Administración de los usuarios

1. Fundamentos

a. Identificación y autenticación

La **identificación** consiste en saber quién es quién para determinar los permisos de la persona que se conecta. Se identifica un usuario mediante un login.

La **autenticación** consiste en aportar la prueba de quiénes somos mediante, por ejemplo, un secreto compartido entre el usuario y el sistema, y que sólo conocen ellos. Se autentifica, generalmente, al usuario con una contraseña, aunque también podría ser una clave.

b. Los usuarios

Un usuario es la asociación de un nombre de conexión, el login, con un UID y al menos un GID.

- **UID**: User ID.
- **GID**: Group ID.

Los UID y los GID suelen ser únicos. El login es único. Sin embargo, se puede considerar la asociación de varios logins al mismo UID, sabiendo que el sistema trabaja a veces con el login.

El UID identifica el usuario (o la cuenta asociada del servicio) a lo largo de su conexión. Se utiliza para el control de sus derechos y de los de los procesos que ha iniciado. Lo que se almacena en la tabla de los inodos, dentro de la tabla de los procesos, son los UID y GID, y no los logins.

El usuario dispone de los atributos básicos siguientes:

- un nombre de inicio de sesión llamado login;
- una contraseña;
- un UID;

- un GID correspondiente a su grupo principal;
- una descripción;
- un directorio de inicio de sesión;
- un comando de inicio de sesión.

Hay otros atributos disponibles mediante la utilización de la seguridad de las contraseñas en el archivo shadow (ver apartado correspondiente).

Se suelen asociar los UID de un valor inferior a 100 a cuentas especiales con derechos extendidos. Así el UID de root, el administrador, es 0. Según las distribuciones, a partir de 100, 500 o 1000, y hasta 65.535 (2^{16}-1) aproximadamente son los UID de los usuarios sin privilegios particulares. La mayoría de los Unix, incluido Linux, pueden usar UID en 32 bits firmados, el límite sería de más de cuatro mil millones (las empresas que tienen más de 65535 empleados ya no tendrán ese problema). Estos parámetros pueden modificarse en **`etc/login.defs`** .

Un login tiene en principio un tamaño de 8 caracteres. En realidad, Linux y otros sistemas aceptan un tamaño mayor, pero con la mayoría de los comandos la visualización de los logins, incluso su gestión, está limitada a 8 caracteres. La talla máxima se puede obtener de la manera siguiente:

```
$ getconf LOGIN_NAME_MAX
256
```

Un login acepta la mayoría de caracteres. No debe empezar por una cifra. Es posible modificar la lista de los caracteres autorizados y forzar la longitud y la complejidad mediante los mecanismos de autenticación PAM y el archivo **`/etc/login.defs`** .

c. Los grupos

Cada usuario forma parte de al menos un grupo. Un grupo agrupa usuarios. Como para los logins, el GID del grupo siempre acompaña al usuario para controlar sus privilegios. Un usuario puede formar parte de varios grupos. En este caso, hay que distinguir su grupo primario de los grupos secundarios. Sin embargo, un grupo no puede formar parte de otro grupo.

Los grupos son también números (GID). Existen grupos específicos para la gestión de algunas propiedades del sistema y, en particular, el acceso a ciertos periféricos.

Cada vez que un usuario crea un archivo, éste recibe como valor de grupo con privilegios el primario del creador. Si el usuario alejandro tiene como grupo primario **users**, entonces los archivos creados por alejandro tendrán como grupo con privilegios **users**.

Un usuario dispone de todos los derechos asociados a sus grupos secundarios. Si seb tiene como grupo secundario **vídeo** y un archivo dispone de los derechos de escritura para este grupo, entonces alejandro tendrá derecho a modificar su contenido.

El comando **id** permite conocer la información esencial sobre un usuario: uid, gid, grupos secundarios.

```
$ id alejandro
uid=1000(alejandro) gid=100(users) grupos=100(users),16(dialout),3(sys),33(video)
```

Alejandro ha creado un archivo. Su propietario es alejandro y su grupo es el grupo primario de alejandro: users.

```
$ touch test
$ ls -l test
-rw-r--r-- 1 alejandro users 0 abr 10 14:30 test
```

Para conocer los tipos de codificación autorizados en su distribución, consulte el manual de la función `crypt (man crypt)` .

d. Las contraseñas

Las contraseñas permiten autenticar a los usuarios. Deben ser lo bastante complejas como para que no se pueden descubrir fácilmente, pero lo bastante intuitivas como para que se puedan recordar. Las contraseñas están codificadas (SHA-256, SHA-512, Blowfish, por ejemplo) y no son directamente legibles bajo su forma cifrada por el usuario para que nadie pueda intentar descifrarlas.

Un usuario debería cambiar regularmente su contraseña, no escribirla nunca en ninguna parte ni llevarla encima. Podría, sin embargo, usar un gestor de contraseñas como **keepass** para almacenarlas. Luego veremos cómo se puede obligar a un usuario a aplicar reglas de creación y duración de contraseñas.

Veamos por ejemplo el resultado encriptado por SHA-512 (reconocible por el 6 que empieza la cadena) de una contraseña:

> contraseña
> 6nrZeMJZT$GCCJz86zIiyBH3O88bQ8YwypCsQ153Wi/
> tgjgv5EbaVCL.N1efycL6wj45GwhXWSyXOjuX0CIbVCjSDfmUQOh.

Para conocer los tipos de codificación autorizados en su distribución, consulte el manual de la función `crypt` (`man 3 crypt`).

2. Los archivos

a. /etc/passwd

El archivo **`/etc/passwd`** contiene la lista de los usuarios del sistema local. Cualquier usuario puede leerlo. La información que contiene es pública y útil tanto para el sistema como para los usuarios. Cada línea representa un usuario y se compone de siete campos.

```
Login:password:UID:GID:comment:homedir:shell
```

- Campo 1: el login o nombre de usuario.
- Campo 2: en las antiguas versiones, la contraseña cifrada. Si hay una x, se coloca la contraseña en **`/etc/shadow`** . Si es un signo de exclamación, se bloquea la cuenta.
- Campo 3: el User ID.
- Campo 4: el GID, o sea, el grupo principal.
- Campo 5: un comentario o descripción. Es un campo de información.

- Campo 6: el directorio de trabajo, personal, del usuario. Es el directorio al que llega cuando se conecta.
- Campo 7: el shell por defecto del usuario. Pero puede ser cualquier otro comando, incluso un comando que prohíbe la conexión.

b. /etc/group

El archivo `/etc/group` contiene la definición de los grupos de usuarios y en cada uno, la lista de los usuarios de los cuales es el grupo secundario. Cada línea se compone de cuatro campos:

```
Group:password:GID:user1,user2...
```

- Campo 1: el nombre del grupo.
- Campo 2: la contraseña asociada. Veremos la explicación justo a continuación.
- Campo 3: el Group Id.
- Campo 4: la lista de usuarios que forman parte de este grupo.

