Universidad Politécnica de Cartagena

Escuela Técnica Superior de Ingeniería de Telecomunicación

SEGURIDAD EN REDES

Práctica 2: Seguridad en el Sistema Operativo (I)

Creación de usuarios
Permisos
Listas de Control de Acceso (ACL)

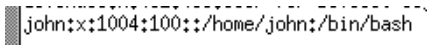
Autores: Josemaría Malgosa Sanahuja
María Dolores Cano Baños

SEGURIDAD EN REDES

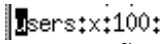
Seguridad en el Sistema Operativo PRÁCTICA 2 (primera parte)

IMPORTANTE: Para la realización de esta práctica será necesario trabajar con el usuario root (recordar que en la máquina virtual, la contraseña del usuario root es root)

CREACIÓN DE USUARIOS

1. Crear el usuario john con el comando `useradd -m john`. Verificar que el usuario se ha creado mirando el contenido del fichero /etc/passwd ¿Cuál es su user ID? Verificar que se ha creado el directorio /home/john 

ID= 1004

2. ¿Qué hace el comando `groups john`? Inspeccionando el contenido del archivo /etc/group determinar el group ID del usuario john ¿Para qué sirve el comando `id john`?
groups nos dice en que grupos está un usuario 

id muestra todas las ID relacionadas con ese usuario
3. Mediante el comando `passwd john` asignarle la contraseña `elloco`

4. De la misma forma, crear los usuarios paul y george y asignarles las contraseñas `elguapo` y `elmalo` respectivamente. Utilizando el comando `su <usuario>`, acceder al menos una vez al sistema utilizando el login de los tres usuarios creados ¿tienen algún tipo de archivos –aunque sean del tipo oculto- en su home? ¿A qué grupo pertenecen los usuarios paul y george? ¿Cuáles son los *group* ID de cada uno? Pertenecen a users con ID 100

5. ¿Qué hace el comando `groupadd cantantes`? Verificarlo inspeccionando el contenido del archivo /etc/group. Usando el comando `usermod -g`, asignar el grupo cantantes como grupo primario a los usuarios `john`, `paul` y `george`. Añade un grupo "cantantes"

6. Un usuario puede pertenecer a varios grupos. Crear los grupos `ritmica`, `bajo` y `solo`. Con el comando `usermod -a -G`, añadir el grupo secundario `ritmica` al usuario `john`, el grupo secundario `bajo` al usuario `paul`, y el grupo secundario `solo` al usuario `george` ¿Cómo podemos ver todos los grupos a los que pertenece un usuario? Con el comando groups "nombre"

7. Eliminar los usuarios john, paul y george con el comando `userdel -r`. Eliminar los grupos cantantes, bajo, ritmica y solo con el comando `groupdel`

PERMISOS

En Linux cada usuario tiene los siguientes atributos:

- Nombre de usuario
- UID: identificador único del usuario. Este identificador es asignado por el sistema operativo
- GID: identificador del grupo al que pertenece ese usuario. En Linux cada usuario puede pertenecer a uno o más grupos
- Contraseña: clave de acceso al sistema
- Directorio personal: ubicado dentro del directorio */home*

Los identificadores del usuario los usa el S.O. para decidir los privilegios o permisos que posee ese usuario a la hora de realizar una acción. Cada elemento del sistema de archivos, directorio o fichero tiene asociados unos permisos que describen quién puede utilizarlos. Sobre un fichero o directorio del sistema se puede realizar una o más de estas tres acciones:

- *read*: Leer el contenido del fichero o del directorio
- *write*: Modificar un fichero o escribir sobre un directorio
- *exec*: Ejecutar un archivo (si es posible), o acceder a un directorio (mediante *cd*)

Los ficheros y directorios tienen permisos concretos para cada una de estas acciones, para el propietario del archivo (a quién pertenece el archivo), para el grupo del fichero (grupo al que pertenece el propietario) y el resto de usuarios.

