificare la configurazione di PAM. E' l'operazione piu' pericolosa, in quanto un errore puo' comportare l'impossibilita' di accedere al sistema. Per prima cosa si modifichi /etc/pam.d/system-auth aggiungendo la riga:

auth sufficient /usr/lib/security/pam_winbind.so
e sostituendo:
auth sufficient /lib/security/pam_unix.so likeauth nullok
con
auth sufficient /lib/security/pam_unix.so likeauth nullok use_first_pass
Poi nel file /etc/pam.d/login aggiungere la prima e ultima riga:
account sufficient /lib/security/pam_winbind.so
session required /lib/security/pam_mkhomedir.so skel=/etc/skel/ umask=0022
Quest'ultima permette di creare al volo la directory dell'utente quando viene fatto il login per la prima volta.

1.6 Avvio, arresto e riavvio del sistema

L'avvio del sistema prevedeva fino al kernel 2.4 alcune varianti. Attualmente e' invece essenziale l'uso di un boot loader, perche' il kernel stesso non contiene piu' un boot sector. I boot loader piu' utilizzati sono LiLo (Linux Loader) e GRUB (GRand Unified Bootloader), con il secondo che tende sempre piu' a prevalere sul primo.

Il boot loader carichera' il kernel, che si proccupera' montare la root partition e attivare (processo init) tutti i processi che conducono alla schemata di login.

L'ambiente che verra' preparato e presentato dipende dal numero di runlevel passato al processo init. Senza scendere, per ora nei dettagli, diciamo che i piu' comuni sono in genere il 3 (console testuale) o 5 (login grafico). Ma possono essere ridefiniti da una distribuzione all'altra. Quello che in genere e' standard sono i runlevel previsti per lo shutdown (0) e il reboot (6).

Per l'arresto del sistema il comando che si usa e' *shutdown* con l'opzione *-h*. Shutdown esegue l'arresto in maniera ordinata, per prima cosa notificando (via wall) a tutti gli utenti loggati dell'imminente chiusura del sistema e bloccando la possibilita' di fare login. Dopo di che, se e' stato specificata la parola chiave "now", invia tutti i processi il segnale SIGTERM (15) che permette ai processi aperti di chiudersi in modo ordinato. Shutdown puo' essere anche invocato con un certo ritardo, con la possibilita' di allertare tutti gli utenti ad intervalli regolari dell'ora della chiusura del sistema.

Per compiere il proprio lavoro shutdown invoca init, con il runlevel 0, appunto quello dedicato alla arresto del sistema.

Esistono dei frontend a shutdown, per la precisione i comandi *halt* e *poweroff*. Essi si premurano di scrivere in /var/log/wtmp (il record di tutti i login e

logout del sistema) che il sistema sta per essere arrestato e poi, se il sistema non si trova gia' nel runlevel 0 oppure 6, invocano shutdown con l'opzione -h. Poweroff viene in genere utilizzato per inviare anche il comando di spegnimento all'alimentatore. Esistono una serie di macchine con BIOS buggati che non riescono a spegnere l'alimentatore.

I comandi di halt e poweroff sono in genere SUID a root, per cui ogni utente potrebbe spegnere il sistema. Al contrario nessun utente in genere puo' utilizzare shutdown.

Per il riavvio del sistema si utilizza in genere il comando shutdown con l'opzione *-r*. Per esso vale tutto quanto detto prima, col l'accoretezza di sostituire il runlevel 0 con il 6.

Anche in questo caso esiste un frontend dal nome *reboot*.

1.7 Il processo init

In ogni sistema Unix, al termine del caricamento del kernel viene montata la partizione di root e viene invocato il processo init (in /sbin, /bin o /usr/sbin). Se questo eseguibile non e' presente, il kernel notifica un errore e il sistema si blocca. E' possibile specificare al boot loader un programma alternativo da lanciare, ma questo non fa parte della comune prassi.

Init e' il processo 1 di tutti i sistemi Unix e rimane in vita per tutto il periodo di funzionamento del sistema. I suoi compiti sono molteplici. Per prima cosa esso e' parent di tutti gli altri processi e in particolare e' necessario per la gestione degli zombie process. Il ruolo primario e' pero' quello di creare i processi che vanno a "costruire" il sistema seguendo le direttive di un file */etc/inittab*.

