Control de acceso y privilegios de *root*

En general, los sistemas informáticos son multiusuario: el control de acceso es el conjunto de reglas que se aplican a las combinaciones de usuarios y acciones, determinando quién puede hacer qué.

Control de acceso tradicional en Unix

Desde el principio del desarrollo de Unix, las siguientes reglas determinaron el diseño del sistema:

- Los objetos (ficheros y procesos) tienen propietarios. Los propietarios tienen un control regulado sobre sus objetos.
- Los objetos creados pertenecen a su creador.
- La cuenta especial *root* es propietario de todos los objetos del sistema.
- Sólo *root* puede realizar todas las acciones (en particular, sólo él puede llevar a cabo ciertas labores administrativas delicadas).

En Unix no hay un único sistema de control de acceso, sino que esta funcionalidad se reparte de manera dispersa por el sistema, y repartida entre el núcleo del sistema operativo y el sistema de ficheros.

Control de acceso del sistema de ficheros

El modelo tradicional organiza a los usuarios en grupos[1]; cada fichero tiene un usuario propietario y un grupo propietario, con diferentes permisos de acción para ambos. De este modo, se pueden compartir y limitar las posibilidades de acción sobre el mismo. Los propietarios y permisos de un fichero pueden encontrarse con la orden ls -l:

```
mrhydden@debian:~$ ls -l
total 0
-rw-r--r-- 1 mrhydden nasty_users 0 may 27 01:57 filename
```

Este fichero es propiedad del usuario '*mrhydden*' y del grupo '*nasty_users*'. Las letras y guiones en la primera columna simbolizan los permisos. Vea el documento sobre el sistema de ficheros para más información sobre cómo interpretar esa información.

Tanto el núcleo como el sistema de ficheros representan a los usuarios y grupos por números. La idea básica es que un UID (*User Identifier*, 'Identificador de usuario') se relaciona con una entrada en el fichero /etc/passwd, y un GID (*Group Identifier*, 'Identificador de grupo') se relaciona con una entrada en el fichero /etc/group. Los nombres textuales son únicamente para consumo humano.

Posesión de procesos

El propietario de un proceso puede enviarle señales y reducir su prioridad. Los procesos en general tienen varias identidades:

1 Actualmente la información sobre grupos se suele almacenar más comúnmente en servidores NIS o LDAP en red, en lugar de utilizar la configuración del sistema (en el fichero /etc/group).

- Un UID y un GID reales, utilizados para tareas de contabilidad.
- Un UID y un GID efectivos (EUID y EGID), que determinan los permisos de acceso.
- Un UID y un GID almacenados (*saved*), que permiten restaurar el estado del EUID tras cambiarlo.

En general el EUID es igual al UID y el EGID al GID. Existen otras identidades (*filesystem UID*) que por considerar menos relevantes no comentaré aquí.

La cuenta root

La cuenta *root* representa al superusuario administrador de UNIX, quien es todopoderoso. La cuenta root se caracteriza por un UID de 0. El Unix tradicional se caracteriza por permitir al superusuario (i.e., cualquier proceso cuyo EUID sea 0) realizar cualquier operación válida[2] en el sistema. Operaciones que sólo el superusuario puede realizar son:

- Cambiar el directorio raíz de un proceso con chroot.
- Crear ficheros especiales de dispositivo
- Cambiar el reloj del sistema.
- Aumentar la prioridad de los procesos y sus límites de recursos.
- Cambiar el nombre del sistema.
- Configurar las interfaces de red.
- Etc.

Ejecución mediante setuid *y* setgid

El control de acceso tradicional se complementa con un sistema de sustitución de identidad que controlan el núcleo y el sistema de ficheros en colaboración. La idea básica es que este sistema permite que determinados programas se ejecuten con permisos elevados, incluso de *root*. Este mecanismo permite a usuarios no privilegiados realizar tareas concretas que requieren de permisos más elevados.

Al ejecutar un fichero con permisos *setuid* o *setgid*, el proceso cambia su EUID o el EGID al del propietario del fichero. Así, por ejemplo, para cambiar sus contraseñas, los usuario utilizan el comando passwd, que es propiedad de *root* y dispone del permiso *setuid*.

