Sistemas Operativos

Seguridad en Sistemas Operativos UNIX

Curso 2024

Facultad de Ingeniería, UDELAR

Agenda

- Seguridad en sistemas UNIX
  - Arquitectura seguridad UNIX
  - Principals, Sujetos y Objetos
  - Principals
  - Sujetos
  - Objetos
  - Algoritmo de control de acceso
- SUID
- Sandbox
  - Sandbox

Seguridad en sistemas UNIX

Modelo de seguridad en UNIX

- Los controles de seguridad en UNIX no están contemplados en los objetivos de diseño iniciales (1970)
- Nuevos controles de seguridad se han incorporado, otros se fortalecieron a demanda
- Siempre buscando **interferir lo menos posible con las estructuras existentes**
- La seguridad es gestionada por administradores con mucha experiencia

Principals

- En Unix son identificados por la pareja user identity (UID) y group identity (GID)
- Ambos enteros de 16 bits
- Algunos UIDs tienen significados especiales
- El super usuario (*root*) en unix es siempre el **uid** 0

Grupos Unix

- Cada usuario está presente en al menos un grupo de usuarios (grupo primario)
- Agregar usuarios a grupos es una base conveniente para **decisiones de control de acceso**
- Ejecutando el comando *groups* se pueden saber los grupos a los que pertenece un usuario

Cuentas de usuarios

Se almacena en:

- en el archivo */etc/passwd*,
- */etc/shadow* y
- el directorio home (*Home Dir*) del usuario

Cuentas de usuarios

Se almacena en:

- en el archivo */etc/passwd*,
- */etc/shadow* y
- el directorio home (*Home Dir*) del usuario

Además debemos tener en cuenta:

- Cada individuo debe tener su propia cuenta
- Evitar el uso de cuentas grupales
- Todas las cuentas deberían autenticarse de la misma forma
- La cuenta root es (en general) el superusuario
- Existen otras cuentas por defecto, pero **siempre** deberían estar **bloqueadas**

/etc/passwd

Contiene una tabla con los siguientes campos:

login:contraseña encriptada:**UID**:**GID**:Comentario:HomeDir:Shell

- El usuario y grupo dueño es root
- Por defecto tiene permiso **rw** para el **dueño** y **r** para el **resto**

/etc/passwd

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:65534:65534:Kernel Overflow User:/:/sbin/nologin
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
```

/etc/shadow

- Si está en el sistema, la contraseña se cambia por un 'x' en /etc/passwd
- El /usuario y grupo **dueño** es **root**
- Permisos: r para el dueño, nada para el resto (grupo y resto del mundo)
- Es una tabla con los siguientes campos ....

/etc/shadow

