Linux: Sistemas de ficheros y permisos

Alberto Molina Coballes

IES Gonzalo Nazareno
CONSEJERÍA DE EDUCACIÓN

19 de enero de 2018

Recordamos los permisos tradicionales UNIX sobre ficheros:

- Esquema ugoa
- Permisos especiales (SUID, SGID, sticky bit)
- Es una implementación de un sistema de DAC (*Discretionary Access Control*) ya que los usuarios pueden modificar los permisos (chmod)

En el inicio fue ugoa

- Tradicionalmente en UNIX se definen los permisos de ficheros para usuario (u), grupo (g), otros (o) o todos (a)
- Los tres permisos básicos son lectura (r), escritura (w) y ejecución (x)
- Para borrar un fichero necesitamos permiso de escritura y ejecución en el directorio padre
- Un usuario tiene un grupo principal y puede pertenecer a otros grupos
- Los permisos iniciales de un fichero se definen mediante la orden `umask`
- Se puede cambiar el grupo principal en una sesión con `newgrp`
- Notación octal: rwx = 111, rw- = 110, etc.

Permisos especiales de ficheros

- Al establecer *set user identification* (`suid`) sobre un fichero ejecutable, este lo puede ejecutar otro usuario con los permisos del propietario.
- Si el propietario es superusuario puede ser arriesgado
- Ficheros con bit de suid activado:

  ```
  find / -perm /4000
  ```

- Al establecer *set group identification* (`sgid`) sobre un fichero ejecutable, ocurre lo mismo que con suid, pero ahora se aplican los permisos del grupo propietario
- Ficheros con bit de sgid activado:

  ```
  find / -perm /2000
  ```

Permisos especiales de ficheros

- sgid sobre un directorio: Todos los ficheros que se creen heredan el grupo propietario del directorio
- suid y sgid se indican con (s)
- Al activar el *sticky bit* en un directorio, sólo el propietario del fichero podrá borrarlo
- Utilizado en directorios donde varios usuarios pueden escribir (por ejemplo /tmp)
- Muy interesante combinado con sgid
- sticky bit se indica con (t)

Linux kernel POSIX capabilities

Introducción

- Tradicionalmente dos privilegios:
  - Procesos privilegiados: Se saltan las comprobaciones de permisos
  - Procesos no privilegiados: Comprobación estricta de permisos
- *Kernel Capabilities*: Mecanismo de seguridad basado en el principio de mínimo privilegio, agrupando ciertos privilegios en una "capacidad"
- Se le puede asignar a un proceso una capacidad específica a nivel del kérnel (*kernel capability*)
- No son originales de linux: POSIX capabilities, detalladas en el borrador (retirado) 1003.1e:
  `http://wt.tuxomania.net/publications/posix.1e/download.html`

Lista de capacidades

`man 7 capabilities`

Algunos ejemplos:

- `CAP_CHOWN`
- `CAP_KILL`
- `CAP_NET_ADMIN`
- `CAP_NET_BIND_SERVICE`
- `CAP_NET_RAW`
- `CAP_SYS_ADMIN`
- `CAP_SYS_MODULE`
- `CAP_SYS_RAWIO`
- `CAP_SYS_TIME`

Conjuntos de capacidades de un ejecutable

- Se pueden definir los siguientes conjuntos de capacidades a un ejecutable

  Permitidas(p) Automáticamente permitidas, independientemente de las capacidades heredadas del proceso padre

  Heredables(i) Se añaden al proceso junto con las del proceso padre para determinar las capacidades permitidas

  Efectivas(e) Usadas para permitir capacidades de linux en aplicaciones que no las soportan directamente

`Andy Pearce: File Capabilities In Linux`

Definiendo capacidades

- Se instala el paquete `libcap2-bin`
- `setcap`: Define las capacidades de un fichero
- `getcap`: Obtiene las capacidades de un fichero
- `getpcaps`: Lista las capacidades de un proceso

Atributos de ficheros

Atributos de ficheros

- Asociados inicialmente a ext?, parcialmente soportados por otros sistemas de ficheros (btrfs, xfs, etc.)
  - a: append only
  - c: compressed
  - d: no dump
  - e: extent format
  - i: immutable
  - j: data journalling
  - s: secure deletion
  - t: no tail-merging
  - u: undeletable
  - A: no atime updates
  - C: no copy on write
  - D: synchronous directory updates
  - S: synchronous updates
  - T: top of directory hierarchy

Usar atributos de ficheros

- Se establecen o modifican los atributos de ficheros con `chattr`, por ejemplo:

```
chattr +u fichero
```

- Se comprueban los atributos que están definidos con `lsattr`
- Ambos programas se incluyen en el paquete `e2fsprogs`
- Algunos atributos hay que establecerlos como root o con la capacidad de núcleo correspondiente

Atributos extendidos de ficheros

Atributos extendidos de ficheros

- Pares clave-valor asociados a un fichero, ordenadas en cuatro clases:
  - security
  - system
  - trusted
  - user
- Las capacidades de núcleo se guardan como atributos extendidos de seguridad
- Es necesario que el sistema de ficheros incluya soporte para xattrs
- Los programas para utilizar xattrs se incluyen en el paquete `attr`
- La clase "user" permite almacenar cualquier información asociada al fichero:

```
setfattr -n user.checksum -v "3baf9ebce4c664ca8d9e5f6aaafb47fb" fichero
```

Listas de control de acceso (ACL)

acl sobre los ficheros

- Permite un control adicional sobre los permisos de los ficheros
- Debe estar habilitado en el sistema de ficheros
- Se definen con `setfacl` del paquete `acl`
- Ejemplo:

  ```
  setfacl -m ‘‘u:usuario:rw’’ fichero
  ```

- Se comprueban las ACLs con `getfacl`
- Cuando hay ACLs definidas se indica con un + al listar:

  ```
  drwxrwxr-x+  2 alberto alberto        6 ene 19 12:15 Directorio
  ```

- Cuidado con el uso de ACLs o esto va a parecerse a Windows ;)