Uno dei concetti base legati al processo init e' quello di *runlevel*. Un runlevel e' una configurazione software del sistema in cui e' permesso esistere solo ad un ristretto gruppo di processi. I processi che sono avviati dal processo init per ogni runlevel sono definiti nel file /etc/inittab. Il runlevel puo' essere cambiato a runtime da un utente privilegiato per mezzo del comando *telinit*. In genere i runlevel 0,6, e 1 sono riservati rispettivamente per l'arresto del sistema, il reboot del sistema e la modalita' singolo utente (per questo esiste anche il runlevel S, che e' utilizzato soprattutto dagli script). I runlevel 7-9 sono validi sebbene non documentati.

Il funzionamento del processo init e' il seguente: al termine del boot il processo init viene invocato, esso cerca il file /etc/inittab e va a vedere se esiste la direttiva *initdefault*, che determina il runlevel iniziale del sistema, se questa entry non esiste, un runlevel dovra' essere specificato alla console.

I runlevel S e s portano il sistema in single user mode, ovvero una root shell viene aperta su /dev/console.

Se l'accesso viene fatto ad una modalita' multi utente per la prima volta, init cerca le direttive boot e bootwait per montare i filesystem prima che gli utenti possano loggarsi. Dopo di che tutte le direttive relative al runlevel sono eseguite.

Dopo aver avviato tutti i processi specificati, init attende la morte di uno dei processi discendenti, un segnale di powerfail oppure un segnale inviato tramite

telinit per cambiare runlevel. Se una di queste condizioni si verifica, il file /etc/inittab viene riesaminato.

Qualora il file /etc/inittab venga modificato e si vogliano notificare i cambiamenti a init occorre usare il comando:

telinit q

Se init non si trova in single user mode e riceve un segnale di powerfail (SIGPWR), esso va a leggere il file /etc/powerstatus e fa partire un comando in base al contenuto del file:

- F(AIL) Il sistema sta lavorando sotto UPS. Vengono eseguite le entry *powerwait e powerfail.*
- O(K) La tensione e' tornata. Viene eseguita la entry *powerokwait.*
- L(OW) Il sistema e' alimentato da UPS e la batteria rimanente e' poca. Viene eseguita la entry *powerfailnow.*

L'uso di questa tecnica e' al momento sconsigliato, in favore dell'utilizzo del canale di controllo /dev/initctl.

Quando a init viene richiesto un cambio di runlevel, esso invia il segnale SEGTERM a tutti i processi non definiti nel nuovo runlevel. Dopo 5 secondi invia il segnale SIGKILL che termina forzatamente i processi. Init presuppone che questi processi siano nello stesso process group che avevano in origine, altrimenti devono essere terminati separatamente.

1.7.1 Il file /etc/inittab

Abbiamo visto che il file che il processo init va a consultare e' /etc/inittab. Il formato di questo file e' piuttosto particolare e merita di essere approfondito. Una entry del file ha il seguente formato:

id:runlevels:action:process

I commenti sono introdotti da #. Altri sistemi Unix, non GNU based introducono invece i commenti con il carattere :. I vari campi della linea hanno i seguenti significati:

- ID: e' una sequenza univoca di 1-4 caratteri che identifica la entry. Per i processi getty e di login, il campo deve essere il suffisso della corrispondente tty.
- RUNLEVELS: lista dei runlevel per i quali la specifica azione deve essere compiuta.
- ACTION: Descrive l'azione che deve essere intrapresa.
  - respawn: il processo verra' riavviato non appena termina.
  - wait: il processo viene avviato una volta per lo specifico runlevel e si aspetta che sia terminato.
  - once: il processo viene eseguito una volta per lo specifico runlevel.

  - boot: il processo viene eseguito durante il boot.
  - bootwait: il processo viene eseguito durante il boot e se ne attende la fine.
  - off: nessuna operazione.
  - ondemand: il processo sara' eseguito solo per i runlevel on-demand
  - initdefault: specifica il runlevel a cui si dovrebbe accedere dopo iol boot (il campo process viene ignorato).
  - sysinit: Il processo verra' eseguito durante il boot prima delle entry boot e bootwait.
  - powerwait
  - powerfail
  - powerokwait
  - powerfailnow
  - ctrlaltdel: Il processo verra eseguito se init riceve il segnale SIGINT, ovvero la combinazione di tasti CTRL-ALT-DEL premuta sulla console.
  - krbrequest: il processo sara' eseguito se init riceve un messaggio dal keyboard handler che una particolare combinazione di tasti e' stata premuta.