```
root:!::0:99999:7:::
bin:*:18831:0:99999:7:::
daemon:*:18831:0:99999:7:::
adm:*:18831:0:99999:7:::
lp:*:18831:0:99999:7:::
sync:*:18831:0:99999:7:::
shutdown:*:18831:0:99999:7:::
halt:*:18831:0:99999:7:::
mail:*:18831:0:99999:7:::
operator:*:18831:0:99999:7:::
games:*:18831:0:99999:7:::
nobody:*:18831:0:99999:7:::
apache:!!:18926::::::
glest:\$1\$A3TUG3YS\$DT4le2QA7FEiric28Nh94.:13305:0:99999:7:::
pedro.ruiz:\$y\$j9T\$8dCv0hFgRsOUe8rjcr5wP0\$iN4fBzMrAAtdqDnBhHnWPJYsKVHDgK3LQ1kmwooSHk2:19020:0:99999:7:::
```

Sujetos

- Los sujetos son los procesos del sistema
- Se identifican con un *Process ID* (PID)
- Se crean nuevos mediante *fork* o *exec*
- Cada proceso tiene asociado un UID/GID **real**, y otro **efectivo**
- El **UID real** (ruid) es heredado del padre
- El **UID efectivo** (euid) es heredado del padre o del archivo que se está ejecutando

Proceso de login

- Los usuarios se identifican mediante nombres de usuario y passwords
- Cuando un usuario se loguea, el proceso *login* **verifica** el **usuario** y **password**
- Si la verificación es exitosa, se cambia el UID/GID al del usuario y se ejecuta la *shell de login*
- Los passwords se cambian mediante el comando *passwd*

Objetos

- En UNIX, los objetos para el control de acceso incluyen: archivos, directorios, dispositivos, I/O, etc.
- Están organizados en un sistema de archivos con estructura de árbol

todo objeto es un archivo

Inodos

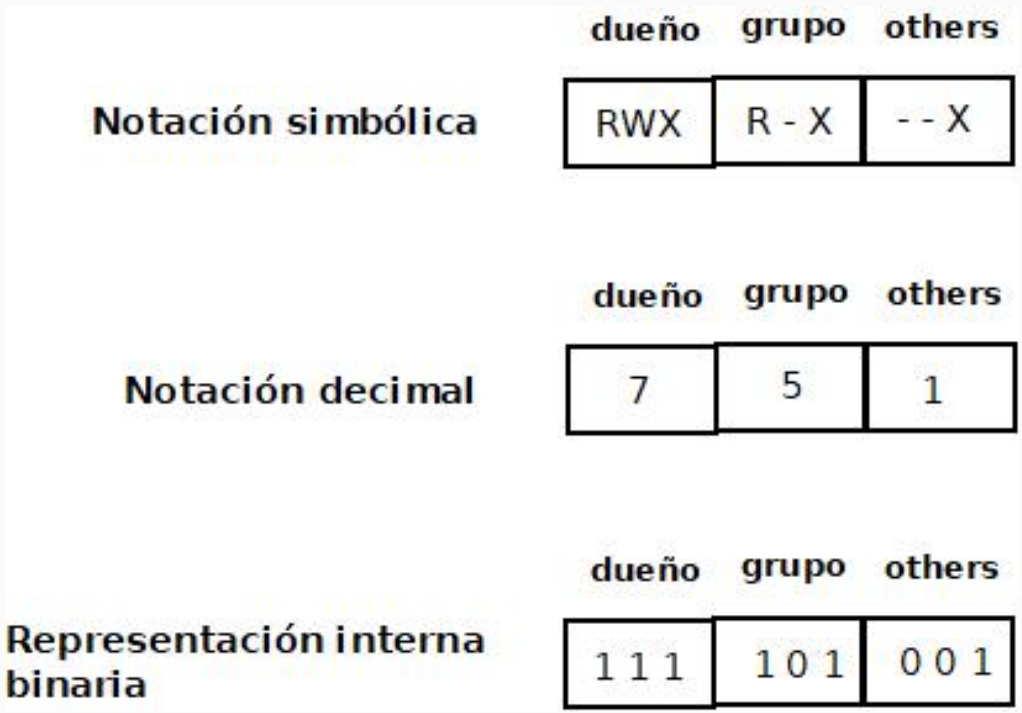
- Cada entrada de archivo en un directorio es un puntero a una estructura de datos llamada *inodo*
- Cada directorio tiene un puntero a sí mismo, el archivo '.', y un puntero a su padre '..'
- Cada archivo tiene un **usuario dueño**, usualmente el que lo creó, y un **grupo dueño** al que pertenece

Permisos de archivos

- Los permisos de los archivos (bits de permisos), se agrupan en tres tripletas;
- para el usuario (**u**) y grupo (**g**) dueño y el resto del mundo o others (**o**).
- En cada tripleta se definen permisos de *read* (**r**), *write* (**w**), y *execute* (**x**).
- Se pueden representar como números **decimales**, separando los nueve permisos (bits) en 3 grupos de 3 (ver ejemplos) o con notación **simbólica**
- Cada derecho de acceso está representado por un bit, el cual si está "prendido" permite el acceso

Permisos de archivos: ejemplos

Permisos UNIX



Permisos por defecto

- Las archivos son creados por defecto con permisos 666 y los directorios 777
- Deben ajustarse con la *umask*
- Los permisos se calculan a partir del AND binario de los valores por defecto y el inverso de la máscara
  (NOT máscara) $\equiv$ (XOR máscara)
- Es modificado con el comando *umask*

Ejemplos de UMASK

- umask **022**: provoca que solo el propietario pueda modificar los archivos
- umask **027**: deja sin permisos de escritura al grupo y ningún permiso al resto del mundo

Permisos para directorios

- Cada usuario tiene un directorio home (*Home Dir*)
- Se crean con el comando *mkdir*
- Para agregar archivos en un **directorio**, se deben tener los permisos adecuados
  - **lectura** permite encontrar (listar) archivos en el directorio
  - **escritura** permite agregar y borrar archivos al directorio
  - **ejecución** para poder hacer ese directorio el actual y abrir archivos

Control de acceso

- Está basado en atributos de los sujetos (procesos) y de los objetos (recursos)
- Las sistemas Unix clásicos (*posix unix*) asocian tres conjuntos de derechos de acceso a cada recurso, correspondientes al owner (**u**), group (**g**) y world o el resto del mundo (**o**)