Es inútil colocar en el cuarto campo a los usuarios que tienen este grupo como grupo principal; lo maneja el sistema.

Le puede sorprender la presencia de un campo de contraseña para los grupos. En la práctica se utiliza pocas veces. Como por supuesto es imposible iniciar sesión como grupo, la explicación está en otra parte. Un usuario tiene derecho a cambiar de grupo para coger, de manera temporal al menos, un grupo secundario como grupo primario con el comando **newgrp**.

En este caso, el administrador puede establecer una contraseña para el grupo para proteger el acceso a este grupo como grupo principal.

c. /etc/shadow

El archivo `/etc/shadow` acompaña al archivo `/etc/passwd` . Ahí se almacenan, entre otras cosas, las contraseñas cifradas de los usuarios. Para ser más precisos, contiene toda la información sobre las contraseñas y su validez en el tiempo. Cada línea se

compone de 9 campos separados por ":":

```
bean:$2a$10$AjADxPEfE5iUJcltzYA4wOZO.f2UZ0qP/8EnOFY.P.m10HifS7J8i:13913
:0:99999:7:::
```

- Campo 1: login.
- Campo 2: contraseña cifrada. El xx inicial indica el tipo de cifrado.
- Campo 3: número de días desde el 1º de enero de 1970 hasta el último cambio de contraseña.
- Campo 4: número de días sin poder cambiar la contraseña (0: se puede cambiar en cualquier momento).
- Campo 5: número de días a partir de los cuales se debe cambiar la contraseña.
- Campo 6: número de días antes del vencimiento de la contraseña durante los cuales se debe avisar al usuario.
- Campo 7: número de días después del vencimiento de la contraseña tras los cuales se desactiva la cuenta.
- Campo 8: número de días desde el 1º de enero de 1970 hasta el momento en el cual se desactivó la cuenta.
- Campo 9: reservado.

En el ejemplo de la línea bean, se ha cambiado la contraseña 13913 días después del 01/01/1970. La contraseña se debe cambiar antes de 0 días, pero seguirá siendo válida, ya que el campo siguiente indica que hay que cambiarla al cabo de 99999 días (273 años) y el campo 5 está vacío (sin obligación de cambio de contraseña). La cuenta se desactiva tras 7 días: no hay riesgo de que ocurra...

Los valores actuales para el cifrado de contraseñas son los siguientes:

- **1**: MD5
- **$2a$**: Blowfish (no es estándar)
- **$5**: SHA-256
- **$6**: SHA-512
- Otro: DES

Para conocer la fecha en función del 01/01/1970, utilice el comando **date** como a continuación, añada el número de días deseado:

```
$ date --date "01/01/1970 +18748days"
sáb may 1 00:00:00 CEST 2021
```

d. /etc/gshadow

El archivo `/etc/gshadow` es el equivalente del archivo anterior, pero para los grupos. Sin embargo, algunas de las anteriores distribuciones de Linux no lo soportan por defecto. Se colocan las contraseñas de los grupos en el segundo campo de `/etc/group` .

```
test:$6$FBuQQ/oCSnY/PLbx$MD...rtYrADveathy6xO/8ZYfIpAMF72OZKHUGwJMBox/:root:alejandro
```

El formato de gshadow es el siguiente:

- Campo 1: nombre del grupo.
- Campo 2: contraseña codificada (cifrada). El xx inicial indica el tipo de cifrado, SHA-512 en este caso.
- Campo 3: administradores del grupo (los que pueden modificar la contraseña o los miembros del grupo).
- Campo 4 : miembros del grupo: pueden acceder sin contraseña. Es la misma lista que en /etc/group.

3. Gestión de los usuarios

a. Creación

La creación de un usuario podría ser totalmente manual, ya que Linux (como cualquier Unix) se apoya en una serie de comandos que “sólo” modifican archivos planos ya existentes y que crean y vuelven a copiar archivos y carpetas en el sitio correcto con los privilegios correctos.

La creación de un usuario consiste en:

- añadir una línea en **`/etc/passwd`**,
- añadir una línea en **`/etc/shadow`**,
- añadir una información si es preciso en **`/etc/group`**,
- crear el directorio personal y colocar el contenido de la carpeta **`/etc/skel`**,
- cambiar los permisos y el propietario del directorio personal,
- cambiar la contraseña (cifrada).

Se puede crear directamente una cuenta editando los archivos con un editor, aunque no es nada aconsejable. Si desea hacerlo de todas maneras, utilice el comando **vipw**, que actualizará las diversas cachés asociadas a la gestión de las cuentas.

El comando **vipw** admite cuatro argumentos:

- `-p`: edición de **`/etc/passwd`**.
- `-g`: edición de **`/etc/group`**.
- `-s`: edición de **`/etc/shadow`**.
- `-gs`: edición de **`/etc/gshadow`**.

Evidentemente, en la práctica hay que evitar editar estos archivos a mano, ya que hay otros muchos mecanismos creados a partir del uso de comandos específicos para la creación o modificación de cuentas, como por ejemplo, la comprobación de la complejidad de la contraseña.

Cabe destacar que existe una restricción asociada al formato del login. Responde a una norma POSIX: sólo puede empezar por un carácter alfabético, seguido de caracteres portables (letras, cifras, guión, guión bajo, etc.).

Todo esto se puede llevar a cabo con el comando **useradd**. Añade una nueva cuenta y efectúa las principales operaciones:

- creación del usuario y edición de los archivos;
- creación de un grupo privado del usuario (con el mismo nombre que éste);
- creación del directorio personal, edición y modificación de los derechos.

```
# useradd <options> login
```

Si no se ha especificado ninguna opción, se recuperarán los valores por defecto dentro del archivo **`/etc/default/useradd`** . Se aceptan las opciones principales siguientes:

Opción	Función
-m	Crea también el directorio personal. Se incluye a veces por defecto, pero es mejor comprobar que el directorio personal esté presente después de la utilización del comando si no utiliza esta opción.
-u	Especifica el UDI numérico del usuario, para forzarlo. Dicho de otro modo, se calcula el UID según las reglas del archivo login.defs y los UID existentes.
-g	Especifica el grupo primario del usuario, por GID o por su nombre (variable GROUP).
-G	Especifica los grupos adicionales (secundarios, del usuario) separados por comas (variable GROUPS).
-d	Ruta del directorio personal. En general /home/<login>, pero no se puede especificar cualquier ruta (variable HOME/<login>).
-c	Un comentario asociado a la cuenta. Puede ser cualquiera, pero a veces algunos comandos como **finger** lo pueden utilizar. El usuario puede modificar su contenido con el comando **chfn**.
-k	Ruta del directorio que contiene el esqueleto del árbol del directorio del usuario. Suele ser /etc/skel (variable SKEL).
-s	Shell (intérprete de comandos) por defecto del usuario (variable SHELL). El usuario puede cambiarlo mediante el comando **chsh**.
-p	La contraseña del usuario. ¡Cuidado! ¡La contraseña debe estar ya cifrada! A no ser que se vuelva a copiar la contraseña de una cuenta genérica, lo mejor es usar el comando **passwd**.