- PROCESS: specifica il processo che deve essere eseguito. Se il campo processo inizia con il carattere +, init non fara' accounting per il processo su wtmp e utmp.

1.8 Altri sistemi operativi liberi della famiglia Unix

Fra i vari sistemi operativi liberi, appartenenti al ramo principale di sviluppo Unix (quello che deriva da Sys V e BSD) vale la pena di citare tre sistemi operativi che rivestono particolare importanza, soprattutto nell'ambito delle reti. L'ultima release di BSD (4.4BSD-lite release 2) datava 1995 ed era del tutto priva del codice che aveva portato AT&T ad intraprendere un azione legale. Fin da quel momento una serie di nuove release erano in fase di sviluppo, in particolare:

- **FreeBSD**: e' un sistema operativo capace di girare su piattaforme Intel e AMD (a 32 e 64 bit), DEC Alpha, Sun UltraSparc e PowerPC. Esso nasce come un sistema operativo completo, nel senso che viene sviluppato sia il kernel che le userland utilities. Il modello di sviluppo di FreeBSD prevede il mantenimento di due release contemporanee: la Stable e la Current. La prima dedicata alla produzione e l'altra allo sviluppo. L'organizzazione del software di FreeBSD e' costituita dai cosiddetti ports. FreeBSD offre la compatibilita' binaria con una serie di Unix, in particolare Linux. FreeBSD e' naturalmente coperto dalla licenza BSD che permette a chiunque di usare e redistribuire il codice a proprio piacimento.

- **NetBSD**: e' nato come un progetto parente di FreeBSD con in mente un modello di sviluppo maggiormente aperto. Il punto di forza di NetBSD e' il supporto di un amplissimo numero di piattaforme (al momento 54) e architetture di processore, tutte dal medesimo ramo CVS di sviluppo. Lo sviluppo dei device driver e' quindi indipendente dalla piattaforma hardware, questa caratteristica viene sfruttata a pieno nell'implementazione dei sistemi embedded. Come per FreeBSD, anche NetBSD fornisce la compatibilita' binaria con una serie di Unix. Anche in questo caso la licenza con cui e' rilasciato il progetto e' quella BSD. Il software nativo di NetBSD e' organizzato tramite *pkgsrc*.

- **OpenBSD**: e' nato da un fork di NetBSD ponendo come obiettivi il software open source e la qualita' della documentazione, oltre al fulcro centrale del progetto che e' la sicurezza. E' un sistema che fornisce una quantita' di security features che nessun altro sistema operativo puo' vantare. E' in grado di operare su 16 diverse piattaforme. Come per gli altri BSD lo sviluppo prevede sia il kernel che le applicazioni userland, ma il modello di sviluppo implica che solo codice open source possa essere utilizzato, quindi non sono presenti driver per i quali dovesse essere siglato un Non-Disclosure Agreement. La licenza di OpenBSD e' sempre stata quella BSD (oppure MIT), ma ultimamente si sta preferendo per il nuovo software la ISC license, che permette ancora piu' liberta nel limiti della convenzione di Berna. Il codice rilasciato sotto GPL non viene accettato nello sviluppo (salvo rarissimi casi). Gli sviluppatori di OpenBSD dedicano moltissimo tempo all'auditing del codice, alla ricerca di bug e potenziali falle. Questa continua ricerca porta OpenBSD a dichiarare: *"Only one remote hole in the default install, in more than 10 years."*

References

[1] Appunti di informatica libera - Appunti Linux Copyright © 1997-2000 Daniele Giacomini Appunti di informatica liberaCopyright © 2000-2006 Daniele Giacomini Via Morganella Est, 21 – I-31050 Ponzano Veneto http://na.mirror.garr.it/mirrors/appuntilinux/HTML/a2.htm

[2] Integrare Linux in un dominio WinNT/2000 con Winbind - http://openskills.info/infobox.php?ID=863

[3] Matthias Kalle Dalheimer, Matt Welsh - Running Linux (5th edition) (O'Reilly)

[4] Man Pages - http://www.tldp.org/docs.html#man

[5] FreeBSD - http://www.freebsd.org/

[6] OpenBSD - http://www.openbsd.org/

[7] NetBSD - http://www.netbsd.org

Contents