- El super usuario no está sujeto a estos chequeos

Algoritmo de control de acceso

- Si el **uid efectivo (euid)** del proceso indica que es dueño del archivo, los bits de permisos de owner deciden si se tiene acceso
- Si no se es dueño del archivo, pero el **gid efectivo (egid)** indica que su grupo es el dueño, se aplican los permisos del grupo
- Si no, se aplican los permisos de resto del mundo (Others)

Ejemplos control acceso UNIX

Ejemplo 1

- El proceso con UID/GID efectivos *jose.rodriguez/sistoper*
- desea permisos de *lectura (r)* sobre
- el archivo **ejemplo.txt** del cual
  - *pedro.martinez, sistoper* es el UID, GID dueño y
  - *rw-,-wx,rwx (637)* son los permisos definidos

Ejemplos control acceso UNIX

Ejemplo 1

- El proceso con UID/GID efectivos *jose.rodriguez/sistoper*
- desea permisos de *lectura (r)* sobre
- el archivo **ejemplo.txt** del cual
  - *pedro.martinez, sistoper* es el UID, GID dueño y
  - *rw-,-wx,rwx (637)* son los permisos definidos

Resultado

Ejemplos control acceso UNIX

Ejemplo 1

- El proceso con UID/GID efectivos *jose.rodriguez/sistoper*
- desea permisos de *lectura (r)* sobre
- el archivo **ejemplo.txt** del cual
  - *pedro.martinez, sistoper* es el UID, GID dueño y
  - *rw-,-wx,rwx (637)* son los permisos definidos

Resultado

- Se chequean los permisos del grupo dueño

Ejemplos control acceso UNIX

Ejemplo 1

- El proceso con UID/GID efectivos *jose.rodriguez/sistoper*
- desea permisos de *lectura (r)* sobre
- el archivo **ejemplo.txt** del cual
  - *pedro.martinez, sistoper* es el UID, GID dueño y
  - *rw-,-wx,rwx (637)* son los permisos definidos

Resultado

- Se chequean los permisos del grupo dueño
- El acceso solicitado será **denegado**

Ejemplos control acceso UNIX

Ejemplo 2

- El proceso con UID/GID efectivos *jose.rodriguez/arqsis*
- desea permisos de *lectura (r)* sobre
- el archivo **ejemplo.txt** del cual
  - *pedro.martinez, sistoper* es el UID, GID dueño y
  - *rw-,-wx,rwx (637)* son los permisos definidos

Ejemplos control acceso UNIX

Ejemplo 2

- El proceso con UID/GID efectivos *jose.rodriguez/arqsis*
- desea permisos de *lectura (r)* sobre
- el archivo **ejemplo.txt** del cual
  - *pedro.martinez, sistoper* es el UID, GID dueño y
  - *rw-,-wx,rwx (637)* son los permisos definidos

Resultado

Ejemplos control acceso UNIX

Ejemplo 2

- El proceso con UID/GID efectivos *jose.rodriguez/arqsis*
- desea permisos de *lectura (r)* sobre
- el archivo **ejemplo.txt** del cual
  - *pedro.martinez, sistoper* es el UID, GID dueño y
  - *rw-,-wx,rwx (637)* son los permisos definidos

Resultado

- Se chequean los permisos de others

Ejemplos control acceso UNIX

Ejemplo 2

- El proceso con UID/GID efectivos *jose.rodriguez/arqsis*
- desea permisos de *lectura (r)* sobre
- el archivo **ejemplo.txt** del cual
  - *pedro.martinez, sistoper* es el UID, GID dueño y
  - *rw-,-wx,rwx (637)* son los permisos definidos

Resultado

- Se chequean los permisos de others
- El acceso solicitado será **otorgado**

Control de acceso en UNIX: limitaciones

- Los objetos en Unix solo tienen un usuario y un grupo dueño
- Los permisos solo controlan el acceso de lectura, escritura y ejecución
- Otras operaciones de control de acceso deben implementarse a nivel de aplicaciones
- Se hace impracticable implementar políticas de seguridad más complejas

Control de acceso en UNIX: limitaciones