El comando siguiente crea la cuenta "roberto" con la mayoría de las opciones básicas especificadas. Es sólo un ejemplo, excepto a veces la `-m`, ya que si no se especifica nada, son las opciones por defecto en relación con las especificadas en el archivo **`/etc/defaults/useradd`**.

```
# useradd -m -u 1010 -g users -G video,dialout,lp -s /bin/bash -d
/home/roberto -c "Cuenta de Roberto" roberto
# grep roberto /etc/passwd
robert:x:1010:100:Cuenta de Roberto:/home/roberto:/bin/bash
```

El comando no crea la contraseña. Hay que hacerlo a mano con el comando **passwd**.

```
# passwd roberto
Changing password for roberto.
Nueva contraseña:
Vuelva a introducir la nueva contraseña:
Contraseña cambiada.
```

Si desea generar usted mismo una contraseña ya codificada (para useradd o usermod), puede utilizar la línea de comando siguiente. El primer parámetro es la contraseña en texto claro, la segunda un “salt” que debera ser muy largo y aleatorio:

```
$ echo $(perl -e'print crypt("foobar", "\$6\$salttogenerate\$")')
$6$salttogenerate$2y5GK3joHstsKC0CKulp19OtPVn2mQpC1hWhinU1vUu/sFFcQdFa/
RaqZ32wiGtHpl3dyBvrXSYpNj4GqcWTI0
```

b. Seguridad de las contraseñas

Cambiar la contraseña

El comando **passwd** permite gestionar las contraseñas, pero también las autorizaciones de inicio de sesión, así como la mayoría de los campos presentes en **`/etc/shadow`**.

Cualquier usuario tiene derecho a cambiar su contraseña en el plazo especificado por el campo 4 de /etc/shadow. La acción por defecto consiste en cambiar la contraseña del

usuario actual. Se requiere la antigua contraseña por seguridad (en particular para evitar que una persona mal intencionada modifique su contraseña a sus espaldas). La inserción está enmascarada.

```
$ id
uid=1000(alejandro) gid=100(users) ...
$ passwd
Changing password for alejandro.
Antigua contraseña:
Nueva contraseña:
Vuelva a introducir la nueva contraseña:
Contraseña cambiada.
```

Los módulos **PAM** (*Pluggable Authentication Module*) pueden imponer exigencias más o menos estrictas para el cambio de contraseña: en cuanto a la longitud, que no se base en una palabra del diccionario, etc. Veamos lo que ocurre cuando se quiere utilizar pepe (demasiado corto), qwerty (demasiado largo) y María (diccionario):

```
$ passwd
Changing password for alejandro.
Antigua contraseña:
Nueva contraseña:
Contraseña incorrecta: demasiado corta
Nueva contraseña:
Contraseña incorrecta: demasiado simple
Nueva contraseña:
Contraseña incorrecta: basada en una palabra del diccionario
passwd: Número máximo de intentos agotado para el servicio
```

El usuario root tiene derecho a modificar las contraseñas de todos los usuarios del sistema, sin que sea preciso que conozca la contraseña anterior. Aún mejor: puede forzar el uso de una contraseña incluso aunque no haya sido validada por PAM:

```
# passwd alejandro
Changing password for alejandro.
Nueva contraseña:
```

> Contraseña incorrecta: basada en una palabra del diccionario
> Vuelva a introducir la nueva contraseña:
> Contraseña cambiada.

Gestionar la validez

Se pueden modificar todos los campos de `/etc/shadow` con el comando **passwd**. Veamos algunas de las opciones disponibles.

Opción	Función
-l	Lock: bloquea una cuenta al añadir un ! delante de la contraseña cifrada.
-u	Unlock: desbloquea la cuenta. No se puede activar una cuenta que no tenga contraseña. Hay que utilizar -f para ello.
-d	(root) Suprime la contraseña de la cuenta.
-n <j>	(root) Duración de vida mínima en días de la contraseña.
-x <j>	(root) Duración de vida máxima en días de la contraseña.
-w <j>	(root) Número de días antes de un aviso.
-i <j>	(root) Período de gracia antes de la desactivación si ha vencido la contraseña.
-S	(root) Estatus de la cuenta.

En el ejemplo siguiente, se ha modificado la cuenta bean de esta manera:

- Debe esperar 5 días tras insertar una nueva contraseña para poder cambiarla.

- Su contraseña es válida 45 días.
- Se le avisa 7 días antes de que deba cambiar la contraseña.
- Si no cambia la contraseña tras 45 días, dispone aún de 5 días antes de que sea bloqueada.

```
# passwd -n 5 -x 45 -w 7 -i 5 bean
Password expiry information changed.
```

Veamos la línea de `/etc/shadow` asociada.

```
bean:$6$Avbxum6ZnqwPEun0$gESuKjswfet6qZIBr8oH8FMukfRmAygrwOrJOz39MepA
LuRWiEqJvK2rKw.vMxjIWhTULZpq/pI3B.GcYHgvG1:18748:5:45:7:5::
```

El comando **chage** permite hacer más o menos lo mismo. Es un comando del root. Iniciado sin otro argumento que el login del usuario, es interactivo. Observe al final la posibilidad de modificar la fecha del último cambio de la contraseña y una fecha fija de expiración de la contraseña (campo 8):

```
# chage bean
Changing the aging information for bean
Enter the new value, or press ENTER for the default

    Minimum Password Age [5]:
    Maximum Password Age [45]:
    Last Password Change (YYYY-MM-DD) [2017-01-28]:
    Password Expiration Warning [7]:
    Password Inactive [5]:
    Account Expiration Date (YYYY-MM-DD) [-1]: 2020-05-15
```

Veamos la línea `/etc/shadow` resultante:

```
bean:$6$Avbxum6ZnqwPEun0$gESuKjswfet6qZIBr8oH8FMukfRmAygrwOrJOz39MepALu
RWiEqJvK2rKw.vMxjIWhTULZpq/pI3B.GcYHgvG1:18748:5:45:7:5:18762:
```

Se aceptan los parámetros siguientes:

Opción	Función
-m	Mindays: equivale a passwd -n.
-M	Maxdays: equivale a passwd -x.
-d	Fecha de última modificación de la contraseña (desde el 01/01/1970).
-E	Fecha de vencimiento de la contraseña (desde el 01/01/1970).
-I	Inactive: equivale a passwd -i.
-W	Warndays: equivale a passwd -w.
-l	List: muestra todos los detalles.