En Linux sólo permite que los usuarios realicen las acciones que los permisos atribuidos al archivo autorizan. Estos permisos definen la palabra de protección, que consiste en nueve bits repartidos en tres grupos de tres caracteres. Se visualiza al ejecutar un **`ls -l`** de la siguiente manera:

```
rwx rwx rwx
```

Empezando por la izquierda, el primer grupo de tres bits corresponde a los permisos del propietario del fichero. El siguiente a los permisos del grupo y el último al resto de usuarios. Si el bit está a uno, significa que está autorizado para hacer esa acción y si está a cero, no. Por ejemplo, si un archivo tiene la siguiente palabra:

```
rw-r-----
```

Equivale a 110-100-000, es decir, el propietario tiene permiso de lectura y escritura del archivo, el grupo solo de lectura y el resto de usuarios no tiene ningún permiso.

En la máquina remota existen los usuarios alberto, beatriz y carmen (sus contraseñas coinciden con el nombre de usuario). Conectarse como usuario `alberto` y responder a las siguientes preguntas: su alberto

8. ¿A qué grupo pertenecen los usuarios `alberto`, `beatriz` y `carmen`? profes

9. ¿Quién es el propietario de los archivos ubicados en `~/doc`? ¿A qué grupo pertenecen? ¿Cuáles son los permisos de los archivos doc_priv_alberto.txt y doc_pub_alberto.txt? ¿le parecen adecuados? El propietario es Alberto y pertenece al grupo profes Publico sí, privado debería ser rw-------

10. ¿Para qué sirve el comando `chmod`? En el directorio ~/bin ejecutar los siguientes comandos y mediante el comando `ls -l`, comprobar cuáles son los permisos que se modifican:
chmod sirve para cambiar los permisos de un archivo -rw-rw-r-- 1 alberto profes 142 Sep 14 11:00 exe_alb

```
chmod g+w exe_alb
ls -l exe_alb
```
-rw-rw-r-- 1 alberto profes 142 Sep 14 11:00 exe_alb

```
chmod o+w exe_alb
ls -l exe_alb
```
-rw-rw-rw- 1 alberto profes 142 Sep 14 11:00 exe_alb

```
chmod u="rw",g="rw",o="r" exe_alb
ls -l exe_alb
```
-rw-rw-r-- 1 alberto profes 142 Sep 14 11:00 exe_alb

```
chmod 644 exe_alb
ls -l exe_alb
```
-rw-r--r-- 1 alberto profes 142 Sep 14 11:00 exe_alb

```
chmod 751 exe_alb
ls -l exe_alb
```
-rwxr-x--x 1 alberto profes 142 Sep 14 11:00 exe_alb

11. ¿Qué indican los permisos cuando se refieren a un directorio? Establecer para cada directorio y archivo los permisos que le parezcan más idóneos

1. **Lectura (r):** En archivos Puede listar, copiar o visualizarlo. En Directorios pueden ver el contenido, se pueden listar a través del comando ls
2. **Escritura (w):** En archivos significa que se puede modificar o borrar el contenido, incluso puede modificar los permisos. En Directorios significa que puede crear, eliminar archivos y directorios dentro de ese directorio.
3. **Ejecución (x):** En archivos significa que se puede ejecutar el contenido. En Directorios significa que podemos entrar en la carpeta (comando cd).

LISTAS DE CONTROL DE ACCESO (ACL)

El sistema tradicional es insuficiente en una gran cantidad de situaciones, ya que solo permite que un fichero pertenezca a un único propietario. Por otra parte, aunque sí permite que un fichero tenga asignado varios grupos, solo se le pueden asignar permisos al grupo principal. Sin embargo, en muchas ocasiones es necesario que un archivo pertenezca a varios usuarios y grupos simultáneamente, cada uno con sus propios permisos de lectura (r), escritura (w) y ejecución (x).

