

---

# Seguridad del sistema

Diseño y Administración de Sistemas y Redes

*Juan Céspedes <cespedes@gsync.escet.urjc.es>*



*Curso 2005–2006*

---



## Seguridad en un sistema

- Existen muchos estudios acerca del papel que juega la seguridad en un sistema y cómo estudiarla.
- El modelo habitual de seguridad se divide en 7 capas:
  - Identificación y autenticación
  - Control de acceso y autorización
  - Integridad del sistema
  - Disponibilidad
  - Detección de intrusos
  - Seguridad física
  - Datos

## Seguridad en UNIX

- La seguridad no tuvo un papel destacado en el diseño de UNIX.
- A pesar de ello, siempre se pueden tomar medidas que hagan el sistema más seguro.
- La seguridad es inversamente proporcional a la comodidad de uso.
- Querremos proteger:
  - Información sensible
  - Recursos (CPU, disco, comunicaciones)
  - Integridad del sistema

## Sentido común

- Se deben conocer y controlar todos los puntos de acceso al sistema: acceso físico (consola), **telnet**, **ssh**, etc.
- Debemos procurar impedir el acceso al sistema por personas no autorizadas (eligiendo contraseñas adecuadas. . .)
- Existen herramientas de seguridad que proporcionan informes de posibles vulnerabilidades.
- Monitorizar ficheros de *logs* del sistema.
- En caso de duda, recurrir a expertos en seguridad.
- Si todo falla, tener copias de seguridad actualizadas es fundamental.

## /etc/passwd

- Es el método más directo para conseguir acceso a un sistema.
- Los ficheros `/etc/passwd` y `/etc/shadow`:
  - Deben poder modificarse únicamente por el usuario `root`.
  - Deben tener contraseñas seguras.
  - No deben contener usuarios que ya no se usen.
  - Las cuentas han de ser personales e intransferibles.
  - El campo "`shell`" debe ser una de las permitidas.
- Además, conviene realizar de vez en cuando tareas de comprobación rutinarias, como ver si hay usuarios dos usuarios con el mismo UID (especialmente con UID=0), etc.

## Configuración de cuentas de usuario

- La configuración inicial de las cuentas que se abren en el sistema está en el fichero `/etc/adduser.conf` (Debian) y en el `/etc/default/adduser` (Red Hat).
- En la configuración de las cuentas de usuarios, hay que elegir:
  - Política de cambio y expiración de contraseñas.
  - Grupos a que pertenece cada usuario.
  - `shell` por defecto.
  - Directorios de inicio.
  - Permisos de los directorios de inicio.
  - `umask`: permisos con que se crean los ficheros por defecto.

## Permisos de ficheros

- Regulan el acceso por parte de los usuarios a cada fichero.
- Cada fichero tiene:
  - Permisos para el propietario
  - Permisos para el grupo
  - Permisos para el resto
- Los tipos de permisos son:
  - Lectura (**r**)
  - Escritura (**w**)
  - Ejecución (**x**)

```
-rw-r----- 1 root shadow 880 2005-03-15 09:55 /etc/shadow
```

## Permisos en directorios

- En los directorios, el significado de los tipos de permisos es ligeramente distinto al del resto de ficheros:
  - **r**: Se puede ver el nombre de los ficheros que contiene el directorio.
  - **w**: Se pueden crear, borrar o renombrar ficheros.
  - **x**: Se puede acceder al contenido del directorio, y a subdirectorios de éste.

## Permisos especiales

- Existen otros 3 permisos que afectan únicamente a ejecutables y a directorios:
  - *set-uid* (**u+s**): cambia el UID efectivo del proceso que ejecuta esta orden al del dueño del fichero.
  - *set-gid* (**g+s**): cambia el GID efectivo del proceso que ejecuta esta orden al del grupo al que pertenece el fichero. En directorios, hace que los ficheros que se creen dentro pertenezcan al mismo grupo que el directorio, sin importar a qué grupo pertenece el usuario que lo crea.
  - *sticky bit* (**o+t**): En directorios, evita que los usuarios puedan borrar o renombrar un fichero a no ser que sean el dueño de ese fichero. Muy usado en directorios temporales, compartidos por mucha gente.

## Bits *set-uid* y *set-gid*

- Los permisos de *set-uid* y *set-gid* son muy útiles, pero pueden llegar a ser muy peligrosos, ya que un proceso cambia de identidad y se convierte temporalmente en otro, normalmente con muchos más privilegios.
- Para evitar problemas, es imprescindible que el número de ficheros con los bits *set-uid* y *set-gid* del sistema sea el mínimo imprescindible.
- En los ejecutables que no haya más remedio que usarlos, hay que cuidar especialmente bien la seguridad, para evitar que tengan demasiadas vulnerabilidades.