Los detalles son mucho más legibles con **chage** que con **passwd**:

```
# passwd -S bean
bean P 05/01/2021 5 45 7 5

# chage -l bean
Last password change                                    : may 01, 2021
Password expires                                        : jun 15, 2021
Password inactive                                       : jun 20, 2021
Account expires                                         : may 15, 2021
Minimum number of days between password change          : 5
Maximum number of days between password change          : 45
Number of days of warning before password expires       : 7
```

Un usuario cualquiera puede visualizar sus propios detalles, pero se le podría pedir su contraseña.

c. Modificación

Utilice el comando **usermod** para modificar una cuenta. Utiliza la misma sintaxis y las mismas opciones que useradd, pero dispone también de una sintaxis complementaria que necesita algunas precisiones.

Opción	Función
-L	Bloquear la cuenta, como passwd -l.
-U	Desbloquar la cuenta, como passwd -u.
-e <n>	Vencimiento: la contraseña expira n días después del 01/01/1970.
-u <UID>	Modificar el UID asociado al login. Se modifica en consecuencia el propietario de los archivos que pertenecen al antiguo UID dentro del directorio personal.
-l <login>	Modificar el nombre de login.
-m	Move: implica la presencia de -d para especificar un nuevo directorio personal. Se mueve el contenido del antiguo directorio al nuevo.

d. Eliminación

Suprime un usuario con el comando **userdel**. Por defecto no se suprime el directorio personal. Para ello, debe pasar la opción -r.

```
# userdel -r bean
```

El comando no comprueba la totalidad del sistema de archivos, los archivos que estén fuera de su directorio personal no se borrarán. Es el administrador quien debe buscarlos y borrarlos. Para ello, puede usar el comando **find**.

4. Gestión de los grupos

a. Creación

Puede crear un grupo directamente en el archivo `/etc/group` o bien utilizar los comandos asociados. Si edita el archivo manualmente, utilice el comando **vigr** (o vipw -g).

El comando **groupadd** permite crear un grupo. Su sintaxis sencilla acepta el argumento `-g` para especificar un GID preciso.

```
# grep amigos /etc/group
amigos:!:1234:
```

b. Modificación

El comando **groupmod** permite modificar un grupo. Sus parámetros son los siguientes:

Opción	Función
-n <nombre>	Renombra el grupo.
-g <GID>	Modifica el GID. Cuidado: no se modifica el grupo al que pertenecen los archivos correspondientes.

```
# groupadd grp1
# groupmod -n grp2 grp1
# grep grp /etc/group
grp2:x:1003:
```

c. Eliminación

El comando **groupdel** permite suprimir un grupo. Primero el comando comprueba si el grupo que desea suprimir es el grupo primario de un usuario. En este caso, no se permite suprimir el grupo.

Pero si finalmente se puede, no se efectúa más acción que la de suprimir la línea correspondiente en /etc/group. Le corresponde a usted comprobar el sistema de archivos (y la configuración de las aplicaciones si es necesario) para suprimir cualquier traza de este grupo.

```
# groupdel amigos
```

d. Contraseña

El comando **gpasswd** permite asignar una contraseña a un grupo. Esta contraseña se almacena en /etc/group, o, en el caso más frecuente, en /etc/gshadow. Modifique la contraseña del grupo test según este modelo:

```
gpasswd test
```

Agregue un usuario al grupo:

```
@ gpasswd -a alejandro test
```

Defina la lista de todos los miembros del grupo:

```
# gpasswd -M alejandro, steph test
```

Defina la lista de administradores del grupo, los que podrán gestionar los usuarios y sus contraseñas:

```
# gpasswd -A admin test
```

Si proporciona una contraseña a un grupo, todos los usuarios que conozcan esta contraseña, incluidos los que no forman parte del grupo, podrán usarla. El grupo se convierte en su grupo principal.

```
$ id -gn
uid=1001(steph) gid=1002(steph) grupos=1002(steph)
$ newgrp test
Contraseña:
$ id
uid=1001(steph) gid=1001(test) grupos=1002(steph),1001(test)
```

El acceso al grupo puede restringirse limitando a sus miembros con el parámetro -R. Es decir, los miembros del grupo no tendrán que introducir la contraseña.

5. Comandos adicionales

a. Conversión de los archivos

Seguramente empleará pocas veces los comandos siguientes. Son útiles sobre todo en el contexto de migraciones de servidores Linux hacia otros Unix, y viceversa. Algunos sistemas Unix no utilizan por defecto (hay que activarla después) la gestión de las cuentas con los archivos **shadow**. En este caso, puede ser necesario convertir los archivos `/etc/shadow` y `/etc/passwd` en uno único, `/etc/passwd` . Es la función del comando **pwunconv**.

En el ejemplo siguiente, se ha convertido el archivo `/etc/passwd` . Una vez ejecutado el

comando, cualquier traza de `/etc/shadow` ha desaparecido.

```
# pwunconv
# grep bean /etc/passwd
bean:$2a$10$dwbUGrC75bs3l52V5DHxZefkZyB6VTHsLH5ndjsNe/vF/HAzHOcR2:1001:100:
toto:/home/bean:/bin/bash
# ls -l /etc/shadow
ls: no puede acceder a /etc/shadow: Ningún archivo o directorio de este tipo
```

Cuidado: el comando **pwunconv** es destructivo. Toda información de la validez de las contraseñas será destruida. Ya no es posible utilizar las diversas opciones de chage o de passwd relativas a los periodos de validez.

El comando **pwconv** hace lo contrario: crea el archivo `/etc/shadow` asociado a `/etc/passwd` , le mueve dentro las contraseñas y pone los ajustes por defecto, tal y como están definidos en el archivo `/etc/login.defs` .

```
# grep bean /etc/passwd
bean:x:1001:100:pepito:/home/bean:/bin/bash
p64p17bicb3:/home/alejandro # grep bean /etc/shadow
bean:$2a$10$dwbUGrC75bs3l52V5DHxZefkZyB6VTHsLH5ndjsNe/vF/HAzHOcR2
:13984:0:99999:7:::0
```

Los comandos **grpconv** y **grpunconv** hacen lo mismo para los grupos.

b. Verificar la coherencia

Puede ser útil iniciar herramientas de comprobación de la coherencia de los archivos de los grupos y de las contraseñas. Si está acostumbrado a modificar estos archivos manualmente, nada le asegura que todo esté en orden: puede que falte un grupo, que no

exista un shell, que un directorio personal esté ausente, etc. El comando **pwck** efectúa una verificación de los archivos `/etc/passwd` y `/etc/shadow` y reporta los errores.

