

Modulo 2:

Controllo degli accessi e autenticazione

Controllo degli accessi in Linux

1 Utenti in Linux

In Linux gli utenti del sistema hanno un loro *user account* (o *username*) che serve per identificarli in fase di *login*. Di default Linux dispone di un unico utente, cioè l'utente `root` che è l'amministratore del sistema ed ha accesso a tutte le risorse.

In fase di creazione, ad un utente viene associato un numero di 16 bit chiamato *user identity (UID)*. Alcuni numeri sono riservati e predefiniti, ad esempio `0` per `root` e i numeri da `1` a `99` per servizi standard. Si tratta di un valore unico per ogni utente. Inoltre gli utenti appartengono ad uno o più gruppi. In particolare, ogni utente appartiene ad un gruppo primario. Ad ogni gruppo viene associato un numero di 16 bit chiamato *group identity (GID)*.

Le informazioni relative ad un utente sono contenute nei file `/etc/passwd` e nel file `.profile` nella home dell'utente, mentre il file `/etc/group` contiene una lista di tutti i gruppi. La struttura delle righe di questo file è semplice:

`nomeGruppo : passwordGruppo :GID: listaUtenti`

- `nomeGruppo` è il nome utilizzato per identificare il gruppo.
- `passwordGruppo` è la password cifrata. Di solito non viene utilizzata e di conseguenza non viene inserita, ma viene inserito un asterisco (*) al suo posto.
- `GID` è il numero identificativo del gruppo.
- `listaUtenti` è la lista degli utenti che appartengono al gruppo. Si tratta di un elenco di nomi di utente separati da virgolette.

La possibilità di raggruppare gli utenti in gruppi permette una migliore gestione del controllo degli accessi. Ad esempio è più facile esprimere i permessi per una categoria di utenti o per una tipologia di servizio. Per esempio si può fare in modo che tutti gli utenti che possono accedere alla posta elettronica facciano parte di un gruppo `mail`.

1.1 Comandi per la gestione di utenti: creazione o rimozione un utente

Per questi comandi è necessario essere amministratore. La creazione di un utente si ottiene con il comando `useradd`, mentre la rimozione di un utente si ottiene con il comando `userdel`.

1.2 Autenticazione degli utenti

Gli utenti sono identificati dal loro *username* e autenticati tramite una *password*.

I file delle password sono il file `/etc/passwd` e il file `/etc/shadow`. In passato esisteva un unico file quindi si parlava di *unshadowed password* in quanto il file era leggibile da tutti perché conteneva altre informazioni relative all'utente. Nella

nuova modalità con due file le informazioni relative agli utenti sono separate da quelle relative a *salt* e *password*, quindi il file `/etc/shadow` è leggibile solo da `root`.

Una *entry* nel file `/etc/passwd` ha la seguente struttura:

nomeUtente : passwordUtente :UID:GID:datiPersonalI:directoryHome:shell

- `nomeUtente` è il nome utilizzato per identificare l'utente.
- `passwordUtente` è la password cifrata. Se questa indicazione manca, l'utente può accedere senza indicare alcuna password. Se questo campo contiene un asterisco (*) o una x, indica che la password si trova memorizzata (cifrata) nel file `/etc/shadow`.
- `UID` è il numero identificativo dell'utente (User ID).
- `GID` è il numero identificativo del gruppo a cui appartiene l'utente (Group ID).
- `datiPersonalI` è il nominativo completo dell'utente (nome e cognome).
- `directoryHome` è la directory assegnata all'utente.
- `shell` è la shell assegnata all'utente, che viene avviata dopo il login.

Il file `/etc/shadow`, detto delle *shadowed password* è, come già detto, un file leggibile solo da `root`. Il file contiene solo il *salt* in chiaro e gli *hash* delle password (ed eventualmente informazioni relative alla data di ultima modifica, intervallo di tempo per modificare password, etc.)

Nella versione *unshadowed* del file delle password sono possibili attacchi *off-line* di tipo dizionario o a forza bruta. Infatti un qualsiasi utente può leggere e salvarsi in locale il file delle password, calcolare *hash(salt+parola)* per ogni parola nel dizionario e confrontare il risultato con l'*entry* nel file associata al *salt* che è stato concatenato alla parola del dizionario. Il *salt* rende i dictionary attack più difficili in quanto in assenza di *salt* basterebbe confrontare *hash(parola)* con l'*hash* di tutte le password.

2 Controllo degli accessi

Linux adotta una politica *discrezionale* basata su identificatore utente (UID) e identificatore di gruppo (GID). Ad ogni file (quindi ad ogni risorsa) sono associati un UID e GID che corrispondono a quelli del processo che ha creato il file.