Las ACL (*Access Control List*) permiten especificar los permisos de acceso a los ficheros con mucha mayor granularidad. Con las ACL, un fichero puede pertenecer a tantos usuarios y grupos como se desee. Para cada fichero el sistema asocia una lista con todos los usuarios y grupos a los que pertenece y los permisos asociados con ellos (`rwx`). De esta forma, se solventan muchas de las restricciones del sistema anterior

12. Verificar con el comando `tune2fs -l /dev/sda2` de la máquina virtual que la partición `/dev/sda2` ha sido formateada y montada para poder usar ACL

13. Cambiar al usuario `alberto` y situarse en su *home*. Tomar nota del propietario y grupo al que pertenece el archivo `acl.bin`, así como sus permisos asociados ¿Qué hace el comando `getfacl acl.bin`? En lo referente a los permisos de usuario ¿aporta la misma información que el comando `ls -l acl.bin`? `-rw-rw---- 1 alberto profes 71 Oct 19 20:23 acl.bin`

```
alberto@localhost:~> getfacl acl.bin
# file: acl.bin
# owner: alberto
# group: profes
user::rw-
group::rw-
other::---

alberto@localhost:~> ls -l acl.bin
-rw-rw---- 1 alberto profes 71 Oct 19 20:23 acl.bin
```

14. Ejecutar `setfacl -m u:beatriz:rw acl.bin` Volver a ejecutar el comando `getfacl acl.bin` Ejecutar de nuevo el comando `ls -l` ¿Cómo se aprecia el hecho de que ahora hay más usuarios que los tres tradicionales de Linux? A la vista del resultado del comando `ls`, ¿han cambiado los permisos del grupo *profes*? ¿y del comando `getfacl`? ¿Cuál de los dos cree que proporciona la información correcta?

La información correcta la proporciona "getfacl"

```
alberto@localhost:~> ls -l acl.bin
-rw-rw---- 1 alberto profes 71 Oct 19 20:23 acl.bin
alberto@localhost:~> setfacl -m u:beatriz:rw acl.bin
alberto@localhost:~> getfacl acl.bin
# file: acl.bin
# owner: alberto
# group: profes
user::rw-
user:beatriz:rw-
group::rw-
mask::rw-
other::---

alberto@localhost:~> ls -l acl.bin
-rw-rw----+ 1 alberto profes 71 Oct 19 20:23 acl.bin
```

Las ACL -por cuestiones de compatibilidad con aplicaciones muy antiguas- definen el concepto de máscara. La máscara representa los permisos que todos los usuarios y grupos añadidos podrán tener como máximo. Cuando se utilicen ACL, los 3 bits que antes indicaban los permisos del grupo por defecto, ahora indicarán la máscara del fichero (o directorio).

15. ¿Cuál es la máscara asociada al archivo? ¿Puede ahora el usuario beatriz modificar el contenido del archivo acl.bin? Verificarlo haciendo que beatriz ejecute el siguiente comando: `echo "nueva linea al final" >> acl.bin` `rw-`

16. Ejecutar `setfacl -m u:beatriz:rwx acl.bin` y ejecutar el comando `getfacl acl.bin`; a la vista del resultado ¿ha cambiado el valor de la máscara? ¿puede ahora el usuario beatriz ejecutar el archivo acl.bin? Ejecutar el comando `chmod 664 acl.bin` ¿ha cambiado el valor de la máscara? ¿puede ahora el usuario beatriz ejecutar el archivo acl.bin? ¿de qué otra forma se puede cambiar la máscara? el valor de la máscara es rwx no puede ejecutarlo

17. Ejecutar `setfacl -m u:carmen:--- acl.bin` y ejecutar el comando `getfacl acl.bin`; a la vista del resultado ¿puede ahora el usuario carmen (que pertenece al grupo *profes*) ver el contenido del archivo acl.bin? No puede ver el archivo

18. Ejecutar `setfacl -x u:carmen acl.bin` ¿Qué ha ocurrido? Se borra la configuración de Carmen

19. Ejecutar `setfacl --set u::rw,g::rw,o::-,u:alumno:r acl.bin` ¿Qué ha ocurrido?

```
alberto@localhost:~> getfacl acl.bin
# file: acl.bin
# owner: alberto
# group: profes
user::rw-
user:alumno:r--
group::rw-
mask::rw-
other::---
```