## Bits *set-uid* y *set-gid* (2)

- Precauciones a tomar en *set-uids* y *set-gids*:
  - Evitar *shell scripts* que utilicen estos bits.
  - Si el ejecutable solo deben usarlo un grupo de personas, usar permisos para forzarlo.
  - No invocar *shells* ni ninguna orden que pueda darle el control a un usuario (ejemplo: editores).
  - No confiar en el valor del directorio actual, ni de ninguna variable de entorno, ni de la corrección de los argumentos que se pasen al programa.
  - En concreto, no confiar en el **PATH** (por ejemplo, si hay que ejecutar un programa externo, usar **exec1()** en lugar de **exec1p()**).

## Tipos de ataques a un sistema

- Captura de información sensible (*sniffers*...).
- Búsqueda exhaustiva de usuarios y contraseñas.
- Vulnerabilidades en el sistema.
- Ingeniería social.
- SPAM.
- Virus.

## Captura de información sensible

- Permisos de ficheros.
- *Passwords* guardadas en sitios poco seguros.
- En tiempo real:
  - Red ethernet (**dsniff**, **sniffit...**).
  - Red inalámbrica (**airsnort**, **kismet...**).
  - Teclado (*key loggers*).
  - Sesión X (**xkey**).
  - Pseudo-terminales (**ttysnoop**).

## Búsqueda exhaustiva de usuarios y contraseñas

- Diferentes medios dependiendo del acceso que tengamos al sistema:
  - Prueba remota (**telnet**, **pop...**).
  - Prueba local (**login**, **su**).
  - Búsqueda *off-line* de contraseñas que coincidan con el campo *passwd* cifrado (**john...**).

## Vulnerabilidades en el sistema

- Descuidos de usuarios (malas contraseñas, permisos, etc).
- Descuidos del administrador (permisos de ficheros...).
- Funcionalidades no conocidas de programas.
- Fallos de programación.

Las vulnerabilidades pueden ser locales y remotas, y según su severidad, pueden permitir:

- Denegación de servicio.
- Acceso a datos.
- Corrupción de datos.
- Acceso a otra cuenta (usuario normal o superusuario).

## Vulnerabilidades en el sistema (2)

- Descuidos de usuarios
  - Las contraseñas han de ser seguras.
  - Las contraseñas no deben apuntarse ni darlas a terceras personas.
  - No se debe prestar la cuenta a nadie.
  - Cuidado con el PATH: "." no es una buena idea.
  - No ejecutar nunca código ajeno del que no se esté seguro de su funcionalidad e integridad (virus, troyanos, etc).
  - El *umask* y los permisos de los ficheros propios ha de ser adecuado.
  - Cuidado con el uso de `/tmp` a la hora de programar!

## Vulnerabilidades en el sistema (3)

- Descuidos del administrador
  - Instalación de programas en el sistema solo si la fuente es conocida y fiable.
  - Ningún fichero escribible por todos los usuarios, salvo los directorios `/tmp` y `/var/tmp`.
  - Ningún dispositivo fuera de `/dev`, y aún allí, cuidado con sus permisos.
  - Mucho cuidado con los ficheros `set-uid` y `set-gid`: solo los imprescindibles y asegurándonos de su corrección.
  - Ejecución de órdenes como `root` solo si es imprescindible y en un entorno controlado.
  - Servicios en el sistema: solo los imprescindibles.

## Fallos de programación

- Fallos de diseño.
- Fallos en la declaración de rangos (errores “*off-by-one*”).
- Condiciones de carrera (mal uso de ficheros temporales. . .).
- Olvidos a la hora de comprobar errores en la ejecución de funciones o procedimientos que *no suelen* fallar.
- *Buffer overflows*: desbordamiento de la pila interna de un programa al introducirle datos para los que no está preparado, *obligándole* a ejecutar código malicioso.

## Buffer overflows

- Probablemente, son la causa más habitual de *exploits* debidos a fallos de programación.
- Consisten en utilizar una variable local de un tamaño determinado para almacenar un valor que en circunstancias normales siempre es menor que el tamaño reservado.
- Si en realidad se introduce un valor con un tamaño mayor, y si no se comprueba el tamaño, se podrían sobrescribir direcciones de memoria que pertenecen a otras variables. . .
- Las variables locales de una función se almacenan en la pila interna del programa, junto con la dirección de retorno de esa misma función, que es donde continuará el flujo de ejecución del programa al terminar ésta.