Veamos un ejemplo en el cual se han alterado los datos del usuario bean de manera voluntaria para probar el comando: el grupo no existe, tampoco el shell. Entre tanto, se ha descubierto otro problema en la máquina de prueba.

```
# pwck
Checking `/etc/passwd'
User `suse-ncc': directory `/var/lib/YaST2/suse-ncc-fakehome' does
not exist.
User `bean': unknown group `14400'
User `bean': shell `/bin/bashr' is not executable.
Checking `/etc/shadow'.
```

El comando **grpck** hace lo mismo para los grupos. En este caso, los controles están menos extendidos y se limitan a los duplicados y a la existencia de los usuarios para los grupos secundarios.

```
# grpck
Checking `/etc/group'
Group `users': unknown user `zorg'
```

c. Comprobar las conexiones

Puede trazar las conexiones en su máquina usando dos comandos. El comando **lastlog** se basa en el contenido de /var/log/lastlog. Acepta los parámetros `-u` (especificación de un usuario) y `-t` para buscar las conexiones de los últimos n días.

```
# lastlog -u alejandro
Username         Port     From             Latest
alejandro        pts/0    192.168.1.22     sáb may 1 08:31:17 +0200 2021
```

El comando **last** hace más o menos lo mismo, pero se basa en `/var/log/wtmp`, que brinda información adicional, como el origen de la conexión (IP, nombre de la consola, etc.)

y las fechas de conexión y desconexión, así como la duración de conexión y si el usuario sigue conectado. Observará en el ejemplo los eventos de reinicio que indican la versión del núcleo (5.5.2), el que se compiló en el capítulo Inicio de Linux, servicios, núcleo y periféricos.

```
# last
alejandro    pts/0     192.168.1.22      Thu Apr 29 09:25   still logged in
alejandro    :0        :0                Thu Apr 29 09:25   still logged in
reboot system boot    5.3.0-29-generic   Thu Apr 29 09:24   still running
alejandro    pts/1     192.168.1.22      Thu Apr 29 09:18 - 09:23  (00:05)
alejandro    :0        :0                Thu Apr 29 09:15 - down   (00:08)
reboot system boot    5.5.2              Thu Apr 29 09:14 - 09:24  (00:09)
reboot system boot    5.5.2              Thu Apr 29 09:11 - 09:24  (00:12)
alejandro    pts/0     192.168.1.22      Wed Apr 28 21:33 - 22:25  (00:51)
...
```

d. Acciones de usuario

El usuario dispone de algunas acciones sobre las informaciones de su cuenta. Entre otras cosas, puede:

- cambiar su shell de conexión,
- cambiar sus datos personales,
- cambiar de grupo principal,
- tomar la identidad de otra persona.

Cambiar de shell

El comando **chsh** permite al usuario modificar de manera definitiva (o hasta el próximo comando **chsh**) el shell de conexión. No puede elegir cualquier cosa. El shell (o cualquier otro comando) debe estar en `/etc/shells` . A esta lista se accede mediante el parámetro `-l` del comando. La modificación se hace dentro de `/etc/passwd` . Sólo root tiene derecho a modificarla para otros usuarios. Se especifica el nuevo shell con el parámetro `-s`.

```
$ cat /etc/shells
/bin/sh
/bin/bash
/usr/bin/bash
/bin/rbash
/usr/bin/rbash
/bin/dash
/usr/bin/dash
/usr/bin/tmux
/usr/bin/screen
...
$ id
uid=1004(bean) gid=100(users) ...
$ chsh -s /bin/ksh
Changing login shell for bean.
Contraseña:
Shell changed.
# grep bean /etc/passwd
bean:x:1004:100:pepito:/home/bean:/bin/ksh
```

Cambiar el comentario

El usuario puede modificar el comentario del archivo `/etc/passwd` usando el comando **chfn**. Es preferible utilizarlo de manera interactiva (se reserva a root el paso de parámetro en modo no interactivo).

```
$ chfn
Changing finger information for bean.
Contraseña:
Enter the new value, or press ENTER for the default
    Full Name: pepito
    Room Number []: Mister Bean
    Work Phone []: 0102030405
    Home Phone []: 0605040302
    Other:
Finger information changed.
$ grep bean /etc/passwd
bean:x:1004:100:Mister Bean,0102030405,0605040302:/home/bean:/bin/bash
```

Cambiar de grupo primario

El comando **newgrp** permite cambiar de grupo primario de manera temporal, con la condición de que el nuevo grupo especificado sea un grupo secundario del usuario o que el usuario disponga de la contraseña del grupo. Utilizado solo, **newgrp** vuelve al grupo de origen. Las modificaciones son temporales, no se modifica el archivo de las contraseñas.

```
$ id
uid=1004(bean) gid=100(users) grupos=16(dialout),33(video),100(users)
$ newgrp video
$ id
uid=1004(bean) gid=33(video) grupos=16(dialout),33(video),100(users)
```

¿Qué ocurre si intenta coger como grupo primario un grupo que no pertenece a sus grupos secundarios y que está protegido por una contraseña? En el ejemplo siguiente, bean intenta coger como grupo primario grptest.

```
$ id
uid=1004(bean) gid=100(users) grupos=16(dialout),33(video),100(users)
$ newgrp grptest
Password:
$ id
uid=1004(bean) gid=1000(grptest) grupos=16(dialout),33(video),
100(users),1000(grptest)
```

Cambiar de identidad

El usuario puede adoptar la identidad de otra persona mientras dura un comando o toda una sesión. Se suele tratar de root, ya que, como sabe, uno nunca debe conectarse de manera permanente como root (o al menos hay que evitarlo). Por lo tanto, para las tareas administrativas, hay que convertirse en root (u otro usuario) el tiempo necesario.

El comando **su** (*substitute user*) permite abrir una sesión, o ejecutar un shell, o un comando dado, con otra identidad. Obviamente, debe conocer la contraseña de este usuario.

```
su [-c comando] [-s shell] [-] [usuario]
```

Si no se especifica ningún usuario, se utiliza root.

```
$ id
uid=1000(alejandro) gid=100(users) ...
$ su
Contraseña:
# id
uid=0(root) gid=0(root) grupos=0(root)
```

Observe que se utiliza por defecto el entorno del usuario de origen. No se carga el entorno de root (o de cualquier otro usuario).

```
# echo $LOGNAME $USER
alejandro alejandro
```

Para cargar el entorno completo del usuario de destino, añada el guión como parámetro:

```
$ su - bean
Contraseña:
$ echo $USER $LOGNAME
bean bean
```

Para ejecutar de manera puntual un comando con otra identidad, utilice `-c`.

```
$ su -c "make install"
```

El comando **sg** (*substitute group*) es idéntico a **su**, pero emplea un nombre de grupo como argumento.

El comando mágico

En muchos de los ejemplos del libro, los comandos se ejecutan como root. Normalmente,

y este tema se abordará en el capítulo sobre la seguridad, deberá efectuar el mínimo posible de acciones como administrador. Cuando sea necesario adquirirá los privilegios de root, usando el comando su, pero también el comando sudo, de manera temporal. El siguiente comando le permite simplemente cambiar a root:

```
$ sudo su -
```

e. Consultar al sistema

El comando **getent** permite consultar al sistema pasando por las librerías NSS (*Name Service Switch*) sin pasar por los archivos ni los comandos vistos antes. Es muy útil porque los archivos locales no son la única fuente de identificación y autenticación. Su servidor podrá, de esta forma, consultar un directorio LDAP o NIS, o una base de datos Active Directory. En ese caso, otras cuentas serán utilizables en su sistema.