20. Ejecutar `setfacl -b acl.bin` ¿Qué ha ocurrido?
Se borran los ajustes.

Universidad Politécnica de Cartagena

Escuela Técnica Superior de Ingeniería de Telecomunicación

SEGURIDAD EN REDES

Práctica 2: Seguridad en el Sistema Operativo (II)

Funciones *hash* aplicadas a la autenticación de usuarios

Autores: Josemaría Malgosa Sanahuja
María Dolores Cano Baños

SEGURIDAD EN REDES

Funciones *hash* aplicadas a la autenticación de usuarios PRÁCTICA 2 (segunda parte)

El uso más frecuente de funciones *hash* es para firmar documentos y para autenticar usuarios. En esta práctica se ejemplificará el uso que hace el sistema operativo Linux de los *hash* DES, MD5, SHA-256 y SHA-512 para autenticar usuarios. En otra práctica se estudiará el uso de los *hash* para firmar documentos.

En Linux, el método de autenticación más común es mediante un *login* y un *password*. El sistema comprueba si el usuario existe buscando el *login* en el fichero */etc/passwd*. En caso de existir, calcula el *hash* del *password* introducido por el usuario y lo compara con el almacenado en el fichero */etc/shadow*. Solo en el caso de que coincidan, el usuario accede a su cuenta de Linux.

Estos sistemas sufren dos tipos de ataques:

- Ataque por fuerza bruta: Dado un alfabeto (por ejemplo [0-9A-Za-z] de 62 caracteres) y fijando la longitud de la contraseña a L caracteres, se trata de calcular el *hash* de todas los posibles combinaciones de tamaño L hasta encontrar una coincidencia con el *hash* almacenado en la base de datos (en nuestro caso, en */etc/shadow*)

- Ataque por diccionario: Existe una base de datos que almacena multitud de contraseñas utilizadas por usuarios reales, así como sus posibles variantes. Se trata de calcular el *hash* de cada una de ellas hasta encontrar una coincidencia con el *hash* almacenado en la base de datos (en nuestro caso, en */etc/shadow*)

1. ¿Cuál es la estructura del fichero */etc/passwd*? ¿Cuáles son los permisos del fichero */etc/passwd*?

2. ¿Cuál es la estructura del fichero */etc/shadow*? ¿Quién tiene acceso a este fichero? ¿Por qué no se acostumbra a almacenarse los *hash* de las contraseñas en */etc/passwd*?

Los algoritmos se utilizan para el cálculo del *hash* son DES, MD5, SHA-256 y SHA-512 (el DES es el realidad algoritmos de cifrado adaptados para que funcionen como *hash*). Todos estos algoritmos admiten SALT y ROUND.

3. ¿Qué es un SALT y un ROUND? ¿Qué algoritmos admiten ROUND? ¿Cuál es el formato con el que se almacenan los *hash* en el archivo */etc/shadow*?

El programa *makepasswd.c* (ver anexo) calcula el *hash* de la contraseña que se le proporcione. Estudiar el código de dicho programa y responder razonadamente a las siguientes preguntas:

4. ¿Qué opciones admite el programa? ¿Es necesario que el usuario especifique un SALT?

5. ¿Qué utilidad tienen los SALT y ROUND? Si se tiene acceso al archivo */etc/shadow*, ¿proporcionan los SALTS y ROUNDS mayor seguridad?

6. ¿Cuál debería ser el valor del *hash* del usuario *alumno* en el fichero */etc/shadow* si quisiéramos que la contraseña fuera lssei? ¿Qué hace el comando *passwd*?

7. ¿Cree que es factible disponer del fichero */etc/shadow* de un ordenador del cuál usted no tiene permisos de super-usuario? ¿Cree que es factible saber la longitud aproximada de una contraseña?

Suponga que el *hash* de una determinada contraseña es “OQoOcHTS.8Eiw” (es decir, del tipo DES). Supóngase que también se sabe que la contraseña está compuesta por 4 caracteres (letras mayúsculas, minúsculas y dígitos numéricos):

8. Utilizando el programa *crack.c* (ver anexo II) encontrar la contraseña asociada al *hash* anterior mediante un ataque por fuerza bruta ¿Cuánto tiempo ha tardado el ordenador en encontrarla? ¿Existe alguna otra contraseña de longitud 4 que tenga el mismo *hash*?