Control de acceso moderno

El sistema de acceso tradicional ha soportado el paso del tiempo debido a que es excelente en la mayoría de los casos. No obstante, tiene ciertas limitaciones:

- La cuenta *root* es un punto de fallo de seguridad potencial; una vez comprometida, la integridad de todo el sistema se tambalea.
- El único modo de repartir los privilegios del *root* es mediante el permiso *setuid*. Sin embargo, es difícil escribir programas seguros, y a mayores el sistema debería incorporar ciertas funcionalidades sin necesidad de software dedicado.

2 La operación debe ser 'válida': por ejemplo, no es posible ejecutar un fichero que no tiene permisos de ejecución.

- Este modelo no es seguro en un entorno en red.
- Algunas políticas de seguridad no pueden ser implementadas en el modelo de Unix (e.g., impedir la publicación de documentos restringidos).
- Al estar incluido en el núcleo, no es fácil redefinir el comportamiento del sistema sin reescribir el código y recompilar.
- No hay soporte para auditoría: no es fácil saber qué se permite hacer a un usuario o grupo, etc.

Por todas estas razones nuevos aspectos de control de acceso se han incorporado a los sistemas Unix con el tiempo. Comentamos algunos de ellos a continuación:

SELinux

SELinux es un proyecto de la Agencia Nacional de Seguridad de los E.E.U.U., cuyo principal objetivo es proporcional «control de acceso obligatorio» (MAC, por sus siglas en inglés, *Mandatory Access Control*). Bajo un sistema MAC, todos los permisos son establecidos por el administrador, y un usuario no puede delegar o cambiar los permisos sobre los objetos que posee.

Pluggable Authentication Modules (PAM)

PAM es una tecnología de autentificación, y no de control de acceso, y constituye un componente de control de acceso importante en la mayoría de los sistemas.

Tradicionalmente las contraseñas se cotejaban con el contenido del fichero /etc/shadow, de modo que se seleccionara el UID apropiado. Sin embargo, con la presencia de redes, criptografía, identificación biométrica, etc., es necesaria más flexibilidad. PAM es un *wrapper* para bibliotecas de autentificación variadas. Un administrador selecciona los métodos deseados y los contextos de uso. Así, un programa que desea autentificar a un usuario simplemente llama al sistema PAM.

Kerberos

Como PAM, Kerberos es un sistema de autentificación. Pero a diferencia de aquél, este es un método específico. Se usan frecuentemente en conjunto: PAM como interfaz y Kerberos como implementación real.

Kerberos utiliza un servidor de terceros confiable para autentificar en una red completa: en lugar de autentificarse en la máquina usada, un usuario proporciona las credenciales a Kerberos, el cual expede la información criptográfica necesaria para autentificarse en otros servicios.

Listas de control de acceso (ACL)

Las listas de control de acceso suponen una generalización del modelo de permisos de usuario/grupo/otros. Es parte del sistema de ficheros, y permite adaptar la noción de permisos a varios usuarios y grupos al mismo tiempo.

El mundo real

A pesar de todas estas posibilidades, la realidad es que en el día a día es necesario encontrar un equilibrio entre seguridad y simplicidad. Así, es frecuente que en la mayoría de los sistemas se siga utilizando el modelo tradicional apoyado por herramientas como sudo ('**s**uper **u**ser **do**' o '**s**witch **u**ser and **do**'). Este programa permite a los usuarios ejecutar programas con privilegios diferentes de manera segura, convirtiéndose temporalmente en otro usuario (cambiando su EUID – habitualmente al de la cuenta *root* –). El software consulta el fichero /etc/sudoers, que contiene la lista de usuarios que pueden utilizarlo, y las órdenes que pueden ejecutar con él. Si la combinación es correcta, sudo solicita la identificación del usuario y ejecuta la orden.

Otros pseudo-usuarios

Generalmente *root* es el único usuario que tiene un estatus especial desde el punto de vista del kernel, pero otros pseudo usuarios se definen para el sistema. Se característica principal es un UID bajo (generalmente inferior a 100).

La costumbre es sustituir la contraseña en /etc/shadow por un asterico, de modo que no es posible hacer login en estas cuentas, y establecer su shell a /bin/false o /bin/nologin, para evitar exploits remotos.

Un ejemplo característico es la cuenta *nobody*, que sirve para representar *roots* de otros sistemas en el NFS, de modo que se les retiran sus privilegios de *root*. Usualmente su UID es -1 o -2, aunque realmente es arbitrario (e.g., Red Hat usa el UID 99).

Otros ejemplos son cuentas como *bin* o *daemon*, a las que se asignan a veces ficheros y procesos del sistema que no es necesario que *root* posea.