## Ejemplo de *buffer overflow*

El problema más habitual es declarar un *array* de caracteres de un tamaño fijo, y copiar a ese *array* un valor externo.

Ejemplo (en C):

```
#include <stdio.h>
#include <stdlib.h>

int
main(int argc, char *argv[]) {
    char home[128];

    strcpy(home, getenv("HOME"));
    printf("El valor de HOME es %s\n", home);
    return 0;
}
```

## Denegación de Servicio (DOS)

- Más conocido como “DOS” (*Denial Of Service*).
- Consiste en provocar que un determinado servicio no se preste con normalidad.
- Se puede provocar utilizando alguna vulnerabilidad o simplemente forzándolo, realizando muchas peticiones por segundo.
- Se puede evitar detectando el “ataque” y bloqueando las conexiones desde la máquina atacante.
- Versión mejorada: “DDOS” (*Distributed Denial Of Service*): ataques simultáneos desde multitud de máquinas de la red. Para muchos DDOS no se conoce aún una solución aceptable.

## Ataques desde dentro

- Acceso físico al equipo.
- Abuso de autoridad o de privilegios.
- Uso de información clasificada.
- Denegación de servicio.

Estos ataques se pueden minimizar con una buena política de uso de cuentas, contraseñas, grupos y permisos, y con herramientas como:

- Cuotas de disco: `quotacheck`, `repquota`, `edquota`, `quotaon`, `quotaff`, `quota`.
- Limitación de recursos (CPU, memoria, ficheros, etc): `ulimit`.

## Otros tipos de ataques

- Ingeniería social:
  - Contacto directo con algún usuario de la máquina para averiguar información sobre esta: contraseñas, direcciones, etc.
- “Ataques” por correo electrónico:
  - Virus: se aprovechan de vulnerabilidades de sistemas de correo electrónico.
  - SPAM: Envío de publicidad masiva no solicitada.
  - HOAX: Engañan al usuario, normalmente con la intención de hacerle perder datos.
  - SCAM: Timo, estafa (como los de Citybank, eBay, Caja Madrid. . .).

## Ataques remotos

Los más habituales son:

- Denegación de servicio (DoS).
- Aprovechando vulnerabilidades (especialmente *buffer overflows*) de determinados servicios.
- Búsqueda exhaustiva de contraseñas.

Para limitarlos, conviene tener al día actualizaciones del sistema operativo, vigilar qué servicios ofrecemos (`netstat -ntu`) y eliminar los que no sean necesarios.

## Cifrado de información

Es necesario proteger nuestros datos en cuatro aspectos:

- Autenticación
- Integridad
- Cifrado
- No-repudio

Para ello, podemos usar los programas:

- crypt (solo cifrado)
- PGP (Pretty Good Privacy)
- GPG (GNU Privacy Guard)

Para conseguirlo, hay que usar cifrado mediante *clave pública*.

## Los logs

Si sospechamos que nuestro sistema se ha visto comprometido, es importante mirar los *logs* para ver si se observa algún comportamiento anómalo.

Los *logs* están siempre a partir del directorio `/var/log`, y están compuestos por distintos ficheros, dependiendo del sistema.

El encargado de escribir los *logs* es el demonio `syslogd`, cuya configuración está en `/etc/syslog.conf`.

Los mensajes a guardar se clasifican en:

- *facilities* (AUTH, AUTHPRIV, CRON, DAEMON, FTP, KERN, LPR, MAIL, NEWS...)
- *priorities* (EMERG, ALERT, CRIT, ERR, WARNING, NOTICE, INFO y DEBUG).

## Detección de intrusos

Además de los *logs*, si sospechamos que un sistema está comprometido, es importante:

- Revisar */etc/passwd*, */etc/group*, etc.
- Comprobar cualquier hecho inusual (uso de CPU, de memoria, de disco, de la red. . .).
- Revisar los procesos en ejecución.
- Si es posible, comprobar la integridad de todos los ficheros del sistema.
- Si todo lo demás falla, reinstalar el sistema.

## Herramientas

Existen algunas herramientas que pueden ayudarnos en la tarea de asegurar un sistema o comprobar si un sistema está comprometido, como:

- Snort
- SATAN (Security Administrator Tool for Analyzing Networks)
- TIGER
- AIDE (Advanced Intrusion Detection Environment)
- . . .

## Enlaces de seguridad en Linux

- <http://www.cert.org/>
- <http://www.securityfocus.com/>
- <http://www.linuxsecurity.com/>
- <http://lsap.org/>
- <http://www.faqs.org/faqs/computer-security/>
- <http://www.insecure.org/>
- <http://isec.pl/>