Ogni file ha associata una lista di nove permessi: `read (r)`, `write (w)`, `execute (x)` definiti a livello di *utente*, *gruppo* e *altri*. Come già detto, l'amministratore del sistema (`root`) ha tutti i permessi su tutti i file.

In particolare, per i file:

- `r`: permette di leggere il file;
- `w`: permette di scrivere il file;
- `x`: permette l'esecuzione del file.

Mentre per le directory:

- `r`: permette di leggere il contenuto delle directory (elenco dei file);
- `w`: permette di modificare il contenuto delle directory (creare o rinominare file all'interno di una directory);
- `x`: permette l'attraversamento della directory.

Il comando `ls` usato con le opzioni `al` (quindi con la sintassi `ls -al`) ha la funzione di stampare a video l'elenco dei file (anche quelli nascosti) e delle cartelle presenti nella posizione corrente e delle informazioni relative ai file, in particolare vengono stampati la colonna dei permessi, il nome del proprietario e quello del gruppo primario.

La colonna dei permessi contiene è composta da una stringa di dieci caratteri (si veda Figura 1 per un esempio):

```

cartella_esempio — bash — 80x24
Chiaras-MacBook-Pro:cartella_esempio chiara$ ls -al
total 8
drwxr-xr-x  3 chiara  staff  102 Feb 21 08:37 .
drwxr-xr-x+ 23 chiara  staff  782 Feb 21 07:51 ..
-rw-r--r--  1 chiara  staff   8 Feb 21 08:37 readme.txt
Chiaras-MacBook-Pro:cartella_esempio chiara$ 

```

Figura 1: Esempio di utilizzo del comando `ls -al`

- il primo spazio indica la tipologia dell'elemento e può avere i seguenti valori: `d` (*directory*), `l` (*link simbolico*), `-` (*file*);
- i seguenti nove spazi indicano i permessi; più precisamente si tratta di tre distinti gruppi di 3 permessi (`r` = lettura; `w` = scrittura; `x` = esecuzione). Il primo gruppo da tre riguarda il proprietario, il secondo riguarda il gruppo ed il terzo riguarda gli altri utenti. `-` indica l'assenza del permesso corrispondente.

La valutazione dei permessi al momento di una richiesta avviene nel seguente ordine:

- Se lo UID indica che il richiedente è il proprietario del file, i bit relativi al proprietario vengono considerati;
- Se il richiedente NON è il proprietario del file ma il GID è quello del file, i bit relativi al gruppo vengono considerati;
- Se i due punti precedenti non sono validi, vengono considerati i bit relativi agli altri utenti.

2.1 Permessi: rappresentazione numerica

I privilegi possono essere rappresentati in forma numerica tramite numeri ottali dividendo i 9 permessi in gruppi di tre, come specificato in Figura 2. Questa diversa rappresentazione è utile quando si devono modificare i permessi di un file, come specificato in sezione 2.3.

Cifra ottale	Testuale	Binario	Significato
0	---	000	accesso negato
1	--x	001	execute
2	-w-	010	write
3	-wx	011	write e execute
4	r--	100	read
5	r-x	101	read e execute
6	rw-	110	read e write
7	rwx	111	qualsiasi accesso

Figura 2: Permessi: rappresentazione numerica

2.2 Privilegi addizionali

Ci sono tre tipi di permessi speciali che sono disponibili per i file eseguibili e per le directory pubbliche e sono `Setuid`, `Setgid` e `Sticky bit`. In genere i primi due privilegi vengono usati per permettere ad utenti non privilegiati di eseguire particolari programmi con i privilegi dell'amministratore. Ad esempio, il privilegio `setuid` al comando `mount` permette anche agli utenti normali di montare dei filesystem residenti su memorie di massa rimovibili, come le chiavi USB o il CD-ROM.