9. El programa *hashtime.c* (ver anexo III) utiliza la función *gettimeofday* para medir con exactitud el tiempo que el programa tarda en calcular el *hash* del tipo DES, MD5, SHA-256 y SHA-512 de una contraseña de L caracteres y compararlo con un patrón conocido (ésta es precisamente la función básica de un ataque por fuerza bruta). Suponiendo L = {4, 6, 8} hacer 1.000 medidas de cada caso y tomar nota del valor medio ¿Depende dicho tiempo de la longitud de la clave?

10. Utilizando el resultado del apartado anterior y suponiendo que el alfabeto es de 80 caracteres, calcular la expresión general del tiempo medio necesario para encontrar una contraseña de L caracteres por fuerza bruta. Evaluar dicha expresión para cada algoritmo con L = {4,6, 8}

11. Si dispusiera de *k* ordenadores trabajando en paralelo ¿podría reducirse dicho tiempo? ¿Cuántos ordenadores necesitaría para encontrar una contraseña SHA-256 de seis dígitos en un tiempo razonable? Sacar conclusiones

1. -rw-r--r-- 1 root root 2008 Oct 19 19:24 passwd

```
<nombre>,<password><uid>, <gid>,<descripción opcional> <carpeta>, <shell>
```

<nombre> No se admiten números al comienzo de un nombre de usuario.

<password> Una «**x**» indica que el password está almacenado en /etc/shadow, en el caso de ser una «**!**» es que el usuario está bloqueado. Si tiene «**!!**» es que no tiene.

<uid> Cada usuario lleva un no identificador (uid) entre 0(root) y 65535. Se reservan algunos para usuario root(el cero siempre), y para usuarios de servicios varios del sistema.

Red hat y derivados entre 1 y 499.
Debian y derivados entre 1 y 999.

<gid> – grupo id, cada usuario tiene un id de grupo principal, pero puede pertenecer a más grupos.

<carpeta> La usará como la carpeta de inicio del usuario, al iniciar sesión con él será la que cargue por defecto.

<shell> Los usuarios de servicios y usuarios con permisos limitados no deben tener shell, es decir iniciar sesión en consola, normalmente se les deja con /usr/bin/nologin o /bin/false

2. -rw-r----- 1 root shadow 1294 Oct 19 19:24 shadow

```
<nombre><password cifrado><1><2><3><4><5><6>
```

<nombre> Nombre del usuario.

<password cifrado> Pues eso…

- Codificación del campo <password>:

SALT+HASH (DES)
$ALGORITMO$SALT$HASH (resto)

1-Días transcurridos desde 1-1-1970 donde el password fue cambiado por última vez.
2-El mínimo número de días entre cambios de contraseña.
3-Días máximos de validez de la cuenta.
4-Días que avisa antes de caducar la contraseña.
5- Días después de que un password caduque para deshabiltar la cuenta
6- Fecha de caducidad. días desde 1-1-1970, donde la cuenta es deshabilatada y el usuario no podrá iniciar sesión

Las contraseñas cifradas no se almacenan en *passwd*, ya que este archivo puede ser leído por cualquier usuario y con programas de descifrado se pueden descubrir

3. El formato del hash en *shadow* es con ROUND (depende del algoritmo de cifrado).

- SALT: Es un número de dígitos aleatorios que se le agrega a la contraseña ya sea al principio o al final y que el usuario no conocerá. Con esto, la contraseña se hace de mayor longitud y por lo tanto más compleja, de esta manera será mucho más difícil encontrar el hash en una tabla y más aún obtener la contraseña, ya que está combinada con el salt.
- ROUND: Consiste en repetir el SALT n veces.