- Estas limitaciones han llevado a que se hayan incorporado distintas "extensiones" o mejoras al modelo de control de acceso de Unix tradicional
- Algunos ejemplos:
  - Access Control Lists (**ACLs**) a nivel de archivos
  - Variantes del kernel de linux "seguras": **SeLinux**, **Trusted Solaris**, etc
  - Mejoras en capas intermedias como p.e. **Sudo**, **TCP wrappers**

Outline

Set User ID (SUID)

- En ciertas circunstancias necesitamos poder ejecutar procesos con privilegios de root
  - abrir un socket en puertos ¡ 1024 (privilegiado)
  - o modificar la contraseña de un usuario
- En sistemas UNIX, la solución adoptada es el uso de Set UserID (**SUID**) o Set GroupID (**SGID**)
- Programas con el **SUID** o **SGID** ejecutan con el UID/GID efectivo del usuario o grupo dueño del archivo

SUID root

- En archivos con **SUID root**, el usuario/proceso que lo ejecuta obtiene privilegios de root durante la ejecución
- Algunos ejemplos:
  - */bin/passwd*: cambio de contraseña
  - */bin/login*: programa de login
  - */bin/su*: Cambiar el UID (Switch Userid)
- Debemos cuidar que estos programas hagan **solo** lo que deben hacer

Riesgos de SUID root

- Si logramos *"engañar"* a un programa con SUID de root, estamos logrando acceso de root
- Estos programas deben procesar con mucho cuidado los parámetros de entrada :-)
- Usemos SUID/SGID sólo cuando es estrictamente necesario
- Debemos controlar en especial la integridad de estos programas (Tripwire, AIDE, Yafic)

Outline

Invocaciones controladas

- Combinando ownership, permisos y programas SUID (ver caso de estudio)
- usando técnicas complementarias:
  - **Jail root**,
  - Containers o **sandboxing**,
  - **wrappers**

Sandbox con chroot

- Implementamos restricciones de control de acceso, restringiendo los procesos a un ambiente de sandbox
- No se permite el acceso a objetos fuera del sandbox
- El comando chroot cambia la raiz del file System
- Solo accedemos a los objetos bajo la nueva raíz *¡directory¿*
- Algunos servicios que usan este mecánismo: *bind*, *postfix*

Otros mecanismos de sandbox

- Virtualización a nivel de Sistema Operativo
  Solaris containers, Linux-VServers, OpenVZ, etc
- para-virtualización o full virtualization
  VMware, QEMU, KVM, Xen, VirtualBox

Invocaciones controladas

- Combinando ownership, permisos y programas SUID
  Ver caso de estudio
- también con técnicas complementarias:
  - **Jail root**,
  - Containers o **sandboxing**,
  - **wrappers**

Sandox

chroot

- Implementamos restricciones de control de acceso, restringiendo los procesos a un ambiente controlado (de sandbox)
- No se permite el acceso a objetos fuera del sandbox
- El comando chroot cambia la raiz del file System
  chroot ¡directory¿ ¡command¿
- Solo accedemos a los objetos bajo la nueva raíz **¡directory¿**
- Los procesos (además) podrían ejecutar con un eUID de la aplicación
- Algunos servicios que usan este mecánismo: *bind*, *postfix*

Sandbox

Virtualización

- Virtualización a nivel de Sistema Operativo
  Solaris containers, Linux-VServers, OpenVZ, etc
- para-virtualización o full virtualization
  VMware, QEMU, KVM, Xen, VirtualBox

Bibliografía

- Dieter Gollmann, Computer Security, 3rd edition, Editorial Wiley, 2011. Cap. 3, 6 y 7.
- A.Silberschatz, P.Galvin and G. Gagne, Operating Systems Concepts, 9th edition, Editorial Wiley, 2014. Cap 14 y 15.
- M. Souppay, K. Scarfone, Guide to Malware Incident Prevention and Handling for Desktop and Laptops, NIST Special Publication SP 800-53. Cap. 2