Un error habitual consiste en verificar si un usuario o un grupo existe en el sistema efectuando consultas en los archivos /etc/passwd y /etc/group sin tener en cuenta la información remota. El autor encontró este problema con algunos productos de muy buena reputación (docker), pero por fortuna antes de que estos fueran modificados adecuadamente.

Los comandos para los usuarios y grupos son:

```
getent passwd
getent shadow
getent group
getent gshadow
```

He aquí el retorno de `getent passwd` en un servidor conectado a un directorio donde los UID superiores a 10000 se encuentran reservados para las cuentas LDAP:

```
$ getent passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

```
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
...
charlie.chaplin:*:10116:2012:Charlie Chaplin:/home/charlie.chaplin:/bin/bash
bruno.mars:*:10134:2012:Bruno mars:/home/bruno.mars:/bin/bash
jackie.chan:*:10104:2012:Jackie Chan:/home/jackie.chan:/bin/bash
daft.punk:*:10154:2012:Daft Punk:/home/daft.punk:/bin/bash
```

6. Configuración avanzada

Muchos comandos, como, por ejemplo, **login**, **useradd**, **groupadd**, **passwd**, **su**, utilizan el archivo **`/etc/login.defs`** para definir algunos valores por defecto y la validez de los logins. Su contenido puede variar en función de las distribuciones. Suele contener:

- una regla de validez de las cuentas (caracteres autorizados, longitud, etc.);
- los UID mínimo y máximo durante la creación de un usuario;
- los GID mínimo y máximo durante la creación de un grupo;
- los comandos que hay que llamar para crear/modificar/eliminar un usuario;
- las reglas por defecto para la validez de las contraseñas;
- la creación o no de un directorio personal;
- etc.

El contenido del archivo login.defs puede entrar en conflicto con algunas opciones de los demás archivos presentes en /etc/default. En este caso, debe comprobar en los manuales (o manpages) de su distribución qué archivo prevalece.

```
# Login autorizado incluso si el acceso al home es imposible
DEFAULT_HOME       yes

# Path por defecto para el comando login
ENV_PATH           /usr/local/bin:/usr/bin:/bin

# Ídem para la conexión root
ENV_ROOTPATH       /sbin:/bin:/usr/sbin:/usr/bin

# Plazo en segundos entre dos intentos de login
FAIL_DELAY       3

# Los usuarios en el archivo no ven los mensajes de cnx
HUSHLOGIN_FILE    /etc/hushlogins

# Visualiza la fecha de última conexión
LASTLOG_ENAB       yes

# Se trazan los intentos de conexión de los logins inexistentes
LOG_UNKFAIL_ENAB    no

# tres intentos de login en caso de contraseña incorrecta
LOGIN_RETRIES       3

# Timeout de 60 segundos
LOGIN_TIMEOUT       60

# Ubicación del motd (o de las , separadas por :)
MOTD_FILE    /etc/motd

# tipo de terminales por defecto
TTYTYPE_FILE    /etc/ttytype

# Permiso por defecto de los terminales
TTYGROUP    tty
TTYPERM       0620

# Requiere o no una contraseña para chsh y chfn
CHFN_AUTH       yes
```

```
# Restricción del chfn (f:nombre, r:oficina, w:tel trabajo, h:tel casa)
CHFN_RESTRICT       rwh

# PASS_MAX_DAYS    Duración max de la contraseña
# PASS_MIN_DAYS    Plazo mín entre dos cambios de contraseña
# PASS_WARN_AGE    Aviso antes del cambio
PASS_MAX_DAYS     99999
PASS_MIN_DAYS     0
PASS_WARN_AGE     7

# UID/GID Mín y Máx por defecto y sistema para useradd
SYSTEM_UID_MIN         100
SYSTEM_UID_MAX         499
UID_MIN             1000
UID_MAX             60000

#
# Min/max values for automatic gid selection in groupadd
#
# SYSTEM_GID_MIN to SYSTEM_GID_MAX inclusive is the range for
# GIDs for dynamically allocated administrative and system groups.
# GID_MIN to GID_MAX inclusive is the range of GIDs of dynamically
# allocated groups.
#
SYSTEM_GID_MIN         100
SYSTEM_GID_MAX         499
GID_MIN             1000
GID_MAX             60000

# Regexp para los nombres de logins autorizados mediante login/useradd
CHARACTER_CLASS       [A-Za-z_][A-Za-z0-9_.-]*[A-Za-z0-9_.$-]\?

# Umask por defecto para la creación del homedir
UMASK           022

# Comando ejecutado realmente durante la creación de un grupo
GROUPADD_CMD           /usr/sbin/groupadd.local

# Comando ejecutado realmente durante la creación de un usuario
USERADD_CMD       /usr/sbin/useradd.local
```

```
# Comando ejecutado antes de la supresión de un grupo
USERDEL_PRECMD      /usr/sbin/userdel-pre.local

# Comando ejecutado después de la supresión de un grupo
USERDEL_POSTCMD      /usr/sbin/userdel-post.local
```

Además de los comandos **useradd** y **userdel**, le ofrecemos una lista no exhaustiva de los comandos que utilizan los parámetros de este archivo:

- **login**: DEFAULT_HOME, ENV_PATH, ENV_ROOTPATH, FAIL_DELAY, HUSHLOGIN_FILE, LASTLOG_ENAB, LOG_UNKFAIL_ENAB, LOGIN_RETRIES, LOGIN_TIMEOUT, MOTD_FILE, TTYPERM, TTYTYPE_FILE;
- **newusers**: PASS_MAX_DAYS, PASS_MIN_DAYS, PASS_WARN_AGE, UMASK;
- **passwd**: OBSCURE_CHECKS_ENAB, PASS_MAX_LEN, PASS_MIN_LEN, PASS_ALWAYS_ WARN, CRACKLIB_DICTPATH, PASS_CHANGE_TRIES;
- **pwconv**: PASS_MAX_DAYS, PASS_MIN_DAYS, PASS_WARN_AGE.

7. Notificaciones al usuario

a. /etc/issue

Cuando un usuario se conecta por consola, se suele visualizar un mensaje justo antes de la línea de comandos de inserción de su login. El archivo `/etc/issue` contiene este mensaje. Se trata de un mensaje de bienvenida y por este motivo puede contener todo lo que desee. Por defecto, suele contener el nombre de la distribución Linux y el número de versión del núcleo. Admite secuencias de caracteres para poderle aplicar sustituciones.

Veamos el ejemplo del contenido de `/etc/issue` en una distribución Ubuntu 16.10:

```
$ cat issue
Ubuntu 19.10 \n \l
```

b. /etc/issue.net

El mensaje de bienvenida puede ser diferente cuando un usuario se conecta desde una consola remota (telnet, ssh, etc.). A menudo es el mismo, pero sin los caracteres de control relacionados con un shell dado. Para modificar este mensaje específico, edite el contenido del archivo **`/etc/issue.net`** .

c. /etc/motd

Motd significa *Message of the day*, el mensaje del día. Un vez conectado el usuario desde una consola (local o remota), puede aparecer un mensaje que el administrador puede modificar si edita el archivo **`/etc/motd`** . Por defecto está vacío. Se puede modificar para, por ejemplo, avisar a los usuarios de que un reboot de mantenimiento tendrá lugar tal día a tal hora, y eso evita mandar n mails...

Las distribuciones como Ubuntu utilizan mensajes de bienvenida muy completos e interactivos. El directorio **`/etc/update-motd.d`** contiene los shell scripts que serán ejectuados bajo demanda por un módulo PAM (ver la sección Resumen general de PAM, en este mismo capítulo) durante una conexión.