4. No es necesario que el usuario especifique un SALT.

```
localhost:~/Downloads # ./makepasswd
Password must be specified

Usage: makepasswd [OPTIONS]... password
Crypts the PASSWORD using crypt(3).

-m, --method=TYPE     select method TYPE
-s, --salt=SALT       use the specified SALT
-r, --rounds=NUMBER   use the specified NUMBER of rounds
-h, --help            display this help and exit
-v, --version         output version information and exit
```

5. Sí
6. El comando *passwd* sirve para cambiar la contraseña del usuario.

```
localhost:~/Downloads # ./makepasswd -m sha-512 lssei
$6$Qb6xy05/qtSRj$p5Xo2LBu4bQ2A/PTEmgCHloIxk7vLcZhu8dJRwtizdFJP4AL.WANDKdZW/ukwCWr9zR15VCIFo4VXh4DfRU/N/
```

7. No se puede estimar la longitud de una contraseña. Ni tener el archivo /etc/shadow sin ser super-usuario.
8. La contraseña es "*paco*"

```
localhost:~/Downloads # time ./crack -s OQ -l 4 -h OQoOcHTS.8Eiw
password found = paco

real    0m18.556s
user    0m11.580s
sys     0m6.644s
```

9.

```
localhost:~/Downloads # ./hashtime -l 4 -p 1000 -m des
des mean value: 0.002629 miliseconds
localhost:~/Downloads # ./hashtime -l 6 -p 1000 -m des
des mean value: 0.003494 miliseconds
localhost:~/Downloads # ./hashtime -l 8 -p 1000 -m des
des mean value: 0.00346 miliseconds

localhost:~/Downloads # ./hashtime -l 4 -p 1000 -m md5
md5 mean value: 0.158895 miliseconds
localhost:~/Downloads # ./hashtime -l 6 -p 1000 -m md5
md5 mean value: 0.15626 miliseconds
localhost:~/Downloads # ./hashtime -l 8 -p 1000 -m md5
md5 mean value: 0.153476 miliseconds

localhost:~/Downloads # ./hashtime -l 4 -p 1000 -m sha-256
sha-256 mean value: 3.33172 miliseconds
localhost:~/Downloads # ./hashtime -l 6 -p 1000 -m sha-256
sha-256 mean value: 3.33976 miliseconds
localhost:~/Downloads # ./hashtime -l 8 -p 1000 -m sha-256
sha-256 mean value: 3.67951 miliseconds

localhost:~/Downloads # ./hashtime -l 4 -p 1000 -m sha-512
sha-512 mean value: 2.83639 miliseconds
localhost:~/Downloads # ./hashtime -l 6 -p 1000 -m sha-512
sha-512 mean value: 2.75942 miliseconds
localhost:~/Downloads # ./hashtime -l 8 -p 1000 -m sha-512
sha-512 mean value: 2.9751 miliseconds
```

10. $T = L^{80} \cdot media$
    - DES
        - $L = 4 \rightarrow T = 4^{80} \cdot 0.002629 = 3.842D + 45$
        - $L = 6 \rightarrow T = 6^{80} \cdot 0.003494 = 6.243D + 59$
        - $L = 8 \rightarrow T = 8^{80} \cdot 0.00346 = 6.113D + 69$
    - MD5
        - $L = 4 \rightarrow T = 4^{80} \cdot 0.158895 = 2.322D + 47$
        - $L = 6 \rightarrow T = 6^{80} \cdot 0.15626 = 2.792D + 61$
        - $L = 8 \rightarrow T = 8^{80} \cdot 0.153476 = 2.712D + 71$
    - SHA-256
        - $L = 4 \rightarrow T = 4^{80} \cdot 3.33172 = 4.869D + 48$
        - $L = 6 \rightarrow T = 6^{80} \cdot 3.33976 = 5.968D + 62$
        - $L = 8 \rightarrow T = 8^{80} \cdot 3.67915 = 6.500D + 72$
    - SHA-512
        - $L = 4 \rightarrow T = 4^{80} \cdot 2.83639 = 4.145D + 48$
        - $L = 6 \rightarrow T = 6^{80} \cdot 2.75942 = 4.931D + 62$
        - $L = 8 \rightarrow T = 8^{80} \cdot 2.9751 = 5.257D + 72$
11. $1.892D + 56$ para tardar 1 año