

# *kernel capabilities*



IES Gonzalo Nazareno  
**CONSEJERÍA DE EDUCACIÓN**

Alberto Molina Coballes



17 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 kernel (*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