```
# ll
total 52
drwxr-xr-x 2 root root 4096 abr 27 08:50 ./
drwxr-xr-x 90 root root 4096 may 1 08:43 ../
-rwxr-xr-x 1 root root 1220 ago 27 2019 00-header*
-rwxr-xr-x 1 root root 1157 ago 27 2019 10-help-text*
lrwxrwxrwx 1 root root 46 abr 27 08:50 50-landscape-sysinfo ->
/usr/share/landscape/landscape-sysinfo.wrapper*
-rwxr-xr-x 1 root root 4650 ago 27 2019 50-motd-news*
-rwxr-xr-x 1 root root 97 ago 11 2018 90-updates-available*
-rwxr-xr-x 1 root root 299 ene 28 2016 91-release-upgrade*
-rwxr-xr-x 1 root root 165 ago 8 2019 92-unattended-upgrades*
-rwxr-xr-x 1 root root 129 ago 11 2018 95-hwe-eol*
-rwxr-xr-x 1 root root 111 nov 13 2018 97-overlayroot*
-rwxr-xr-x 1 root root 142 ago 11 2018 98-fsck-at-reboot*
-rwxr-xr-x 1 root root 144 ago 11 2018 98-reboot-required*
```

Observe el contenido del archivo 50-landscape-sysinfo:

```
# cat 50-landscape-sysinfo
#!/bin/sh
# pam_motd does not carry the environment
[ -f /etc/default/locale ] && . /etc/default/locale
export LANG
cores=$(grep -c ^processor /proc/cpuinfo 2>/dev/null)
[ "$cores" -eq "0" ] && cores=1
threshold="${cores:-1}.0"
if [ $(echo "`cut -f1 -d ' ' /proc/loadavg` < $threshold" | bc) -eq 1 ]; then
   echo
   echo -n " System information as of "
   /bin/date
   echo
   /usr/bin/landscape-sysinfo
else
   echo
   echo " System information disabled due to load higher than $threshold"
fi
```

Cuando un usuario inicie sesión, se mostrará:

```
System information as of sáb may 1 08:31:16 CEST 2021

  System load:  0.0               Processes:           99
  Usage of /:  10.2% of 31.37GB   Users logged in:     1
  Memory usage: 7%                IP address for ens18: 192.168.1.100
  Swap usage:   0%
```

d. wall, write y mesg

El comando **wall** permite la emisión de una notificación en la consola del conjunto de usuarios conectados. La notificación es o bien un mensaje introducido mediante la entrada estándar o el contenido de un archivo. Cada usuario recibirá este mensaje, al igual que un banner que especifica el emisor y la fecha.

```
$ wall
Hello !
^D
Broadcast Message from alejandro@ubuntu (pts/0)( (Sat May 1 09:03:37 2021)):
Hello !
```

El comando **write** hace lo mismo, pero para un usuario específico.

Los usuarios pueden aceptar o rechazar recibir estos mensajes con el comando **mesg** seguido de y o n.

8. El entorno de usuario

a. /etc/skel

En el momento de crear un usuario y su directorio personal, se establece el entorno de usuario. El entorno contiene, por ejemplo, las variables de entorno, los alias, la ejecución de varios scripts. Se encuentra en archivos cargados en el arranque del intérprete de comandos (shell). Los archivos de configuración de un shell están, a menudo, escondidos (su nombre empieza por un punto):

```
$ ls -aA /etc/skel/
.bash_logout  .bashrc  .profile
```

Durante la creación de una cuenta, se copian los diferentes archivos de configuración desde el contenido del directorio /etc/skel (skeleton) hacia el directorio personal. Si desea modificar los entornos de manera global ANTES de crear los usuarios, puede colocar en **`/etc/skel`** todos los archivos que desea y modificarlos a su conveniencia.

b. Scripts de configuración

El capítulo El shell y los comandos GNU le mostró los diferentes archivos cuyo contenido ejecuta el shell. Durante el proceso de inicio de sesión de un usuario, si el shell por defecto

es bash, se ejecutan los siguientes scripts, en este orden:

- **`/etc/profile`** : define las variables de entorno importantes como PATH, LOGNAME, USER, HOSTNAME, HISTSIZE, MAIL y INPUTRC.
- **`/etc/profile.d/*`** : /etc/profile (y no bash) invoca a todos los scripts presentes en ese directorio. Estos scripts pueden completar la configuración global añadiendo por ejemplo, la configuración de los parámetros de idioma, de los alias globales, etc.
- **`~/.bash_profile`** : está en el directorio de usuario y llama a otro script: **`~/.bashrc`** , que a su vez llama a **`/etc/bashrc`** . En .bash_profile puede definir variables adicionales. Por su parte, en **`~/.bashrc`** podrá determinar alias y funciones. No hay reglas estrictas.
- **`~/.bash_login`** : pocas veces presente, es el siguiente de la lista.
- **`~/.profile`** : es el último script ejecutado a través de un shell de conexión.

Si el shell es interactivo, es decir ejecutado manualmente a través de un script o en línea de comandos, estos archivos reemplazan los anteriores:

- Se utiliza **`/etc/bashrc`** para definir las funciones y alias para todo el sistema y todos los usuarios en bash.
- **`~/.bashrc`** en el directorio del usuario.

Estos dos últimos son generalmente llamados por el archivo **`.bash_profile`** . Esto permite un comportamiento idéntico, ya sea para un shell de conexión o un shell interactivo.

Finalmente, durante la desconexión, se ejecutará el script **`~/.bash_logout`** .

c. Grupos privados y setgid

En cuanto a la seguridad de los usuarios, la política de Red Hat y otras distribuciones asociadas como Fedora consiste en crear para ellos de manera sistemática un grupo privado y asignarles una máscara 002. Así, por defecto se protegen mejor los archivos, ya que ningún grupo puede acceder a los archivos de forma automática: los archivos tienen como grupo con privilegios uno que sólo incluye un miembro.

Dado que un usuario puede formar parte de varios grupos, en teoría puede extender el acceso a sus archivos y directorios con el comando **chgrp**, encargado de cambiar el grupo de un archivo. Del mismo modo, puede cambiar de manera temporal de grupo primario con el comando **newgrp**.

Sin embargo, el usuario suele preferir no "rayarse" y dar todos los permisos con un **chmod** (p. ej.: chmod 777), lo que constituye una brecha en la seguridad.

En este caso, la solución consiste en utilizar el bit **setgid** en un directorio.

- Se define un grupo común a todos los usuarios que deben poder acceder a un directorio y a sus archivos.
- Se crea un directorio cuyo grupo propietario es el grupo común.
- Se dan los derechos al directorio con el umask 002 (`rwxrwxr-x`) u otro, pero con todos los derechos sobre el grupo.
- Se añade al directorio el derecho setgid (s) sobre el grupo.