Nel caso in cui `Setuid` (Set User ID) sia settato, un utente esegue il file con i privilegi dell'utente proprietario del file, ovvero la `UID` del processo che esegue un file coincide con `UID` del file. In questo caso, la `x` dei permessi associati all'utente viene sostituita da:

- `s` se `setuid` è settato e se l'utente ha il privilegio di esecuzione sul file (es. `rws-----`)
- `S` se `setuid` è settato e se l'utente non ha il privilegio di esecuzione sul file (es. `r-S-----`)

Nel caso di `Setgid` (Set Group ID), il `GID` del processo che esegue un file coincide con il `GID` del file. In questo caso, la `x` dei permessi associati al gruppo viene sostituita da:

- `s` se `setgid` è settato e se il gruppo ha il privilegio di esecuzione sul file (es. `rwxrws---`)
- `S` se `setgid` è settato e se il gruppo non ha il privilegio di esecuzione sul file (es. `rwxr-S---`)

Nel caso lo `Sticky bit` sia settato solo il proprietario della directory (e `root`) possono rimuovere o rinominare i file contenuti nella directory. Viene comunemente usato per le directory destinate a contenere file temporanei di più utenti. In questo caso la `x` dei permessi associati ad altri viene sostituita da:

- `t` se sticky bit è settato e se altri hanno il privilegio di esecuzione sul file (es. `rwxrwxrwt`)
- `T` se sticky bit è settato e se altri non hanno il privilegio di esecuzione sul file (es. `rwxrwxr-T`)

2.3 Comandi per la gestione dei privilegi

2.3.1 Variare i permessi

Il comando per variare i bit di protezione (permessi) è `chmod [u g o] [+ -] [rwx] nomefile`, dove `[rwx]` specifica quali permessi siano concessi (+) o negati (-), `[u g o]` specifica a chi devono essere modificati i permessi, ovvero:

- `u`: utente proprietario del file
- `g`: gruppo proprietario
- `o`: other
- `a`: tutti gli utenti indifferentemente

I permessi possono essere concessi o negati dal solo proprietario del file (o da `root`).

Esempi:

- `chmod 666 file`

Assegna a tutti permessi `r` e `w`.

- `chmod -R go-rwx ./*`

Toglie al gruppo e agli altri utenti la possibilità di scrittura, lettura ed esecuzione della directory corrente e delle sotto-directory (opzione `-R`).

- `chmod -R a+rwx ./*`

Assegna a tutti gli utenti il permesso di lettura ed esecuzione partendo dalla directory corrente e considerando le sotto-directory.

2.3.2 Cambiare il proprietario

Il comando per cambiare proprietario ad un file è `chown [opzioni] [utente] [:gruppo] file`

Esempi:

- `chown andrew myfile`

Cambia l'utente proprietario del file `myfile` impostando l'utente `andrew`.

- `chown andrew:users myfile`

Cambia l'utente e il gruppo del file `myfile` nell'utente `andrew` e nel gruppo `users`, rispettivamente.

- `chown -R from=root andrew miadir`

Cambia il proprietario dei file appartenenti all'utente `root` nell'utente `andrew`.

2.3.3 Cambiare gruppo al file

Il comando per cambiare gruppo al file è `chgrp [opzioni] gruppo file ...`

Esempi:

- `chgrp users file`

Imposta il gruppo `users` associato al file `file`.

- `chgrp -R users /home/andrew`

Imposta in modo ricorsivo il gruppo `users` associato ai file contenuti nella directory e sottodirectory indicata.

2.3.4 Il comando `umask`

Ogni volta che viene creato un file, vengono impostati dei permessi di default, impostati secondo una maschera predefinita. In genere permessi per un nuovo file sono `666`. La maschera di default dei permessi viene impostata tramite il comando `umask` e definisce i permessi che **non** vengono attribuiti. Ad esempio, con il comando `umask 022` non vengono attribuiti permessi scrittura ed esecuzione né al gruppo né ad altri utenti. Lanciando il comando senza specificare un parametro maschera viene mostrato sullo standard output il valore corrente.

L'opzione `-s` mostra il valore corrente dei permessi effettivi espresso in notazione simbolica invece che numerica. L'opzione `-p` mostra il valore di negazione dei permessi corrente impostato da `umask`. Se si desidera lasciare invariati i propri permessi, negandoli completamente agli utenti del gruppo e a tutti gli altri utenti, si usi `umask 077`.

Si noti che i permessi di ogni file o directory creati dipendono sia dai permessi richiesti dal programma che li ha creati che dall'impostazione data tramite `umask`. Quando un file viene creato usando `cp`, i permessi del file sono derivati da `umask`. Quando un file viene creato usando `mv` (quindi semplice ridefinizione), i permessi del file sono gli stessi del file di origine.

Valori comunemente usati per `umask` sono:

- `002` per impedire la scrittura agli utenti che non sono proprietari del file e che non fanno nemmeno parte del gruppo assegnato al file;
- `022` per impedire la scrittura a utenti diversi dal proprietario (valore di default);
- `044` per impedire la lettura a utenti diversi dal proprietario;
- `077` per impedire lettura, scrittura ed esecuzione a utenti diversi dal proprietario.