```
# mkdir dir
# chgrp grpcomun dir
# chmod 2770 dir
```

Cuando se ubica **setgid** en un directorio, todos los archivos creados en este directorio pertenecen al grupo del directorio, y no al grupo del usuario. Así, todos los miembros del grupo común podrán acceder a los datos. Sin embargo, el archivo sigue perteneciendo a su creador.

Asociando el Sticky Bit, se obtiene un buen nivel de protección y acceso a los datos:

- todos los miembros del grupo pueden crear en él archivos y directorios;
- todos los archivos pertenecen de manera automática al mismo grupo;
- sólo los propietarios de los archivos pueden suprimirlos.

```
# chmod 3770 dir
# ls -ld dir
drwxrws--T 2 alejandro grpcomun 1024 2008-04-24 11:39 dir/
```

```
$ ls -l test
-rw-r--r-- 1 alejandro grpcomun 0 2008-04-24 11:43 test
```

9. Resumen general de PAM

PAM (*Pluggable Authentication Modules*) es un conjunto de módulos y una librería que permiten instalar mecanismos de autenticación avanzados para todas las herramientas que necesitan una seguridad aumentada. La autenticación se basa en módulos. Cada módulo puede utilizar mecanismos diferentes para intentar autenticar a un usuario. Uno se basará en una autenticación Unix clásica; otro, en LDAP; un tercero, en Active Directory, y un último, en el reconocimiento de una firma digital. Este mecanismo permite también la comprobación mediante un dongle USB...

El principio es bastante sencillo; la configuración, más complicada. Un programa, que necesita confirmar una autenticación, llama a la librería **`libpam.so`** . Esta librería lee un archivo de configuración en el cual se puede requerir la invocación de varios módulos. Cada módulo al que se llama devuelve verdadero o falso. Según la configuración, verdadero puede ser un requisito para seguir con otro módulo, o ser suficiente para conectarse.

Para saber si un comando utiliza PAM, hay que mirar si está montado con la librería PAM:

```
$ ldd /usr/bin/passwd|grep libpam >/dev/null && echo "Utiliza PAM"
Utiliza PAM
```

Se colocan los archivos de configuración en **`/etc/pam.d`** . Suele existir uno por aplicación que utiliza PAM y que lleva el mismo nombre. Si falta el archivo, se utiliza **`other`** . La sintaxis es la siguiente:

```
Type_module control módulo argumentos
```

Type_module

- **auth**: módulo de autenticación (por ejemplo, petición de login y contraseña).
- **account** : autorización, gestión de cuentas (verificación del usuario para el servicio dado. ¿Se autoriza?).
- **password** : comprobación y actualización de la información de seguridad (p. ej.: ¿sigue siendo válida la contraseña? Si no es así, petición de una nueva contraseña).
- **session** : modificación del entorno del usuario.

control

- **required** : éxito requerido. En caso de denegación, se llama a los módulos restantes a pesar de todo, pero, pase lo que pase, al final PAM devolverá una denegación.
- **requisite** : la denegación termina de manera inmediata la autenticación. El éxito le autoriza a proseguir.
- **sufficient** : un éxito evita los otros módulos. Dicho de otro modo, PAM devuelve ok pase lo que pase en caso de éxito.
- **optional** : se ignora el resultado.

módulo

- **pam_unix.so** : autenticación estándar mediante la función C getpw().
- **pam_env.so** : definición de las variables de entorno.
- **pam_securetty.so** : prohíbe una conexión de superusuario (root) desde un terminal sin suficiente seguridad. Se coloca la lista de los terminales autorizados en /etc/securetty.
- **pam_stack.so** : llama a otro servicio PAM para cargar módulos adicionales.
- **pam_nologin.so** : prohíbe la conexión de usuarios si el archivo **/etc/nologin** está presente. En ese caso, se visualiza su contenido.
- **pam_deny.so** : siempre devuelve una denegación.
- **pam_time.so** : añade las limitaciones horarias de conexión (**/etc/security /time.conf**).

- **`pam_console.so`** : da permisos adicionales a un usuario local.
- **`pam_group.so`** : añade un usuario en algunos grupos definidos en **`/etc/security/group.conf`** .
- **`pam_selinux.so`** : dispone el contexto de seguridad selinux por defecto.
- **`pam_motd.so`** : muestra el mensaje del día en el momento de la conexión. En la práctica, reemplaza el "motd" clásico, como en Ubuntu.

Los parámetros dependen de cada módulo. Veamos un ejemplo en el caso clásico de un inicio de sesión por la consola. En este caso, el shell de conexión llama al comando **login**. Ojo es sólo un ejemplo ¡normalmente es más largo!.

```
$ cat login
#%PAM-1.0
auth       requisite    /lib/security/pam_securetty.so
auth       required     /lib/security/pam_env.so
auth       sufficient   /lib/security/pam_unix.so
auth       required     /lib/security/pam_deny
auth       required     /lib/security/pam_nologin.so
session    optional     pam_motd.so motd=/run/motd.dynamic
```

- La primera línea comprueba si root intenta conectarse desde un terminal sin seguridad (p. ej.: telnet, rlogin, rsh, etc.). Si es así, PAM devuelve directamente falso y la autenticación fracasa.
- La segunda línea carga el entorno del usuario. Un fracaso no impide aquí la ejecución de las líneas siguientes, pero al final PAM devolverá falso.
- La tercera línea intenta una autenticación mediante los mecanismos Unix clásicos (archivo de contraseñas, NIS, etc.). El éxito detiene el proceso en este punto: se autentica directamente al usuario. Si no, se pasa directamente a la línea siguiente.
- La cuarta línea siempre devuelve falso. Como los módulos de autenticación anteriores han fracasado, la conexión del usuario fracasa. A pesar de todo, se ejecuta la línea siguiente.
- Si el archivo **`/etc/nologin`** existe, se muestra.
- La última línea mostrará el contenido del archivo indicado si el usuario ha

conseguido conectarse. El resultado del comando será ignorado en el caso de una conexión fallida.

Es posible prohibir el acceso a una lista de usuarios dados. Coloque los nombres de los usuarios prohibidos en **`/etc/nologinusers`** y añada la línea siguiente en el archivo de configuración PAM. En nuestro ejemplo, habría que añadirla entre la segunda y la tercera línea.

```
auth required /lib/security/pam_listfile.so onerr=suceed item=user
sense=deny file=/etc/nologinusers
```